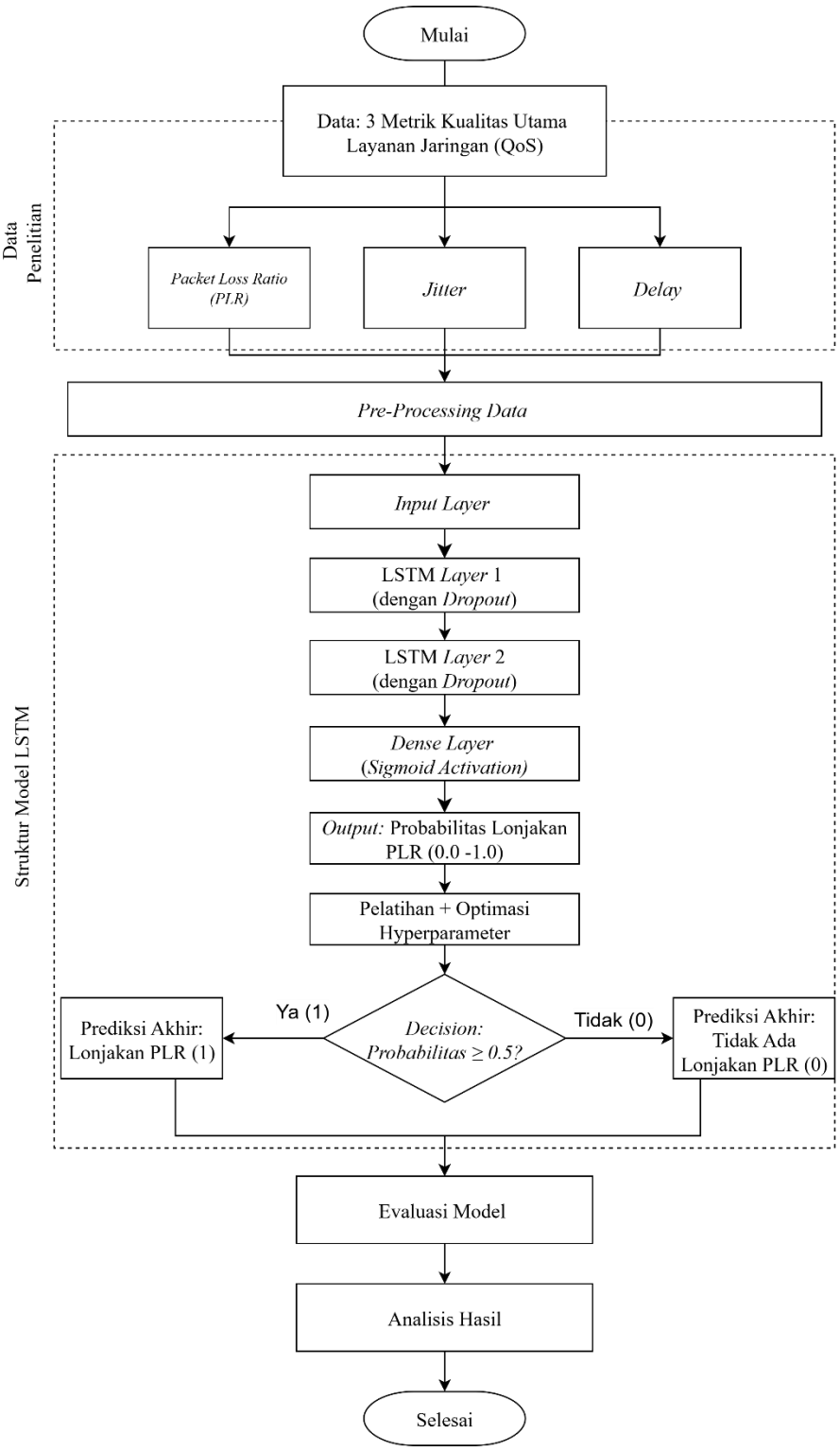
BAB III
METODOLOGI PENELITIAN

3.1 Objek Penelitian

Penelitian ini merupakan penelitian kuantitatif dengan melakukan pemanfaatan *Artificial Intelligence* (AI) melalui pendekatan Metode *Deep Learning*, yang berfokus pada permasalahan lonjakan *Packet Loss Ratio* (PLR) di jaringan telekomunikasi, Telkomsel Lampung. Data yang digunakan adalah *data time series*, meliputi metrik kualitas layanan jaringan (QoS), seperti *Packet Loss Ratio* (PLR), *Jitter* (Ms), *Delay* (Ms). Data tersebut dikumpulkan dari berbagai *Target Device Name* yang berada di kota Bandar Lampung. Penelitian ini bertujuan untuk memprediksi lonjakan PLR guna meningkatkan kualitas layanan pada jaringan Telkomsel Lampung.

3.2 Metodologi Penelitian

Metodologi penelitian ini menguraikan tahapan-tahapan sistematis yang dilakukan untuk mencapai tujuan penelitian, yaitu mengembangkan model prediksi lonjakan *Packet Loss Ratio* (PLR) menggunakan metode *Deep Learning*, khususnya *Long Short-Term Memory* (LSTM). Proses penelitian diawali dengan akuisisi data sekunder, yang kemudian dilanjutkan dengan tahapan *pre-processing* data secara komprehensif guna memastikan data siap digunakan dalam proses pemodelan. Selanjutnya, model *Deep Learning*, khususnya arsitektur LSTM, dibangun dan dilatih hingga menghasilkan model akhir. Tahap selanjutnya mencakup evaluasi model serta analisis hasil untuk menilai performa dan efektivitas model dalam memprediksi PLR. Melalui penelitian ini, diharapkan dapat dihasilkan solusi prediktif yang mampu meningkatkan efisiensi pemantauan kualitas jaringan telekomunikasi secara proaktif. Rangkaian tahapan penelitian tersebut digambarkan dalam Diagram Alir Tahapan Penelitian Gambar 3.1.



Gambar 3. 1 Diagram Alir Tahapan Penilitian.

3.2.1 Data Penelitian

Data yang digunakan dalam penelitian ini terdiri atas tiga metrik utama Kualitas Layanan Jaringan (QoS) yang merepresentasikan kualitas jaringan, yaitu *Packet Loss Ratio* (PLR), *Delay*, dan *Jitter*. Selain itu, informasi waktu (jam dan hari) juga digunakan sebagai variabel pendukung.

1. *Packet Loss Ratio* (PLR)

   Packet Loss Ratio merupakan perbandingan antara jumlah paket yang hilang dengan total paket yang dikirimkan. Parameter ini digunakan sebagai fokus utama penelitian karena mencerminkan kualitas transmisi data dalam jaringan.
2. *Jitter*

   Jitter adalah variasi waktu kedatangan paket. Nilai *jitter* yang tinggi dapat menurunkan kualitas layanan, terutama pada aplikasi real-time seperti *video conference* atau *VoIP*.
3. *Delay*

   Delay atau latensi adalah waktu yang dibutuhkan paket untuk berpindah dari sumber ke tujuan. Nilai *delay* yang besar dapat memengaruhi kualitas pengalaman pengguna (*Quality of Experience*).

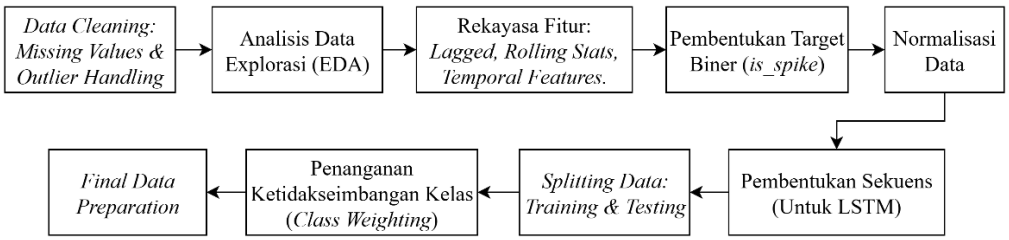
Tahap ini merupakan langkah awal dalam pelaksanaan penelitian, yaitu proses pengumpulan data yang digunakan sebagai dasar dalam melakukan analisis. Data yang digunakan merupakan data sekunder yang diperoleh dari database internal perusahaan Telkomsel, sistem monitoring jaringan, serta laporan kinerja jaringan yang terdokumentasi.

Data dikumpulkan untuk periode Maret hingga Desember 2024, dengan interval waktu per jam. Adapun variabel-variabel yang digunakan dalam penelitian ini dirangkum pada Tabel 3.1. berikut:

Tabel 3. 1. Data Kualitas Layanan (QoS).

Data Kualitas Layanan Jaringan		
Nama Variabel	Keterangan	Kategori Variabel
Lonjakan *Packet Loss Ratio* (%)	Representasi biner dari PLR: 0 = Normal 1 = Lonjakan	Variabel Dependen (Target)
Packet Loss Ratio	Nilai lagged dan statistik bergerak.	Variabel Independen (Fitur)
Delay (Ms)	Nilai aktual, lagged, dan sattistik bergerak.	
Jitter (Ms)	Nilai aktual, lagged, dan statistic bergerak	
Hour (Jam)	Wilayah dalam format jam (0-23)	
Day Of Week (Hari dalam Minggu)	Representasi hari: Senin = 0, Minggu = 6.	

3.2.2 *Pre-Processing Data*



Gambar 3. 2 Diagram Alir Tahap *Pre-Processing Data*

Pre-processing data merupakan tahap penting untuk memastikan data yang digunakan dalam pemodelan berada dalam kondisi bersih, konsisten, dan siap olah. Gambar 3.2. menunjukkan alur *pre-processing* yang terdiri dari beberapa langkah,

dimulai dari pembersihan data, eksplorasi data, rekayasa fitur, pembentukan target, normalisasi, pembentukan sekuens, pembagian data, hingga penanganan ketidakseimbangan kelas. Setiap tahapan bertujuan untuk meningkatkan kualitas data sebelum digunakan dalam pelatihan dan pengujian model LSTM. Penjelasan rinci disajikan pada subbab berikut.

3.2.2.1 Data Cleaning (Pembersihan Data)

Tahapan ini adalah tahap membersihkan dan mempersiapkan data agar lebih akurat serta konsisten ketika digunakan untuk membuat pemodelan.

1. Penangganan *Missing Vakues*

   Nilai yang hilang atau kosong (*missing value)* pada kolom numerik seperti *Packet Loss Ratio, Delay,* dan *Jitter* akan diisi menggunakan metode interpolasi linier. Apabila masih terdapat *missing value* pada awal atau akhir nilai deret waktu, akan diisi dengan metode *forward fill* dan *backward fill*.

2. Penanganan *Outlier* (*Outlier Handling)*

   Permasalahan outlier akan dideteksi menggunakan metode *Interquartile Range* (IQR) dan ditangani dengan teknik *Winsorizationg*.

   - Nilai di bawah Q1 – 1.5 * IQR akan diklip ke *lawer_bound*.
   - Nilai diatas Q3 + 1.5 * IQR akan diklip ke *upper_bound*. Khusus nilai *Packet Loss* akan diklip antara 0 dan 1.

3.2.2.2 Eksploarasi Data Analisis *(EDA)*

Tahapan ini merupakan proses yang melibatkan visualisasi serta ringkasan statistik data untuk memahami karakteristik, distribusi, pola, dan hubungan antar variabel yang terdapat dalam dataset. Proses ini sangat penting untuk mengidentifikasi tren, pola musiman (seasonality), serta korelasi antar variabel yang dapat dimanfaatkan dalam proses

rekayasa fitur (feature engineering) dan tahap pemodelan prediktif selanjutnya.

3.2.2.3 Rekayasa Fitur *(Feature Engineering)*

1. Fitur Lagged: Membuat fitur baru berdasarkan metrik kualitas layanan jaringan dari periode waktu sebelumnya. Fitur ini penting digunakan untuk menangkap dependensi temporal dalam data deret waktu. Contoh fitur yang akan dibuat adalah:
    - *PLR_lag1.*
    - *PLR_lag24* (24 jam sebelumnya)
    - *PLR_lag165* (1 minggu sebelumnya).
    - *Delay_lag1.*
    - *Jitter_lag1*.
2. Statistik Bergerak (Rolling Statistics): Menghitung rata-rata bergerak dan standar deviasi bergerak untuk periode waktu tertentu, contohnya adalah:
    - *PLR_rolling_mean_3h*.
    - *PLR_rolling_std_3h.*
    - *Delay_rolling_mean_3h*.
    - *Jitter_rolling_mean_3h*.
3. Fitur Temporal: Melakukan ekstraksi informasi waktu seperti '*Hour'* (jam dalam sehari), dan '*DayOfWeek'* (hari dalam seminggu) dari data *Timestamp* untuk menangkap pola musiman harian dan mingguan.

3.2.2.4 Pembentukan Target Biner *(Binaru Tagert Formation)*

Tahapan ini akan mengubah masalah prediksi nilai kontinu menjadi masalah klasifikasi biner. Nilai yang akan dikonversi menjadi variabel target *'is_spike'* adalah *Packet Loss Ratio*, contohnya:

$$Is\_spike = \begin{cases} 1, & jika\ PLR \geq 0{,}005 \\ 0, & jika\ PLR < 0{,}005 \end{cases}$$

3.2.2.5 Normalisasi Data *(Data Normalization)*

Tahapan ini merupakan proses transformasi data dengan tujuan untuk menyelaraskan skala setiap variabel agar lebih seragam dan sesuai dengan kebutuhan pemodelan. Normalisasi sangat penting dalam metode *Deep Learning* karena algoritma ini sensitif terhadap skala fitur, sehingga normalisasi dapat membantu proses pembelajaran model menjadi lebih stabil dan konvergen lebih cepat.

Pada penelitian ini, metode normalisasi yang digunakan adalah *Min-Max Scaling*, yaitu teknik yang mengubah nilai setiap fitur ke dalam rentang 0 hingga 1, sesuai dengan rumus berikut:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

3.2.2.6 Splitting Data

Tahapan ini merupakan tahapan dilakukannya pembagian data menjadi 2 bagian secara temporal (berdasarkan waktu), yaitu:

1. Data *Training* (80%)
2. Data *Testing* (20%)

Proses ini digunakan untuk memastikan bahwa model akan dievaluasi dengan data baru secara kronologis.

3.2.2.7 Penanganan Ketidakseimbangan Kelas *(Class Imbalance Handling)*

Tahapan ini dilakukan untuk mengatasi permasalahan ketidakseimbangan data, yaitu ketika jumlah kejadian lonjakan *Packet Loss Ratio* (PLR) yang dilabeli sebagai kelas 1 (*is_spike* = 1) jauh lebih sedikit dibandingkan kondisi normal (kelas 0). Ketidakseimbangan ini dapat memengaruhi kinerja model, khususnya dalam mendeteksi kelas minoritas, karena model cenderung lebih akurat dalam memprediksi kelas mayoritas.

Untuk mengatasi hal tersebut, penelitian ini menerapkan strategi penanganan ketidakseimbangan dengan menggunakan parameter *class_weight*. Pendekatan ini memberikan bobot yang lebih besar pada kelas minoritas, sehingga model akan lebih memperhatikan pola-pola yang terdapat pada kelas lonjakan. Dengan demikian, diharapkan nilai *recall* dalam mendeteksi kejadian lonjakan dapat meningkat secara signifikan.

3.2.2.8 Final Data Preparation

Tahapan ini merupakan tahap akhir dalam proses pre-processing data sebelum masuk ke tahap pemodelan. Setelah melalui proses pembersihan, normalisasi, dan penanganan ketidakseimbangan data, dataset dianggap telah siap untuk digunakan dalam pembangunan model prediktif.

3.2.3 *Long Short-Term Memory* (LSTM)

Model *Long Short-Term Memory* (LSTM) digunakan dalam penelitian ini untuk memprediksi kemungkinan terjadinya lonjakan *Packet Loss Ratio* (PLR). LSTM dipilih karena kemampuannya dalam mempelajari pola jangka panjang pada data deret waktu (*time series*), sehingga relevan untuk menganalisis dinamika metrik kualitas layanan jaringan (*Quality of Service* / QoS).

Gambar 3.1 menggambarkan arsitektur model LSTM yang digunakan dalam penelitian ini. Model dirancang untuk memproses data *time series* yang telah melalui tahap *pre-processing*, sehingga data siap digunakan sebagai *input* dalam bentuk *window_size* dan *num_features*, yang merepresentasikan jumlah langkah waktu dan jumlah fitur pada setiap langkah.

Arsitektur model LSTM dibangun dengan tahapan sebagai berikut:

1. Lapisan Input

   Lapisan ini menerima data sekuensial berupa deret waktu yang telah diproses sebelumnya. Data terdiri atas sejumlah langkah waktu (*time steps*) dan fitur yang mewakili metrik QoS.
2. Lapisan LSTM Pertama

   Data dari lapisan input diproses pada lapisan LSTM pertama yang bertugas menangkap pola-pola jangka panjang dalam *data time series*. Untuk mencegah risiko *overfitting*, lapisan ini dilengkapi dengan mekanisme *dropout*.
3. Lapisan LSTM Kedua

   Hasil keluaran dari lapisan LSTM pertama kemudian diteruskan ke lapisan LSTM kedua yang juga dilengkapi dengan *dropout*. Penambahan lapisan kedua ini bertujuan memperdalam kemampuan model dalam memahami dinamika temporal data jaringan yang lebih kompleks.
4. Lapisan *Dense* (*Fully Connected Layer*)

   Lapisan *dense* berfungsi mengintegrasikan informasi yang telah diekstraksi oleh lapisan LSTM. Lapisan ini menghasilkan representasi akhir sebelum dilakukan proses klasifikasi.
5. Lapisan Output (*Sigmoid Layer)*

   Lapisan ini menggunakan fungsi aktivasi sigmoid untuk menghasilkan nilai probabilitas dengan rentang antara 0 hingga 1. Nilai ini merepresentasikan kemungkinan terjadinya lonjakan *Packet Loss Ratio* pada jaringan.
6. Tahap Keputusan (*Decesion Rule*)

   Nilai probabilitas dari fungsi aktivasi sigmoid selanjutnya dibandingkan dengan nilai ambang batas (*threshold*) yang telah ditentukan. Hasilnya berupa prediksi biner:

- **1** → Jika nilai probabilitas ≥ 0,5 maka diprediksi sebagai kelas “lonjakan”.
- **0** → Jika nilai probabilitas < 0,5 maka diprediksi sebagai kelas “normal”.

Dengan demikian, model LSTM dalam penelitian ini berfungsi sebagai sistem klasifikasi biner yang memberikan peringatan dini mengenai potensi lonjakan PLR, sehingga dapat digunakan sebagai dasar dalam pengambilan keputusan manajemen jaringan.

3.2.4 Pelatihan Model dan Optimasi Hyperparameter

Tahap ini akan berfokus pada pelatihan model LSTM dan optimasi Hyperparameter untuk mencapai kinerja terbaik model. Proses ini dibagi menjadi dua bagian utama, yaitu:

1. *Hyperparameter Tuning*

   Pada tahap ini dilakukan proses pencarian konfigurasi hyperparameter terbaik dengan menggunakan algoritma *Hyperband*. Pemilihan algoritma ini didasarkan pada efisiensinya dalam menjelajahi ruang pencarian *hyperparameter* dengan keterbatasan sumber daya komputasi.

   Optimasi dilakukan dengan tujuan memaksimalkan nilai *val_recall* pada kelas *lonjakan*, sehingga model diharapkan mampu mendeteksi kejadian lonjakan *Packet Loss Ratio* secara lebih akurat.

   Untuk mendukung proses tuning, digunakan beberapa teknik regulasi sebagai berikut:

   - *EarlyStopping*, yang berfungsi menghentikan proses pelatihan secara otomatis apabila performa model pada data validasi tidak menunjukkan peningkatan setelah sejumlah *epoch* tertentu.

- *ReduceLROnPlateau*, yang berfungsi menurunkan laju pembelajaran (*learning rate*) secara adaptif ketika performa model mengalami stagnasi, sehingga dapat membantu mempercepat konvergensi dan mencegah *overfitting*.

2. Pelatihan Model Akhir

   Setelah diperoleh konfigurasi *hyperparameter* terbaik dari hasil tuning, tahap berikutnya adalah pelatihan ulang model LSTM dengan konfigurasi optimal tersebut. Pada tahap ini, seluruh data pelatihan digunakan kembali untuk membangun model akhir (*final model*).

3.2.5 Evaluasi Model

Model yang telah selesai dilatih akan dievaluasi menggunakan data *testing* yang belum pernah dilihat oleh model selama pelatihan berlangsung. Evaluasi bertujuan untuk mengukut kinerja model dalam memprediksi lonjakan *Packet Loss Ratio* pada data baru. Metrik evalusi yang digunakan meliputi:

1. *Accuracy*: Mengukur proporsi prediksi yang benar secara keseluruhan.

   Rumus :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

   Keterangan:

   - TP = *True Positive*
   - TN = *True Negative*
   - FP = *False Positive*
   - FN = *False Negative*

2. *Precision* (untuk kelas ’lonjakan’): Mengukur seberapa akurat prediksi lonjakan yang dibuat model.

   Rumus*:*

$$Precision = \frac{TP}{TP + FP}$$

Keterangan:

- TP = *True Positive*
- FP = *False Positive*

3. *Recall* (untuk kelas 'lonjkan'): Mengukur seberapa banyak lonjakan aktual yang berhasil dideteksi oleh model, evaluasi ini menjadi prioritas utama dalam penelitian ini.

   Rumus:

$$Recall = \frac{TP}{TP + FN}$$

   Keterangan:

   - TP = *True Positive*
   - FN = *False Negative*

4. *F1-Score* (untuk kelas 'lonjakan'): Rata-rata harmonik dari *Precision* dan *Recall*, memberikan keseimbangan pada data tidak seimbang.

   Rumus:

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

5. *ROC AUC (Receiver Operating Characteristic – Area Under the Curve)*: Mengukur kemampuan model dalam membedakan antara kelas positif dan negatif. Nilai AUC berkisar antara 0 dan 1, dimana semakin tinggi semakin baik.

   Rumus:

$$AUC = \text{Luas di bawah kurva } ROC \text{ (TPR vs FPR)}$$

   - TPR (*True Positive Rate*) = TP / (TP + FN) = *Recall*
   - FPR (*False Positive Rate*) = FP / (FP + TN)

Metrik evaluasi ini tidak memiliki rumus langsung, tapi dihitung menggunakan kurva TPR (*True Positive Recall)* vs FPR (*False Positive Recall*).

6. *Confusion Matric*: Representasi visual dari hasil klasifikasi yang menyajikan jumlah prediksi yang benar dan salah berdasarkan label aktual dan prediksi model. melihat distribusi prediksi model terhadap kelas aktual, sehingga memudahkan analisis kesalahan klasifikasi biner. Tabel 3.2 menunjukkan bentuk umum dari *confusion matrix*.

Tabel 3. 2. Ringkasan Dari Hasil Klasifikasi.

	Prediksi = 1	Prediksi = 0
Aktual = 1	TP	FN
Aktual = 0	FP	TN

Dengan contoh nilai:

- TP = 80, FP = 20
- FN = 50, TN = 900

7. Analisis Ambang Batas Optimal: Analisis ambang batas dilakukan dengan mencari nilai threshold yang memberikan nilai *F1-Score* tertinggi, sehingga keseimbangan antara *Precision* dan *Recall* dapat tercapai.

Rumus:

$$F1 = \frac{2 \cdot (P \cdot R)}{P + R}$$

Keterangan:

- P = *Precision*
- R = *Recall*

Selain metrik evaluasi yang telah dijelaskan sebelumnya, penelitian ini juga mempertimbangkan analisis terhadap kesalahan klasifikasi berupa *False Positive* (kesalahan tipe I) dan *False Negative* (kesalahan tipe II). *False Positive* merupakan kondisi ketika model memprediksi terjadinya lonjakan padahal kenyataannya tidak terjadi, sedangkan *False Negative* merupakan kondisi ketika model gagal mendeteksi lonjakan yang sebenarnya terjadi. Pada penelitian ini, pengurangan *False Negative* diprioritaskan karena terlewatnya kejadian lonjakan dapat memberikan dampak yang lebih signifikan terhadap kualitas layanan jaringan.

3.2.6 Analisa Hasil

Tahap ini melibatkan interpreteasi terhadap metrik evaluasi yang telah diperoleh, serta dilakukannya analisis kirtis terhadap kinerja model termasuk kekuatan dan kelemahan model dalam memprediksi adanya lonjakan PLR.

3.3 Alat Dan Perangkat Lunak Yang Digunakan

Dalam penelitian ini terdapat alat dan perangkat lunak yang digunakan untuk implementasi metodoloi. Pada tahap awal dipertimbangkan penggunaan perangkat lunak *data mining* GUI seperti Orange dan RapidMiner sesuai saran dosen penguji. Namun setelah tahap eksplorasi awal ditemukan bahwa kedua aplikasi tersebut kurang kompatibel pada jenis penelitian ini karena adanya keterbatasan dalam:

- Penanganan spesifik data deret waktu yang kompleks, seperti pembentukan sekuens 3D untuk LSTM.
- Fleksibiliras pembangunan arsitektur *Deep Learning*, seperti LSTM dengan *layer* dan *dropout*.
- Implementasi *Hyperparameter Tuning* yang canggih
- Penanganan ketidakseimbangan kelas yang spesifik terhadap model *Deep Learning.*

Oleh karena itu, penelitian ini sepenuhnya diimplementasikan menggunakan lingkungan program yang fleksibel.

3.3.1 Python

Python adalah bahasa prmorgraman utama yang akan digunakan untuk implementasi dalam model LSTM.

3.3.2 Google Colab

Google Colab adalah lingkungan *cloud-based* yang menyediakan akses ke GPU. Lingkungan ini digunakan untuk menjalankan *script* Python dan melatih model *Deep Learning* secara efesien.

3.3.3 Pustaka Python

a. *Pandas*: Untuk manipulasi data dan analisis data.
b. *Numpy*: Untuk operasi numerik.
c. *Scikit-Learn*: Untuk tahap *pre-processing data*.
d. *TensorFlow/Keras*: Untuk membangun, melatih dan mengevaluasi model LSTM.
e. *Keras Tuner*: Untuk melakukan otomatisasi *Hyperparameter Tuning.*
f. *Matplotlib & Seaborn*: Untuk melakukan visualisasi data dan hasil.
g. *Statsmodels*: Untuk membuat plot ACF dan PACF.