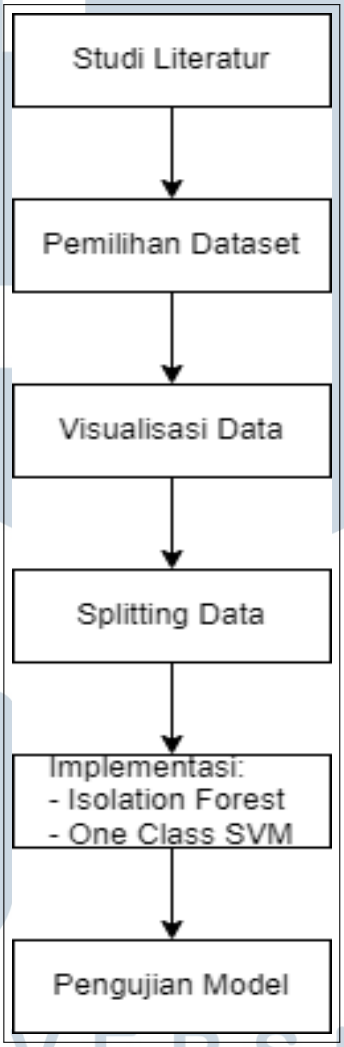
BAB 3
METODOLOGI PENELITIAN

Penelitian akan dilakukan secara berurutan sesuai dengan bagan yang tertera pada gambar 3.1.



Gambar 3.1. Bagan Metodologi Penelitian

3.1 Studi Literatur

Studi literatur dilakukan dengan tujuan untuk mengidentifikasi konsep-konsep terkait dengan deteksi serangan DDoS, penggunaan algoritma *Isolation Forest* dan *One-Class Support Vector Machines* (OCSVM) untuk deteksi anomali. Informasi dari literatur membantu dalam merancang kerangka kerja penelitian dan

menentukan metode yang paling relevan. Tabel 3.1 berikut meliputi penelitian yang digunakan dalam proses.

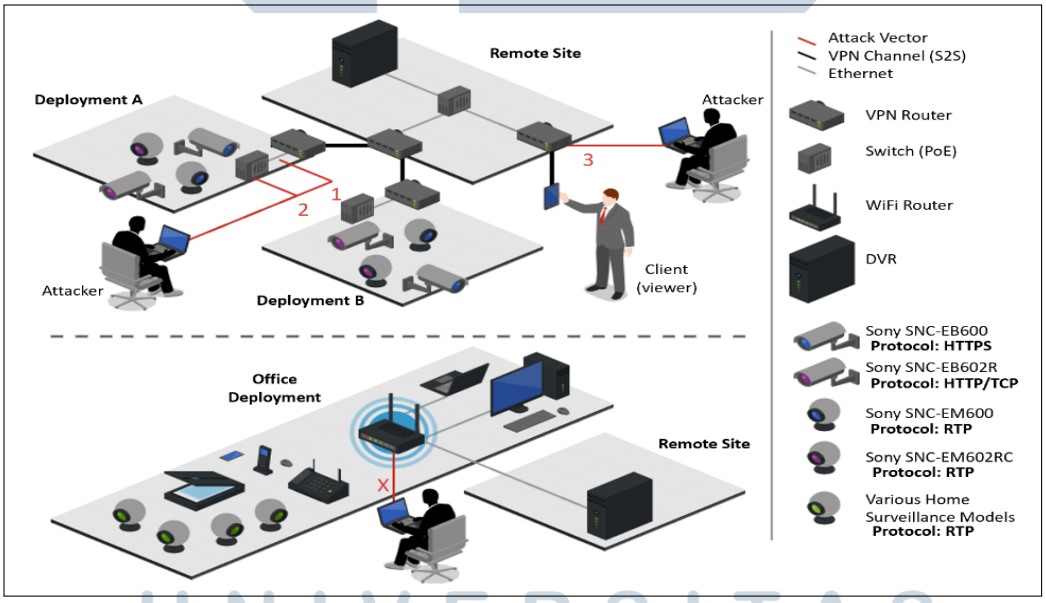
Tabel 3.1. Tabel Studi Literatur

Referensi	**Metode**	**Deskripsi**
[6]	*Firewall*	Membangun *firewall agnostic Early Detection Module* (EDM) menggunakan pola permintaan HTTP sebagai tanda tangan untuk serangan AL-DDoS.
[7]	LSTM/SVM	Mengidentifikasi perilaku mekanis abnormal yang ditangkap oleh pengukuran getaran menggunakan penggabungan LSTM dan SVM.
[8]	*Deep Isolation Forest*	Mengembangkan *Isolation Forest* menggunakan jaringan saraf untuk memetakan data asli kemudian diterapkan untuk melakukan partisi data.
[9]	*SVM*	Membangun model menggunakan SVM yang dibantu dengan *Kernel Principal Component Analysis* (KPCA) dan *Genetic Algorithm* (GA) untuk deteksi serangan DDoS.
[10]	*Isolation Forest* dan *K-Means*	Deteksi serangan DDoS berdasarkan kombinasi algoritma *Isolation Forest* dan *K-Means*.
[22]	*One-Class SVM*	Deteksi *IoT Botnet* menggunakan *One-Class SVM* dengan algoritma *Gray Wolf Optimization* (GWO) untuk mengoptimalkan *hyperparameter*.

[24]	*Isolation Forest*	Deteksi anomali pada jalur gas turbin menggunakan algoritma Isolation Forest

3.2 Pemilihan Dataset

Dataset yang digunakan adalah *Kitsune Attack Network Dataset* dan diambil dari *https://www.kaggle.com/datasets/ymirsky/network-attack-dataset-kitsune/data*[14]. Dataset yang dipilih mencakup kondisi normal dan serangan DDoS digunakan untuk melatih dan menguji model. Dataset ini berisikan data dari 9 jenis serangan. Data diambil dari sistem pengawasan komersial berbasis IP atau jaringan yang penuh dengan perangkat IoT. Setiap dataset berisi jutaan paket jaringan dan berbagai serangan *cyber* di dalamnya. Gambar 3.2 menyajikan topologi jaringan yang digunakan untuk mengumpulkan data.



Gambar 3.2. Topologi Jaringan

Gambar 3.2 menunjukkan vektor serangan di mana serangan tersebut dilakukan. Penangkapan jaringan terjadi pada titik 1 dan titik X pada *router* sebagai tempat sistem deteksi jaringan.[14]

Setiap kumpulan data memiliki m baris (paket) dan 115 kolom (fitur) tanpa header. Fitur tersebut diekstraksi menggunakan *AfterImage feature extractor* yang memberikan gambaran statistik jaringan (host dan perilaku) dalam konteks paket

saat ini yang melintasi jaringan. *AfterImage feature extractor* digunakan karena dapat memproses jutaan aliran secara efisien, real-time, dan bertahap. Oleh karena itu, ekstraktor tersebut cocok untuk menangani lalu lintas jaringan.[14]

Dari 9 jenis serangan, penelitian ini hanya memfokus ke 1 jenis serangan yaitu Syn Dos atau DDos. Dataset yang diambil dari *Kitsune Attack Network dataset* sudah melalui tahap *preprocessing* sehingga hanya dilakukan pengecekan terhadap nilai yang hilang atau *missing values* pada dataset. Sebelum melatih model, *missing values* perlu ditangani karena algoritma yang diimplementasi tidak dapat menangani *missing values* secara langsung. Oleh karena itu baris atau kolom yang memiliki banyak *missing values* akan dihapus agar tidak mempengaruhi kinerja algoritma.

3.3 Visualisasi Data

Visualisasi data adalah langkah penting dalam *machine learning* untuk memahami pola, tren, dan distribusi data dengan lebih mudah. Dalam penelitian ini, proses visualisasi data menggunakan diagram batang untuk membandingkan jumlah lalu lintas jaringan yang normal dengan jumlah lalu lintas jaringan yang dianggap sebagai anomali (serangan DDoS). Diagram batang adalah salah satu jenis visualisasi yang umum digunakan untuk menampilkan data kategorikal.

3.4 *Splitting Data*

Tahap ini akan membagi data ke dalam dua set, yaitu *training data* dan *testing data*. Dataset akan dibagi menjadi dua bagian dengan rasio 8:2 dan 7:3. Di mana data yang akan digunakan sebagai *training data* mencakup 80% dan 70%. Sisanya yang sebesar 20% dan 30% akan digunakan sebagai *testing data*. Tahap ini berfungsi dalam mengevaluasi kinerja model dan memastikan bahwa model memiliki kemampuan generalisasi yang baik ketika diaplikasikan pada data baru.

a. *Training Data*
   Digunakan untuk melatih model dengan tujuan membantu model untuk mempelajari pola, hubungan, dan karakteristik dari dataset.

b. *Testing Data*
   Digunakan untuk menguji model yang telah dilatih untuk mengevaluasi kinerjanya. Data yang masuk dalam set ini tidak digunakan selama proses

pelatihan, sehingga memberikan gambaran baru mengenai kinerja model pada data yang belum pernah dilihat sebelumnya.

3.5 Implementasi Algoritma

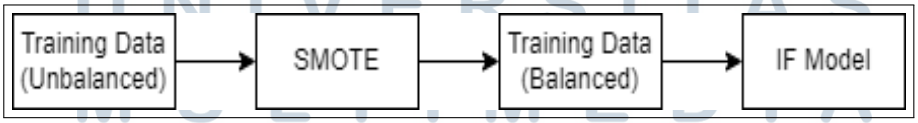
Pada bagian ini, kedua algoritma yang digunakan untuk deteksi anomali yaitu *Isolation Forest* dan *One-Class SVM* akan diimplementasikan. Proses implementasi meliputi berbagai cara sesuai dengan kebutuhan algoritma yang umum digunakan demi meningkatkan kinerja dan efektivitas model dengan kapasitas perangkat.

3.5.1 *Isolation Forest*

Implementasi algoritma *Isolation Forest* melibatkan beberapa langkah yaitu,

1. Membangun dan melatih model
   Model *Isolation Forest* dirancang menggunakan perangkat lunak *Visual Studio Code* dengan *extension Jupyter Notebook* menggunakan *library sklearn*. Model dibuat menggunakan bahasa pemrograman *Python*. Setelah model berhasil dibuat, akan dilakukan pelatihan menggunakan dataset yang telah dibagi.

2. *Balancing data*
   Dalam algoritma *Isolation Forest* dan *One-Class SVM*, proses *balancing data* tidak perlu dilakukan. Hal ini karena kedua algoritma merupakan algoritma *unsupervised machine learning* yang tidak memerlukan label kelas dalam proses pelatihan. Namun jika ada ketidakseimbangan yang signifikan dalam dataset, maka perlu dilakukan evaluasi khusus terhadap kinerja algoritma dalam mendeteksi anomali.



Gambar 3.3. Proses implementasi SMOTE

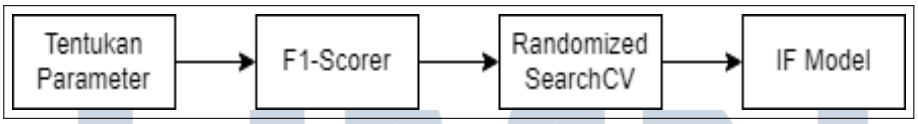
Gambar 3.3 menunjukkan proses *balancing data* menggunakan library *imbalance learn* yang dirancang khusus untuk mengatasi ketidakseimbangan kelas dalam tugas pembelajaran mesin. Metode yang digunakan

adalah *Synthetic Minority Over-sampling Technique* (SMOTE). Metode ini menghasilkan sampel sintetis baru untuk kelas minoritas dengan cara mengambil titik tengah dari pasangan sampel yang berdekatan dalam kelas minoritas. Ini akan membuat sampel baru yang bukan duplikat dari sampel dan mewakili variasi yang lebih besar dari kelas minoritas. Proses ini diimplementasikan setelah proses *splitting data*. Hal ini dikarenakan *balancing* data akan diterapkan hanya pada *training data*. Penerapan ini dilakukan agar tidak terjadi *data leakage* dimana informasi dari *testing data* masuk ke dalam tahap pelatihan model.

3. *Hyperparameter tuning*
   Pada tahap pelatihan, parameter model akan disesuaikan dengan cara *hyperparameter tuning* untuk mencapai tingkat kinerja terbaik. Parameter model *Isolation Forest* yang akan dilakukan *tuning* adalah,

   a. *n_estimators*, menunjukkan jumlah pohon dalam hutan yang akan dibuat.

   b. *max_samples*, jumlah sampel yang digunakan untuk melatih setiap pohon.



Gambar 3.4. Proses implementasi *RandomizedSearchCV*

Gambar 3.4 menunjukkan proses mencari parameter terbaik untuk algoritma *Isolation Forest* dilakukan dengan menggunakan metode *RandomizedSearchCV*. Metode ini bekerja dengan cara memilih kombinasi *hyperparameter* tertentu secara acak untuk dievaluasi. Ini memungkinkan untuk menemukan kombinasi *hyperparameter* yang optimal dengan waktu komputasi yang sedikit.

4. Penggabungan *data balancing* dan *hyperparameter tuning*
   Tahap ini menggabungkan 2 metode yang dapat meningkatkan kinerja model *Isolation Forest*. Proses ini akan melatih model menggunakan *training data* yang tersedia setelah balancing data dengan SMOTE serta menggunakan parameter yang telah dihasilkan saat proses *hyperparameter tuning*.

3.5.2 *One-Class SVM*

Implementasi algoritma *One-Class SVM* berfungsi sebagai pembanding kinerja algoritma *Isolation Forest* dalam melakukan deteksi anomali. Dengan keterbatasan perangkat keras dan ukuran *dataset* yang besar, maka implementasi algoritma *One-Class SVM* akan menggunakan 1% dari dataset. Model akan dibangun dan dilatih dengan cara sama yang dilalui oleh algoritma *Isolation Forest*. Perbedaannya hanya pada bagian *hyperparameter tuning* karena terdapat perbedaan parameter. Parameter yang berbeda adalah sebagai berikut.

a. *nu*, menentukan proporsi maksimal dari anomali dan batas bawah proporsi *support vectors*.

Dengan dataset yang digunakan, parameter yang akan di *tuning* hanya *'nu'*. *kernel* yang digunakan adalah *linear* karena mempertimbangkan fitur yang berjumlah 115, *kernel linear* akan lebih efektif dalam melatih model.

3.6 Pengujian dan Evaluasi Model

Bagian ini berisikan langkah-langkah yang dilakukan untuk mengevaluasi model yang telah dilatih menggunakan *testing data* yang telah disiapkan. Pengujian model adalah tahap kritis untuk memastikan bahwa model bekerja dengan baik pada data yang belum pernah dilihat sebelumnya. Metrik evaluasi seperti akurasi, *presisi*, *recall*, *F1-score*, dan *Confusion Matrix* digunakan untuk mengukur kinerja model.