BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam era digital yang semakin terhubung, serangan siber oleh aktor tidak sah terus mengalami peningkatan, sehingga menekankan pentingnya penerapan sistem keamanan siber yang tangguh (Tao et al., 2019). Salah satu tantangan kritis dalam domain ini adalah mendeteksi perilaku tidak sah, seperti pengambilalihan akun *(account takeover)*, di mana entitas jahat berhasil mengakses akun pengguna dengan tujuan merugikan. Perilaku ini sering kali meniru pola lalu lintas jaringan yang sah secara halus, menjadikannya sulit terdeteksi oleh sistem keamanan konvensional. Oleh karena itu, penerapan pembelajaran mesin sebagai alat untuk mendeteksi anomali telah muncul sebagai pendekatan yang menjanjikan dalam meningkatkan efektivitas pertahanan terhadap ancaman semacam ini (Demirdag et al., 2022).

Pengambilalihan akun dapat menyebabkan berbagai konsekuensi serius, seperti kerugian finansial, kebocoran data pribadi, dan penurunan reputasi organisasi. Perilaku tidak sah ini mencakup manipulasi kredensial pengguna, pemalsuan data perangkat, atau penyalahgunaan alamat IP yang sering kali terlihat mirip dengan aktivitas pengguna yang sah. Kondisi ini menekankan kebutuhan akan sistem deteksi yang tidak hanya akurat tetapi juga mampu mengidentifikasi pola perilaku anomali secara adaptif (Rodríguez et al., 2023).

Penelitian ini bertujuan untuk mengimplementasikan algoritma *Support Vector Machine* (SVM) sebagai model utama untuk mendeteksi anomali pada data login. Dataset yang digunakan mencakup berbagai atribut yang relevan, seperti identitas pengguna, alamat IP, lokasi geografis, penyedia jaringan, detail agen pengguna, jenis perangkat, waktu login, serta indikator perilaku mencurigakan, termasuk akun yang mungkin telah dikompromikan dan alamat IP berisiko tinggi (Wiefling et al., 2023). Untuk mengatasi kompleksitas perilaku tidak sah ini, penelitian menerapkan proses praproses data yang ketat dan seleksi fitur yang ditargetkan guna memastikan representasi data yang optimal.

Masalah ketidakseimbangan data, yang kerap menjadi tantangan dalam mendeteksi aktivitas tidak sah, diatasi dengan menggunakan metode SMOTE *(Synthetic Minority Over-sampling Technique)*. Teknik ini memperbaiki distribusi kelas minoritas, sehingga model lebih peka terhadap pola-pola anomali tanpa bias yang signifikan terhadap kelas mayoritas.

Model SVM dilatih menggunakan prosedur optimisasi yang mencakup normalisasi data dan pengkodean atribut untuk meningkatkan efisiensi pemrosesan. Untuk menentukan kombinasi hyperparameter yang optimal, penelitian ini menggunakan *RandomizedSearchCV*, sebuah metode yang secara efisien mengeksplorasi berbagai kombinasi parameter seperti C, gamma, dan kernel SVM. Pendekatan ini mempercepat proses pencarian tanpa mengorbankan performa model (Ibrahim et al., 2023). valuasi model dilakukan dengan memanfaatkan metrik seperti matriks kebingungan, akurasi, dan skor AUC-ROC, yang memberikan

gambaran yang komprehensif tentang keandalan dan efektivitas model dalam mendeteksi perilaku tidak sah (Hosseinzadeh et al., 2021).

Dengan pendekatan ini, penelitian diharapkan dapat memberikan kontribusi signifikan terhadap pengembangan sistem deteksi yang lebih adaptif dan responsif terhadap ancaman keamanan siber. Algoritma *Support Vector Machine* dipilih karena keunggulannya dalam menangani pola data yang kompleks dan non-linear, menjadikannya alat yang andal dalam mendeteksi perilaku anomali. Proses optimisasi *hyperparameter* menggunakan *RandomizedSearchCV* memungkinkan eksplorasi kombinasi parameter yang lebih luas secara efisien, sehingga meningkatkan akurasi deteksi anomali. Hasil penelitian ini diharapkan mampu memperkuat sistem keamanan siber secara keseluruhan dengan memberikan kemampuan deteksi yang lebih unggul, adaptif terhadap berbagai skenario ancaman, dan memberikan perlindungan yang lebih baik terhadap perilaku tidak sah dalam lingkungan digital.

1.2 Landasan Teori

1.2.1 Anomaly Detection

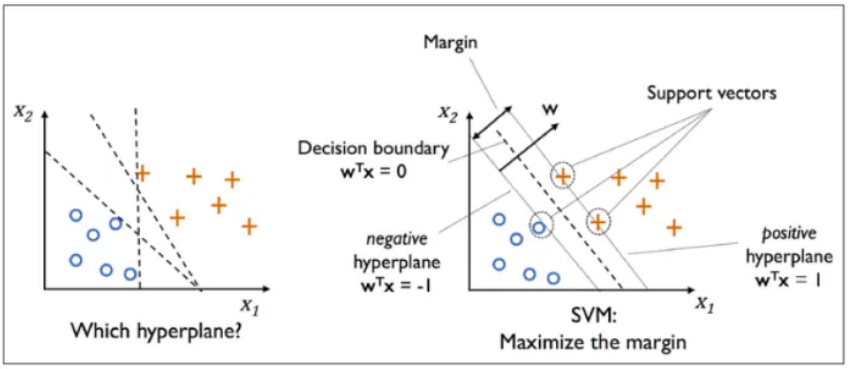
Anomaly Detection adalah kegiatan menganalisis data penting yang digunakan untuk mendeteksi data yang tidak normal pada lalu lintas jaringan yang kemudian dapat membantu manajemen dan masalah keamanan jaringan. Deteksi anomali, juga dikenal sebagai deteksi *outlier*, adalah proses mengidentifikasi data atau perilaku yang tidak biasa atau tidak sesuai dalam sebuah kumpulan data atau sistem. Tujuan utama dari deteksi anomali adalah untuk mengidentifikasi kejadian yang jarang atau langka yang mungkin mencurigakan atau memerlukan perhatian tambahan (Kamal and Setiawan, 2021). Deteksi Anomali juga dapat digunakan untuk mengidentifikasi aktivitas mencurigakan atau serangan siber di jaringan atau sistem komputer. Ini melibatkan pemantauan data lalu lintas jaringan, log aktivitas, atau perilaku pengguna untuk mencari tanda-tanda yang tidak biasa atau mencurigakan, seperti upaya masuk yang tidak sah, perubahan pola lalu lintas yang drastis, atau aktivitas mencurigakan lainnya (Evangelou and Adams, 2020).

1.2.2 Support Vector Machine

Support Vector Machine (SVM) merupakan salah satu metode dalam supervised learning yang biasanya digunakan untuk klasifikasi (seperti *Support Vector Classification*) dan regresi *(Support Vector Regression)*. Dalam pemodelan klasifikasi, SVM memiliki konsep yang lebih matang dan lebih jelas secara matematis dibandingkan dengan teknik-teknik klasifikasi lainnya. SVM juga dapat mengatasi masalah klasifikasi dan regresi dengan linear maupun non linear.

SVM digunakan untuk mencari *hyperplane* terbaik dengan memaksimalkan jarak antar kelas. *Hyperplane* adalah sebuah fungsi yang dapat digunakan untuk pemisah antar kelas. Dalam 2-D fungsi yang digunakan untuk klasifikasi antar kelas disebut sebagai *line whereas*, fungsi yang digunakan untuk klasifikasi antas kelas

dalam 3-D disebut *plane similarly*, sedangan fungsi yang digunakan untuk klasifikasi di dalam ruang kelas dimensi yang lebih tinggi di sebut *hyperplane*.



Gambar 1 Hyperplane yang memisahkan dua kelas positif (+1) dan

Hyperplane yang ditemukan SVM diilustrasikan seperti Gambar 1 posisinya berada ditengah-tengah antara dua kelas, artinya jarak antara hyperplane dengan objek-objek data berbeda dengan kelas yang berdekatan (terluar) yang diberi tanda bulat kosong dan positif. Dalam SVM objek data terluar yang paling dekat dengan *hyperplane* disebut *support vector*. Objek yang disebut *support vector* paling sulit diklasifikasikan dikarenakan posisi yang hampir tumpang tindih *(overlap)* dengan kelas lain. Mengingat sifatnya yang kritis, hanya *support vector* inilah yang diperhitungkan untuk menemukan *hyperplane* yang paling optimal oleh SVM.

Tabel 1 Pseudocode untuk SVM Linier

Langkah	deskripis
1	**Input:** - Dataset D = {(x1, y1), (x2, y2), ..., (xn, yn)} dengan xi sebagai fitur dan yi sebagai label kelas {-1, +1} - Parameter regulasi C (untuk soft margin)
2	**Inisialisasi bobot w dan bias b dengan nilai nol**
3	**Definisikan fungsi keputusan:** f(x) = sign(w · x + b)
4	**Optimasi Hyperplane dengan meminimalkan:** Minimize (1/2) ||w||^2 + C ∑ ξ_i Subject to: yi (w · xi + b) ≥ 1 - ξ_i ξ_i ≥ 0 untuk semua i

5	**Gunakan metode optimasi (misalnya, SMO - Sequential Minimal Optimization) untuk mencari solusi optimal dari w dan b.**
6	**Klasifikasi data baru:** For setiap data u: - Hitung nilai f(u) = sign(w · u + b) - Jika f(u) > 0 → Kelas +1 - Jika f(u) < 0 → Kelas -1.
7	**Output:** - Model SVM dengan w dan b optimal - Fungsi keputusan f(x) untuk prediksi

Tabel 2 Pseudocode untuk SVM Non-Linier

Langkah	deskripis
1	**Input:** - Dataset D = {(x1, y1), (x2, y2), ..., (xn, yn)} - Parameter regulasi C - Pilih fungsi kernel K(xi, xj)
2	**Inisialisasi bobot α (Lagrange multipliers)**
3	**Definisikan fungsi keputusan:** f(x) = sign(∑ α_i * y_i * K(x_i, x) + b)
4	**Optimasi dengan memaksimalkan:** Maximize: L(α) = ∑ α_i - 0.5 ∑∑ α_i * α_j * y_i * y_j * K(x_i, x_j) Subject to: 0 ≤ α_i ≤ C ∑ α_i * y_i = 0
5	**Gunakan algoritma SMO (Sequential Minimal Optimization) untuk mencari α optimal.**
6	**Hitung bobot w dan bias b berdasarkan nilai α.**
7	**Klasifikasi data baru:** For setiap data u: - Hitung f(u) = sign(∑ α_i * y_i * K(x_i, u) + b)

	- Jika f(u) > 0 → Kelas +1 - Jika f(u) < 0 → Kelas -1
8	**Output:** - Model SVM dengan parameter α optimal - Fungsi keputusan untuk klasifikasi

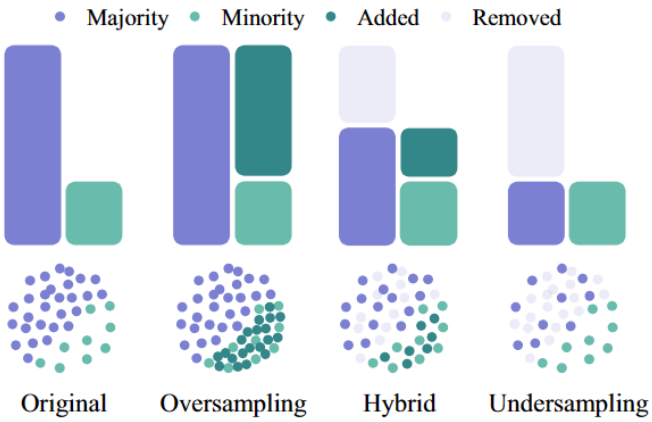
a. Linear SVM
Mencari hyperplane terbaik dengan memaksimalkan margin antara dua kelas menggunakan persoalan optimasi kuadratik.

b. Non-Linear SVM
Menggunakan kernel trick untuk mengubah data ke dimensi yang lebih tinggi sebelum menentukan hyperplane optimal.

c. Metode SMO
Digunakan untuk menyelesaikan persoalan optimasi dengan memperbarui parameter α iteratif.

1.2.3 Imbalanced Class

Imbalanced class merupakan suatu kondisi data yang tidak seimbang dengan jumlah data suatu kelas melebihi jumlah data pada kelas yang lain. Pada kelas data dengan jumlah banyak merupakan kelas mayoritas sedangkan kelas data yang sedikit merupakan kelas minoritas. Pendekatan pada level data untuk mengatasi masalah *Imbalanced class* adalah dengan menggunakan *Sampling-based approaches*. Dengan adanya penerapan sampling pada data *imbalanced*, tingkat *imbalanced class* semakin kecil dan klasifikasi dapat dilakukan dengan tepat (Schistad Solberg and Solberg, 1996). *Sampling-based approaches* merupakan modifikasi distribusi dari data training sehingga kedua kelas data (mayoritas dan minoritas) dipresentasikan dengan lebih baik didalam suatu data training. Sampling dibedakan menjadi dua yaitu *undersampling* dan *oversampling*. Metode *oversampling* dilakukan untuk menyeimbangkan jumlah distribusi data dengan cara meningkatkan jumlah data kelas minoritas sedangkan metode *undersampling* dilakukan dengan cara mengurangi jumlah data kelas mayoritas. Pada penelitian ini digunakan metode *Synthetic Minority Oversampling Technique* (SMOTE).



Gambar 2 Jenis sampling untuk preprocessing data yang tidak seimbang
(Sumber: Werner de Vargas et al., 2023)

Imbalanced bisa berkisar dari yang ringan sampai yang ekstrim, seperti yang ditunjukkan Tabel 3 di bawah ini.

Tabel 3 Imbalanced

Degree of imbalance	Proportion of Minority Class
Mild	20-40% of the data set
Moderate	1-20% of the data set
Extreme	<1% of the data set

1.2.4 Synthetic Minority Over-Sampling Technique(SMOTE)

Synthetic Minority Oversampling Technique (SMOTE) merupakan salah satu metode oversampling dengan teknik penambahan jumlah sampel pada kelas minoritas dengan melakukan replikasi data pada kelas minoritas secara acak sehingga menghasilkan jumlah data yang sama dengan data kelas mayoritas (Chawla et al., 2002). SMOTE digunakan untuk mengangani ketidakseimbangan kelas pada dataset klasifikasi, teknik ini mensintesis sampel baru dari kelas minoritas untuk menyeimbangkan kumpulan data dengan melakukan replikasi *(resampling)* kelas minoritas (Barus, 2022) Metode SMOTE menghasilkan data sintetik dengan menerapkan interpolasi linier antara titik kelas minoritas dan salah satu K tetangga terdekatnya. Penulis menjelaskan secara singkat algoritma SMOTE yang dikembang kan oleh (Chawla et al., 2002). Berikut adalah pseudocede dari Algoritma SMOTE.

Algorithm 1 A description of SMOTE over-sampling algorithm developed by Chawla et al (2002).

repeat

Select a minority sample x_0 at random.

Identify the *K*-nearest neighbors of x_i among the minority samples.

Randomly choose one of the *K* nearest neighbors of i, from the previous step and denote the chosen sample as x_j where k the rank of the chosen neighbor.

Perform linear interpolation between x_0 and the chosen neighbor x_j to create a new synthetic sample $x_{synthesis}$ according to

$x_{synthesis} = x_i + w(x_j - x_i)$ where w is a uniform random variable in the range [0,1.]

until *M* synthetic samples are generated.

Sampel sintesis baru dirumuskan pada persamaan (1)

$$x_{synthesis} = x_i + w(x_j - x_i) \quad (1)$$

Dimana:

$x_{synthesis}$ adalah sampel sitesis yang akan di hasilkan,

x_i sampel dari kelas minoritas yang sedang dipertimbangkan,

λ nilai acak antara 0 dan 1, yang menentukan seberapa jauh $x_{synthesis}$ antara x_i dan x_j. x_j salah satu dari k tetangga terdekat x_i.

Gambar 3 Mekanisme interpolasi SMOTE
(Sumber: Dholakiya, 2023)

Gambar 3 menunjukkan mekanisme pembangkitan SMOTE. Dari gambar tersebut dapat dilihat bahwa pola SMOTE terletak pada garis koneksi antara sampel kelas minoritas dan K tetangga terdekatnya. Posisi pola SMOTE yang masuk ke dalam membuat pola-pola tersebut lebih terkontraksi daripada distribusi aslinya seperti yang disimpulkan oleh Elreedy and Atiya (2019). Hal ini mendukung argumen kami bahwa pola SMOTE tidak selalu mengikuti kepadatan kelas minoritas asli. Dalam makalah ini, penulis secara analitis memperoleh fungsi kepadatan probabilitas dari pola yang dihasilkan SMOTE. Proses ini tidak hanya meningkatkan representasi

kelas minoritas tetapi juga mengurangi rasio ketidakseimbangan, sehingga menjadikan kumpulan data lebih kondusif untuk melatih model AI.

Metode yang digunakan pada algoritma SMOTE yaitu *k-nearest neighbors*. Penentuan jumlah replikasi yang dilakukan disesuaikan dengan jumlah anggota pada kelas mayor (Chawla et al., 2002b). Metode ini adalah tolak ukur standar untuk mengatasi masalah data yang tidak seimbang (Fernandez et al., 2018).

Ide utama di balik SMOTE adalah menjembatani kesenjangan antara kelas minoritas dan mayoritas dengan menghasilkan sampel sintetis. Berikut penjelasan langkah demi langkah cara kerja SMOTE:

a. **Identifikasi Sampel Minoritas**
   Langkah pertama melibatkan mengidentifikasi sampel yang termasuk dalam kelas minoritas dalam kumpulan data.
b. **Identifikasi K-Nearest Neighbors**
   Untuk setiap sampel minoritas, SMOTE mengidentifikasi K-nearest Neighbors-nya di ruang fitur. Biasanya, metrik jarak Euclidean digunakan untuk mengukur kesamaan antar titik data.
c. **Pembuatan Sampel Sintetis**
   Setelah tetangga teridentifikasi, SMOTE memilih tetangga acak dan menghitung perbedaan antara vektor fitur sampel minoritas dan tetangga yang dipilih. Perbedaan ini kemudian dikalikan dengan angka acak antara 0 dan 1 dan menambahkannya ke vektor fitur sampel minoritas. Proses ini menghasilkan terciptanya sampel sintetik baru yang terletak pada segmen garis antara sampel minoritas dan tetangga terpilihnya.
d. **Repeat**
   Proses menghasilkan sampel sintetik diulangi hingga tingkat keseimbangan kelas yang diinginkan tercapai.

1.2.5 RandomizedSearchCV

Penyesuaian hyperparameter merupakan langkah penting dalam meningkatkan performa model machine learning. Proses optimasi ini krusial karena hyperparameter harus ditetapkan sebelum pelatihan dan tidak dapat diubah selama proses pelatihan berlangsung (Benfenati et al., 2024). *RandomizedSearchCV* adalah salah satu metode optimasi hyperparameter yang menggunakan sampling acak untuk memilih subset kombinasi *hyperparameter* yang akan diuji, dengan tujuan meminimalkan atau memaksimalkan fungsi objektif f(x) tanpa perlu mengevaluasi seluruh kombinasi, sebagaimana dilakukan oleh *GridSearchCV* (Bergstra and Bengio, 2012).

Algoritma *RandomizedSearchCV* berfokus pada sampling acak sejumlah kombinasi hyperparameter dalam ruang pencarian, yang bertujuan untuk meningkatkan efisiensi komputasi. Setiap kombinasi akan dinilai berdasarkan metrik performa tertentu, seperti accuracy atau F1-score.

a. Argmin (argument of the minimum)

Argmin digunakan untuk menemukan kombinasi *hyperparameter x* yang menghasilkan nilai minimum dari fungsi objektif *f(x),* atau kesalahan minimum (Benfenati et al., 2024). *RandomizedSearchCV* secara efektif mencoba menemukan kombinasi yang meminimalkan kesalahan atau memaksimalkan performa

$$x^* \in \text{argmin}_{x \in X} f(x) \quad (2)$$

Di mana x* adalah nilai yang dipilih dari ruang pencarian *X* yang meminimalkan *f(x)*.

b. Argmax (argument of the maximum)

Argmax digunakan untuk mencari kombinasi hyperparameter yang memberikan nilai maksimum dari fungsi objektif *f(x).* Dalam beberapa kasus, optimasi dalam RandomizedSearchCV dapat diarahkan untuk memaksimalkan suatu metrik seperti precision atau recall (Bergstra and Bengio, 2012).

$$x^* \in \text{argmax}_{x \in X} f(x) \quad (3)$$

Berikut adalah tabel yang merangkum Pseudocode dari Algoritma *RandomizedSearchCV:*

Tabel 4 Penjelasan Detail *Pseudocode RandomizedSearchCV*

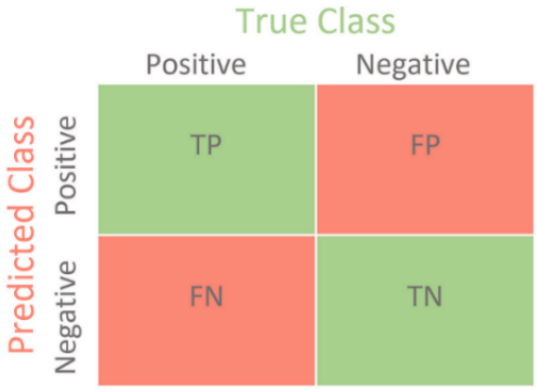
Langkah	deskripis
1	**Inisialisasi Parameter:** Tentukan estimator (model yang akan dioptimalkan), param_distributions (nilai hyperparameter yang akan disampel), n_iter (jumlah iterasi), dan cv (jumlah fold dalam cross-validation).
2	**Loop Utama dari 1 hingga n_iter:** Pada setiap iterasi, algoritma memilih satu kombinasi hyperparameter acak dari param_distributions untuk diuji.
3	**Sampling Kombinasi Hyperparameter:** Algoritma secara acak memilih kombinasi hyperparameter dari param_distributions, memungkinkan eksplorasi ruang pencarian secara efisien tanpa mengevaluasi seluruh kombinasi.
4	**Evaluasi Kombinasi pada Cross-Validation:**

	Untuk setiap kombinasi yang disampel, model dilatih menggunakan K-Fold Cross-Validation, dengan cv menentukan jumlah fold. Rata-rata nilai dari hasil cross-validation digunakan untuk menilai performa.
5	**Penyimpanan dan Pembaruan Kombinasi Terbaik:** Setelah setiap iterasi, algoritma menyimpan kombinasi yang memberikan nilai fungsi objektif terbaik (misalnya argmin atau argmax), dan memperbarui kombinasi terbaik sementara.
6	**Pengembalian Solusi Terbaik:** Setelah seluruh iterasi selesai, RandomizedSearchCV mengembalikan kombinasi hyperparameter yang menghasilkan performa terbaik berdasarkan metrik yang ditentukan.

Tabel 5 Penjelasan Parameter *RandomSearchCV*

Parameter	deskripsi
estimator	Model yang akan dioptimalkan, seperti RandomForestClassifier.
param_distributions	Dictionary berisi himpunan nilai hyperparameter yang akan diuji.
n_iter	Jumlah kombinasi yang akan diuji, biasanya default 10 atau lebih sesuai kebutuhan.
cv	Jumlah fold untuk cross-validation, seperti 5 atau 10.
scoring	Metrik evaluasi, seperti accuracy, precision, atau recall.
n_jobs	Jumlah proses yang digunakan; -1 untuk menggunakan semua inti prosesor yang tersedia.
random_state	Mengatur seed untuk reproducibility.

1.2.6 Confusion Matrix



Gambar 4 Confussion Matrix

Pada confussion matrix, hasil akurasi dikategorikan dengan nilai 0 untuk label negative dan 1 untuk label positive dengan penjelasan sebagai berikut:

a. True Positive (TP)
   Mengacu pada jumlah prediksi di mana pengklasifikasi dengan benar memprediksi **class 1** sebagai **class 1**.
b. True Negative (TN)
   Mengacu pada jumlah prediksi di mana pengklasifikasi dengan benar memprediksi kelas **class 0** sebagai **class 0.**
c. False Positive (FP)
   Mengacu pada jumlah prediksi di mana pengklasifikasi dengan benar memprediksi kelas **class 0** sebagai **class 1**.
d. False Negative (FN)
   Mengacu pada jumlah prediksi di mana pengklasifikasi dengan benar memprediksi kelas **class 1** sebagai **class 0.**

Keterangan:
Class 0 = tidak anomali.
Class 1 =anomali.

Metrik evaluasi yang digunakan untuk tiap model adalah Accuracy, Precision, dan Recall. Dihitung dengan menggunakan nilai confussion matrix yang mengandung nilai true positive (TP), true negative (TN), false positive (FP), dan false negative (FN).

a. *Accuracy*

Accuracy menggambarkan seberapa akurat model dapat mengklasifikasikan dengan benar. Maka, *accuracy* merupakan rasio prediksi benar (positif dan negatif) dengan keseluruhan data. Dengan kata lain, *accuracy* merupakan tingkat kedekatan nilai prediksi dengan nilai aktual (sebenarnya). *Akurasi* menghitung jumlah *true positive*

dan *true negative* dibagi keseluruhan data. Nilai *accuracy* dapat diperoleh dengan persamaan (1)

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (4)$$

b. *Precision (Positive predictive value)*

Precision menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Maka, *precision* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positf. Dari semua kelas positif yang telah di prediksi dengan benar, berapa banyak data yang benar-benar positif. *Precision* menghitung *true positive* dibagi jumlah keseluruhan data yang diprediksi sebagai positif. Nilai *precision* dapat diperoleh dengan persamaan (5).

$$Precision = \frac{(TP)}{(TP + FP)} \quad (5)$$

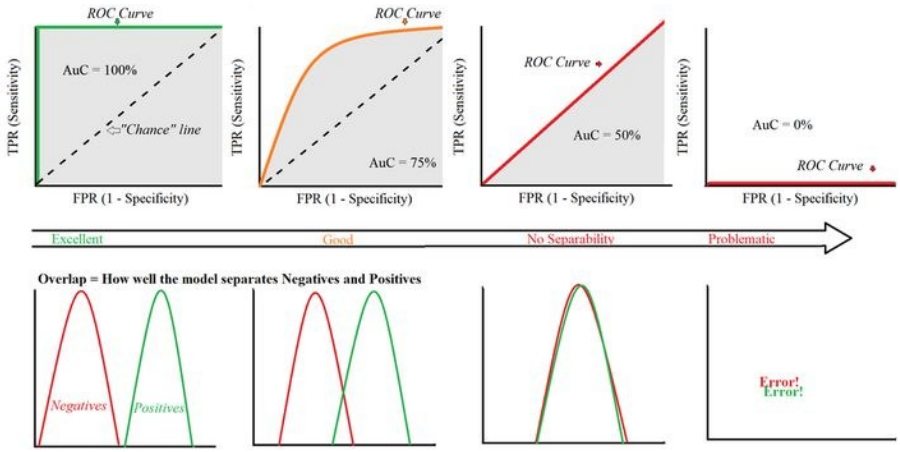
c. *Recall* atau *sensitivity (True Positive Rate)*

Recall menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Maka, *recall* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. Sedangkan *Recall* menghitung true positive dibagi keseluruhan data yang benar positif Nilai *recall* dapat diperoleh dengan persamaan (3).

$$Recall = \frac{(TP)}{(TP + FN)} \quad (6)$$

1.2.7 AUC ROC Curve

Metrik kinerja yang digunakan dalam pembelajaran mesin (ML), khususnya dalam masalah klasifikasi (Bradley, 1997). Kurva *Receiver Operating Characteristic* (ROC) adalah representasi grafis dari kinerja model klasifikasi pada berbagai pengaturan ambang batas.



Gambar 5 AUC ROC Curve

AuC-ROC mengukur seluruh area dua dimensi di bawah seluruh kurva ROC dari (0,0) hingga (1,1). Skor ini memberikan ukuran kinerja agregat di semua ambang batas klasifikasi yang memungkinkan. Model dengan skor AuC yang lebih tinggi umumnya menunjukkan kinerja yang lebih baik, karena hal ini menunjukkan bahwa model tersebut dapat secara efektif membedakan kelas positif dan negatif. AUC-ROC adalah metrik evaluasi yang populer untuk pengklasifikasi biner. Metrik ini dipilih karena tidak bergantung pada ambang batas tertentu dan lebih dekat dengan metrik linear, membuatnya lebih mudah untuk dianalisis (Tafvizi et al., 2022). Yang terpenting adalah berapa luas area di bawah kurva (Area di bawah Kurva = AuC). Kurva ideal pada gambar kiri terisi 100%, yang berarti Anda akan dapat membedakan antara hasil negatif dan hasil positif 100% setiap saat (yang hampir tidak mungkin dilakukan dalam kehidupan nyata). Semakin jauh Anda ke kanan, semakin buruk pendeteksiannya. Kurva ROC di paling kanan memberikan dampak yang lebih buruk dan tidak lebih dari hanya sebuah kebetulan, karena mencampuradukkan sisi negatif dan positif (yang berarti kemungkinan besar mengalami kesalahan dalam penyiapan). Berikut adalah interpretasi umum dari nilai AUC:

- **AUC = 1**: Model memiliki kinerja sempurna dalam membedakan antara kelas positif dan negatif.
- **0.5 < AUC < 1**: Model memiliki kinerja yang lebih baik daripada tebak-tebakan acak.
- **AUC = 0.5**: Model memiliki kinerja yang setara dengan tebak-tebakan acak.
- **AUC < 0.5**: Model memiliki kinerja yang lebih buruk daripada tebak-tebakan acak (model mungkin melakukan klasifikasi terbalik

1.3 Perumusan Masalah

Berdasarkan latar belakang masalah di atas, maka rumusan masalah pada penelitian ini adalah bagaimana *RandomizedSearchCV* dapat mengoptimalkan hyperparameter model support vector machine dengan menentukan nilai optimal untuk berbagai parameter guna meningkatkan performa model dengan waktu komputasi yang rendah?

1.4 Tujuan dan Manfaat

1.4.1 Tujuan

Tujuan penelitian yang ingin dicapai pada penelitian ini yaitu mengoptimalkan proses hyperparameter tuning menggunakan *RandomizedSearchCV* untuk mengurangi waktu komputasi tanpa mengorbankan kualitas hasil optimasi.

1.4.2 Manfaat

Manfaat dari penelitian ini adalah sebagai berikut:

1. Berkontribusi dalam pengembangan serta penerapan algoritma untuk optimasi *hyperparameter*, dengan penekanan khusus pada metode *RandomSearchCV*.
2. Menyediakan referensi yang berguna bagi peneliti dan praktisi dalam bidang *machine learning* dan data *science* yang ingin mengembangkan atau menerapkan algoritma klasifikasi untuk permasalahan deteksi anomali pada data login.
3. Memberikan wawasan tentang bagaimana memilih dan mengembangkan algoritma yang tepat untuk menyelesaikan masalah klasifikasi dengan efisiensi waktu komputasi dan akurasi yang tinggi.
4. Menyumbang wawasan berharga bagi komunitas akademik dan praktisi, memberikan pandangan yang komprehensif tentang pengembangan dan penerapan algoritma optimasi *hyperparameter* dalam konteks klasifikasi anomali pada data login.

BAB II
METODE PENELITIAN

2.1 Tempat dan Waktu

Penelitian ini dimulai pada bulan Januari 2024 dan berlangsung hingga Agustus 2024. Adapun mengenai lokasi, penelitian ini dilakukan di Universitas Hasanuddin, Departemen Teknik Informatika.

2.2 Benda Uji dan Alat

Benda dan alat yang digunakan dalam penelitian ini meliputi perangkat lunak dan perangkat keras berikut:

1.1 Perangkat lunak

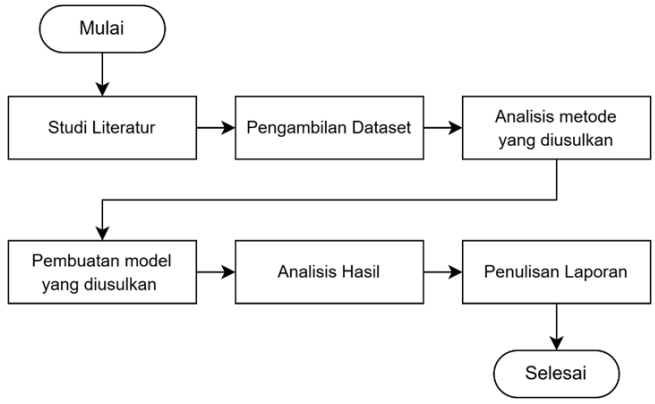
a. *Operating Sistem Operasi Windows 11 Pro, 64 bit*
b. *Conda Environment*
c. *Jupyter Notebook*
d. *Python*

2.1 Perangkat keras

a. Perangkat: Huawei
b. CPU: 11th Gen Intel(R) Core(TM) i5-1155G7 @ 2.50GHz (8 CPUs), 5GHz
c. GPU: Intel(R) Iris(R) Xe Grapichs
d. Ram 16 GB LPDDRX

2.3 Tahapan penelitian

Tahapan penelitian merupakan langkah-langkah yang dilakukan selama proses penelitian agar penelitian dapat berlangsung secara sistematis. Adapun tahapan pada penelitian ini dapat dilihat pada gambar dibawah ini:



Gambar 6 Tahapan Penelitian

2.4 Teknik Pengambilan Data

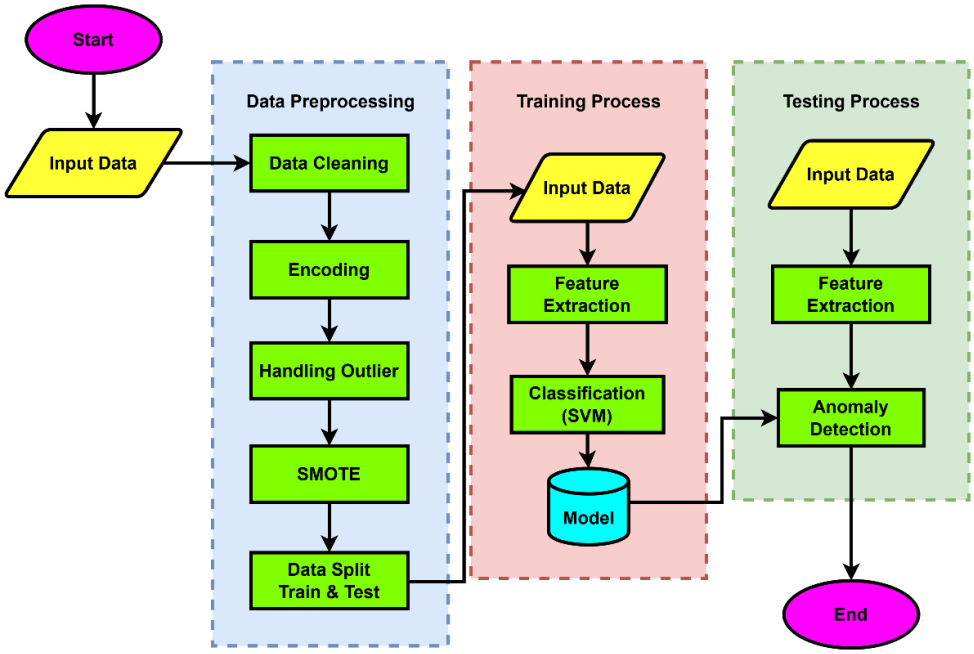
Data yang digunakan dalam penelitian ini adalah *Risk-Based Authentication* (RBA) yang diambil langsung dari Kaggle. Data ini disintesis dari perilaku login di dunia nyata dari lebih dari 3,3 juta pengguna di layanan satu kali login (SSO) berbasis skala besar di Norwegia. Pengguna menggunakan SSO ini untuk mengakses data sensitif yang disediakan oleh layanan online, seperti penyimpanan awan dan informasi tagihan. Kami menggunakan kumpulan data ini untuk mempelajari bagaimana model RBA (Freeman et al., 2016). berperilaku pada layanan online skala besar di dunia nyata.

Tabel 6 Atribut pada dataset

Attributes	**Data Type**	**Description**
IP Addres	String	IP address belonging to the login attempt
Country	String	Country derived from the IP address
Region	String	Region derived from the IP address
City	String	City derived from the IP addres
ASN	Integer	Autonomous system number derived from the IP address
User Agent String	String	User agent string submitted by the client
OS Name and Version	String	Operating system name and version derived from the user agent string
Browser Name and Version	String	Browser name and version derived from the user agent string
Device Type	String	Device type derived from the user agent string
User ID	Integer	Idenfication number related to the affected user account
Login Timestamp	Integer	Timestamp related to the login attempt
Round-Trip Time (RTT) [ms]	Integer	Server-side measured latency between client and server
Login Succesfull	Boolean	True: Login was successful, False: Login failed
Is Attack IP	Boolean	IP address was found in known attacker data set
Is Account Takeover	Boolean	Login attempt was identified as account takeover by incident response team of the online service

2.5 Perancangan dan Implementasi Sistem

Perancangan sistem adalah merancang atau mendesain suatu sistem yang baik dimana isinya adalah langkah-langkah operasi dalam proses pengolahan data dan proses prosedur-prosedur untuk mendukung operasi sistem. Tujuan dari perancangan sistem adalah untuk memenuhi kebutuhan para pengguna sistem serta memberikan gambaran yang jelas dan rancang bangun yang lengkap kepada perogrammer dan ahli-ahli yang terlibat didalam. Alur perancangan sistem pada penelitian ini pada Gambar 7 berikut.



Gambar 7 Rancangan Sistem

Berdasarkan Gambar 7 perancangan sistem terdiri dari beberapa proses, yang diuraikan sebagai berikut:

2.5.1 Data Input

Pada langkah ini, data yang akan digunakan untuk deteksi anomali di input ke dalam sistem.

2.5.2 Data Cleaning

Pada tahap ini, proses pembersihan data sangat penting untuk memastikan data yang digunakan dalam analisis atau pelatihan model memiliki kualitas tinggi dan bebas dari kesalahan. Langkah-langkah dalam pembersihan data meliputi penghapusan nilai yang hilang, duplikasi, serta data yang tidak relevan atau tidak konsisten. Nilai yang hilang dapat menyebabkan masalah dalam analisis dan model prediktif, sehingga harus dihapus atau diimputasi, dengan baris atau kolom yang

memiliki banyak nilai hilang dihapus. Duplikasi data juga diperiksa dan dihapus untuk memastikan setiap entri unik. Data yang tidak relevan atau tidak konsisten, seperti kolom `Round-Trip Time [ms],` `Region,` `City,` `Login Timestamp,` dan `index,` dihilangkan untuk menyederhanakan dataset dan memastikan hanya data yang relevan yang digunakan

2.5.3 Encoding

Proses ini mencakup beberapa langkah penting dalam mempersiapkan data agar dapat diolah dengan model analisis. Pertama, data mentah pada kolom waktu (timestamp) dikonversi menjadi format jam. Hal ini dilakukan untuk mengidentifikasi pola berdasarkan waktu aktivitas, seperti apakah tindakan pengguna lebih sering terjadi pada jam-jam tertentu dalam sehari. Selanjutnya, kolom boolean, yang hanya memiliki dua nilai (misalnya, true atau false), diubah menjadi nilai numerik (biasanya 0 dan 1) untuk memudahkan pemrosesan oleh algoritma. Kemudian, data kategorikal, seperti jenis user agent (contohnya: Chrome, Safari, Firefox) dan sistem operasi (contohnya: Windows, macOS, Linux), difaktorisasi. Ini berarti setiap kategori diubah menjadi kode numerik unik yang merepresentasikan setiap kategori, memungkinkan algoritma statistik atau pembelajaran mesin untuk menganalisisnya. Proses faktorisasi ini sangat penting dalam memastikan data kategorikal dapat digunakan sebagai input dalam model tanpa menghilangkan informasi yang melekat pada jenis kategori tersebut. Dengan cara ini, data telah disusun sedemikian rupa sehingga lebih informatif dan siap untuk analisis lebih lanjut dalam model pembelajaran mesin atau statistik.

2.5.4 SMOTE

Setelah menangani outlier, dilakukan pemeriksaan keseimbangan data. Perbedaan antara akun takeover dan non-takeover melebihi 95%. Oleh karena itu, diperlukan pengambilan sampel untuk mengatasi ketidakseimbangan data ini. Jika masalah ini tidak ditangani, model klasifikasi kemungkinan akan memprioritaskan kelas mayoritas, sehingga mengurangi kontribusi kelas minoritas dan menghasilkan sensitivitas yang lebih rendah. Maka dari itu, penting untuk menangani ketidakseimbangan ini menggunakan metode SMOTE. Teknik ini menerapkan pendekatan oversampling pada kelas minoritas dengan menciptakan data "sintetis" khusus untuk kelas minoritas. SMOTE bekerja berdasarkan tetangga terdekat (nearest neighbours), menggunakan jarak Euclidean antara titik data dalam ruang fitur atau variabel. Pembuatan sampel sintetis diformulasikan dalam Persamaan.

$$x_{synthesis} = x_i + \lambda \times (x_j - x_i) \quad (3)$$

xsynthesis adalah sampel sintetis yang akan dihasilkan, xi adalah sampel dari kelas minoritas yang sedang dipertimbangkan, λ adalah angka acak antara 0 dan 1 yang menentukan seberapa jauh posisi *xsynthesis* antara *xi* dan *xj.xj* adalah salah satu dari k tetangga terdekat dari *xi*.

2.5.5 Data Spliting Train and Test

Langkah selanjutnya dalam proses ini adalah mempartisi dataset menjadi dua bagian yang berbeda. Bagian pertama, yang terdiri dari data pelatihan, digunakan untuk membangun dan mengoptimalkan model klasifikasi. Sementara itu, bagian kedua, yang terdiri dari data pengujian, berfungsi untuk menilai kinerja dan kemampuan generalisasi model. Dalam penelitian ini, dataset dibagi dengan rasio 80:20, di mana 80% data dialokasikan untuk melatih model, sedangkan 20% sisanya disisihkan untuk pengujian dan validasi.

2.5.6 Fature Extraction

Ekstraksi fitur dalam dataset ini bertujuan untuk mengidentifikasi atribut-atribut penting dalam mendeteksi anomali dan potensi ancaman keamanan pada data login. Atribut yang dipilih meliputi Alamat IP, Negara, dan ASN *(Autonomous System Number)* untuk memantau lokasi login serta pola jaringan yang mencurigakan. Informasi terkait *User Agent String*, Nama OS, Versi Peramban, dan Jenis Perangkat diekstraksi untuk mendeteksi perubahan yang tidak biasa pada perangkat atau perangkat lunak. Jam Login digunakan untuk menganalisis perilaku login, sedangkan atribut Login Berhasil, Apakah IP Serangan, dan Apakah Pengambilalihan Akun digunakan untuk mendeteksi upaya login yang mencurigakan atau indikasi pengambilalihan akun. Ekstraksi ini sangat penting untuk meningkatkan proses klasifikasi dan efektivitas deteksi ancaman.

2.5.7 SVM Clasification

Klasifikasi ini bergantung pada algoritma SVM, yang dilatih untuk mengenali pola login yang khas dan mengidentifikasi penyimpangan menggunakan fitur yang telah diekstraksi. Batas ideal dioptimalkan untuk memaksimalkan margin, atau jarak antara batas tersebut dan titik data terdekat dari setiap kelas (dikenal sebagai *support vectors*). Teknik ini bertujuan untuk menemukan batas yang memberikan pemisahan terbesar antara dua kelas data. Batas ini direpresentasikan sebagai garis dalam ruang dua dimensi atau bidang/hiperbidang dalam dimensi yang lebih tinggi. Representasi matematis dari batas ini dinyatakan sebagai:

$$w.x + b = 0 \tag{3}$$

Dimana :

w adalah bobot

x adalah vector

b adalah bias

2.5.8 Model SVM

Model yang telah dilatih disimpan dan digunakan untuk menganalisis data login baru. Setiap login baru akan melalui proses yang sama (ekstraksi fitur dan klasifikasi) untuk menentukan apakah login tersebut tergolong normal atau anomali.

2.5.9 Anomaly Detection

Hasil pengujian dibandingkan dengan model yang telah dilatih dengan mengevaluasi setiap titik data uji untuk mengklasifikasikannya sebagai anomali atau non-anomali. Prediksi model kemudian dibandingkan dengan hasil uji aktual untuk menilai tingkat akurasi. Tingkat kesesuaian yang tinggi antara prediksi dan hasil aktual menunjukkan efektivitas model dalam mendeteksi dan mengenali pola anomali dengan akurat.