BAB 2
LANDASAN TEORI

2.1 Penelitian Terdahulu

Penelitian terdahulu berisi pemahaman mengenai perkembangan metode, pendekatan, serta hasil penelitian yang telah dilakukan sebelumnya pada topik yang relevan. Kajian ini bertujuan untuk mengidentifikasi kelebihan, keterbatasan, maupun celah penelitian yang masih dapat dikembangkan lebih lanjut. Pada penelitian ini, kajian difokuskan pada topik *anomaly detection*, *process mining*, serta penerapan metode *machine learning* pada data *event log*. Selain itu, dilakukan pula tinjauan terhadap penelitian yang menerapkan algoritma DBSCAN, *Support Vector Machine* (SVM), serta pendekatan berbasis *pseudo-labeling* dan kombinasi metode *clustering* dengan klasifikasi dalam proses deteksi anomali. Hasil kajian ini digunakan sebagai landasan dalam menentukan pendekatan, metode, serta pengembangan model yang digunakan pada penelitian ini, sebagaimana dirangkum pada Tabel 2.1.

Tabel 2.1. Ringkasan telaah literatur

Sumber	Judul Penelitian	Metode	Pembahasan
[4]	*Evaluating the Performance of SVM, Isolation Forest, and DBSCAN for Anomaly Detection*	SVM, IF, DBSCAN	Menunjukkan perbandingan performa metode anomaly detection, di mana DBSCAN efektif dalam mendeteksi outlier dan SVM baik untuk klasifikasi, SVM efektif untuk point anomaly, DBSCAN sensitif terhadap outlier.
[5]	*Data-Driven Loan Default Prediction*	Machine learning	Menunjukkan metode preprocessing dan karakteristik data proses bisnis dalam meningkatkan performa model prediksi.

Dilanjutkan ke halaman berikutnya

Tabel 2.1. Ringkasan telaah literatur (lanjutan)

Sumber	Judul Penelitian	Metode	Pembahasan
[8]	*Semi-Supervised Detection using Graph Autoencoders*	Graph-based semi-supervised	Deteksi anomali berbasis graf pada event log dengan pendekatan semi-supervised, namun kompleks secara komputasi.
[9]	*Anomaly Detection using Clustering (K-Means with DBSCAN) and SMO*	K-Means, DBSCAN, SMO	Metode hybrid untuk deteksi anomali dengan kombinasi clustering dan supervised learning.
[10]	*Applied Machine Learning to Anomaly Detection in Enterprise Purchase Processes*	Clustering, Isolation Forest	Menunjukkan pentingnya pendekatan hybrid pada data proses nyata dan potensi pengembangan pipeline berbasis *clustering*.
[14]	*Semi-Supervised Anomaly Detection in Business Process Event Data*	Semi-supervised DL	Pendekatan semi-supervised pada event log dengan performa tinggi, namun menggunakan model kompleks berbasis deep learning.
[15]	*SemiSMAC: A semi-supervised framework for log anomaly detection*	Semi-supervised DL	Framework semi-supervised dengan otomatisasi, menunjukkan pentingnya pipeline adaptif pada data log.
[11]	*Unified Semi-Supervised Pipeline for ASR*	Pseudo-labeling	Menunjukkan efektivitas pseudo-label dalam meningkatkan performa model pada data tanpa label.

Dilanjutkan ke halaman berikutnya

Tabel 2.1. Ringkasan telaah literatur (lanjutan)

Sumber	Judul Penelitian	Metode	Pembahasan
[16]	*Semi-supervised anomaly detection algorithms: A comparative summary and future research directions*	Summary anomaly detection	Mengevaluasi berbagai pendekatan semi-supervised dalam deteksi anomali dan arah penelitian selanjutnya.

Secara keseluruhan, penelitian dalam bidang deteksi anomali menunjukkan perkembangan yang signifikan, terutama pada data proses bisnis yang direpresentasikan dalam bentuk *event log*. Dalam konteks domain perbankan, penelitian seperti [5] menunjukkan bahwa data proses pinjaman memiliki kompleksitas tinggi karena melibatkan urutan aktivitas, atribut temporal, variasi alur proses, serta keterlibatan banyak sumber daya. Karakteristik tersebut menjadikan *event log* sebagai sumber data yang relevan untuk mendeteksi penyimpangan perilaku proses bisnis, seperti keterlambatan proses, pengulangan aktivitas, maupun alur proses yang tidak efisien.

Pendekatan awal dalam deteksi anomali pada data tanpa label umumnya menggunakan metode *unsupervised learning*, salah satunya berbasis *clustering*. Metode ini tidak memerlukan label aktual dan mampu mengidentifikasi pola kepadatan data serta *outlier*. Studi seperti [4] menunjukkan bahwa DBSCAN efektif dalam mendeteksi anomali berbasis kepadatan karena mampu mengenali data yang tidak termasuk ke dalam *cluster* sebagai *noise*. Karakteristik tersebut relevan dengan penelitian ini karena data *event log* yang digunakan tidak memiliki label anomali aktual.

Seiring perkembangannya, beberapa penelitian mulai mengombinasikan hasil *clustering* dengan model klasifikasi melalui mekanisme *pseudo-labeling*. Pendekatan ini banyak dijumpai pada penelitian bertema *semi-supervised learning* maupun model *hybrid*, di mana hasil dari metode tanpa label dimanfaatkan sebagai label sementara (*pseudo-label*) untuk melatih model klasifikasi [11]. Konsep tersebut menunjukkan bahwa *pseudo-label* dapat menjadi alternatif ketika label aktual tidak tersedia, sehingga model tetap dapat mempelajari pola pemisahan antara data normal dan anomali.

Selain itu, beberapa penelitian juga mengembangkan pendekatan yang lebih kompleks, seperti metode berbasis *self-attention* [14] maupun metode berbasis graf [8]. Meskipun mampu menangkap hubungan antaraktivitas secara lebih mendalam, pendekatan tersebut umumnya memerlukan kompleksitas model dan sumber daya komputasi yang lebih tinggi.

Berdasarkan hasil kajian tersebut, penelitian ini mengadopsi konsep *pseudo-labeling* yang banyak digunakan pada pendekatan *hybrid*. DBSCAN digunakan pada tahap awal untuk mengidentifikasi *noise* sebagai kandidat anomali dan menghasilkan *pseudo-label*, sedangkan SVM digunakan untuk mempelajari pola pemisahan antara data normal dan anomali berdasarkan *pseudo-label* tersebut. Berbeda dengan pendekatan *semi-supervised learning* klasik yang memanfaatkan kombinasi label aktual dan data tidak berlabel, penelitian ini tidak menggunakan label anomali dari pakar sebagai data pelatihan. Seluruh label diperoleh dari hasil proses DBSCAN sehingga pendekatan yang digunakan lebih tepat dipandang sebagai kombinasi algoritma berbasis *pseudo-labeling*.

Dalam konteks proses pengajuan kredit perbankan, pendekatan tersebut dipilih karena mampu diterapkan pada data *event log* yang tidak memiliki label anomali aktual. Hasil deteksi tidak dimaksudkan sebagai keputusan akhir, melainkan sebagai sistem deteksi awal (*early detection system*) untuk membantu memprioritaskan kasus yang perlu dianalisis lebih lanjut. Dengan demikian, kandidat anomali yang dihasilkan dapat digunakan sebagai dasar untuk mengidentifikasi potensi permasalahan proses bisnis, seperti durasi proses yang panjang, aktivitas berulang, keterlibatan sumber daya yang tinggi, maupun alur proses yang tidak efisien.

2.2 Event Log dan Proses Bisnis

Proses bisnis merupakan sekumpulan aktivitas yang saling terstruktur dan terkoordinasi untuk mencapai suatu tujuan tertentu dalam organisasi [17]. Proses bisnis umumnya terdiri dari serangkaian tahapan yang melibatkan berbagai entitas, aturan, serta alur keputusan yang saling berkaitan. Dalam praktiknya, proses bisnis dapat memiliki variasi jalur (*process variants*) tergantung pada kondisi, kebijakan, maupun karakteristik kasus yang ditangani [17].

Untuk menganalisis dan memahami perilaku proses bisnis, digunakan data yang dikenal sebagai *event log*. *Event log* merupakan kumpulan catatan kejadian yang merekam aktivitas-aktivitas yang terjadi dalam suatu proses bisnis.

Setiap *event* biasanya memiliki atribut utama, seperti *case identifier* (ID kasus), *activity* (jenis aktivitas), dan *timestamp* (waktu kejadian), serta dapat dilengkapi dengan atribut tambahan seperti sumber daya (*resource*) atau status proses [15]. Sekumpulan *event* yang memiliki *case identifier* yang sama disebut sebagai *trace*, yang merepresentasikan satu instansi proses dari awal hingga akhir.

Struktur *event log* memungkinkan analisis terhadap urutan aktivitas, durasi proses, serta pola interaksi antar tahapan dalam suatu proses bisnis. Karakteristik ini menjadikan *event log* sangat relevan untuk digunakan dalam deteksi anomali, karena penyimpangan perilaku proses dapat diidentifikasi melalui perbedaan pola urutan aktivitas, frekuensi kejadian, maupun aspek temporal. Namun, data *event log* cenderung kompleks karena bersifat sekuensial, memiliki variasi jalur yang tinggi, serta sering kali mengandung ketidakseimbangan antara pola normal dan penyimpangan [3]. Selain itu, data proses bisnis juga dapat mengandung hubungan yang kompleks antar aktivitas, sehingga memerlukan pendekatan analisis yang mampu menangkap keterkaitan struktural maupun temporal dalam data [8].

Dalam konteks deteksi anomali, *event log* berperan sebagai sumber data utama untuk mengidentifikasi penyimpangan dalam proses bisnis. Anomali dapat muncul dalam bentuk urutan aktivitas yang tidak sesuai, aktivitas yang hilang atau berulang secara tidak wajar, maupun durasi proses yang menyimpang dari pola normal [14]. Oleh karena itu, pemahaman terhadap struktur dan karakteristik *event log* menjadi penting sebagai dasar dalam penerapan metode deteksi anomali berbasis *machine learning*, khususnya pada analisis proses bisnis yang kompleks dan dinamis [1]. Selain berfungsi untuk mengidentifikasi penyimpangan, hasil deteksi anomali pada *event log* juga dapat dimanfaatkan sebagai dasar evaluasi dan perbaikan proses bisnis, seperti mengidentifikasi ketidakefisienan alur, potensi kesalahan operasional, maupun pola proses yang tidak optimal, sehingga mendukung upaya peningkatan kinerja dan kualitas proses secara berkelanjutan.

2.3 Deteksi Anomali dalam Proses Bisnis

Dalam konteks analisis data proses bisnis yang direpresentasikan dalam bentuk *event log*, deteksi anomali merupakan proses identifikasi observasi atau kejadian yang menyimpang secara signifikan dari pola yang dianggap normal atau diharapkan [1]. Anomali dalam proses bisnis tidak hanya muncul dalam bentuk nilai ekstrem, tetapi juga dapat berupa penyimpangan pada struktur alur aktivitas, seperti urutan aktivitas yang tidak lazim, durasi proses yang tidak wajar,

pengulangan aktivitas secara berlebihan, maupun kombinasi aktivitas yang jarang terjadi [3]. Dalam literatur, konsep ini juga dikenal sebagai *outlier detection* atau *novelty detection*, yang bertujuan untuk mengidentifikasi pola yang berbeda dari mayoritas data [18].

Berbagai pendekatan telah dikembangkan untuk mendeteksi anomali, mulai dari metode statistik hingga pendekatan berbasis *machine learning* dan *deep learning*. Metode statistik umumnya mendeteksi anomali berdasarkan deviasi terhadap distribusi data tertentu, namun memiliki keterbatasan dalam menangani data proses bisnis yang kompleks dan tidak selalu mengikuti distribusi yang jelas [13]. Pendekatan berbasis *machine learning*, seperti *clustering* dan *one-class classification*, memungkinkan pembelajaran pola normal tanpa asumsi distribusi yang ketat [18]. Sementara itu, pendekatan *deep learning* mampu menangkap pola non-linear dan ketergantungan temporal yang kompleks, namun membutuhkan sumber daya komputasi yang lebih besar [8]. Oleh karena itu, pemilihan metode deteksi anomali sangat bergantung pada karakteristik data serta ketersediaan label dalam proses bisnis yang dianalisis [15].

2.4 Deteksi Anomali dalam Proses Transaksi Bisnis Perbankan

Deteksi anomali pada proses transaksi bisnis perbankan merupakan penerapan khusus dari deteksi anomali pada proses bisnis yang berfokus pada identifikasi penyimpangan yang berpotensi menimbulkan risiko operasional, ketidaksesuaian prosedur, maupun indikasi kecurangan. Dalam sektor perbankan, proses bisnis memiliki tingkat kompleksitas yang tinggi serta melibatkan berbagai tahapan yang saling terintegrasi, sehingga penyimpangan dalam alur proses dapat berdampak signifikan terhadap kualitas pengambilan keputusan dan pengelolaan risiko.

Salah satu proses yang banyak dianalisis dalam konteks ini adalah *loan application process*, yang mencakup tahapan pengajuan permohonan, verifikasi data, analisis kredit, persetujuan, hingga pencairan dana. Proses ini memiliki alur yang terstruktur namun tetap memungkinkan variasi jalur (*process variants*), sehingga menghasilkan pola proses yang beragam dan dinamis.

Dalam implementasinya, analisis dilakukan dengan memanfaatkan data *event log* yang merekam urutan aktivitas dan informasi temporal [3]. Berdasarkan karakteristik tersebut, penyimpangan dalam proses dapat diamati melalui ketidaksesuaian alur, variasi durasi, maupun pola eksekusi yang tidak umum.

Identifikasi terhadap pola-pola ini tidak hanya penting untuk mendeteksi anomali, tetapi juga memberikan wawasan terhadap potensi ketidakefisienan proses, kesalahan operasional, maupun indikasi risiko yang tersembunyi.

Karakteristik proses pengajuan kredit yang bersifat sekuensial, memiliki variasi jalur yang tinggi, serta mengandung ketidakseimbangan antara kasus normal dan anomali menjadikannya sebagai tantangan dalam analisis data [3][18]. Selain itu, keterbatasan data berlabel serta dinamika perubahan kondisi bisnis semakin meningkatkan kompleksitas dalam deteksi anomali. Oleh karena itu, diperlukan pendekatan yang mampu mengakomodasi karakteristik tersebut, khususnya metode yang fleksibel, adaptif, serta mampu memanfaatkan data tidak berlabel untuk menghasilkan deteksi anomali yang lebih efektif sekaligus mendukung evaluasi dan optimalisasi proses bisnis perbankan.

2.5 Pseudo-Labeling Berbasis Clustering

Pada berbagai permasalahan klasifikasi, khususnya deteksi anomali, ketersediaan data yang telah memiliki label sering kali sangat terbatas. Salah satu pendekatan yang banyak digunakan untuk mengatasi kondisi tersebut adalah *pseudo-labeling*, yaitu teknik yang menghasilkan label sementara (*pseudo-label*) secara otomatis berdasarkan karakteristik data untuk dimanfaatkan dalam proses pembelajaran model [16]. Dengan pendekatan ini, data yang sebelumnya tidak memiliki label dapat dimanfaatkan tanpa memerlukan proses pelabelan manual terhadap seluruh data.

Secara umum, *pseudo-labeling* dilakukan dengan memanfaatkan model atau algoritma tertentu untuk menghasilkan label awal yang kemudian digunakan sebagai data latih bagi model pembelajaran berikutnya. Pendekatan ini telah banyak diterapkan pada model hibrida (*hybrid model*), yaitu dengan mengombinasikan algoritma yang memiliki karakteristik berbeda agar dapat saling melengkapi keunggulannya [11]. Pada pendekatan tersebut, algoritma pertama berperan menghasilkan representasi atau label awal dari data, sedangkan algoritma berikutnya memanfaatkan hasil tersebut untuk membangun model klasifikasi yang lebih baik.

Dalam konteks deteksi anomali, algoritma *clustering* sering digunakan sebagai pembangkit *pseudo-label* karena mampu mengidentifikasi struktur alami data tanpa memerlukan informasi label sebelumnya. Hasil pengelompokan tersebut kemudian digunakan sebagai masukan bagi algoritma klasifikasi untuk mempelajari

pola yang telah terbentuk. Pendekatan ini memungkinkan proses klasifikasi dilakukan meskipun data awal tidak memiliki label, sekaligus memanfaatkan karakteristik masing-masing algoritma secara lebih optimal [11].

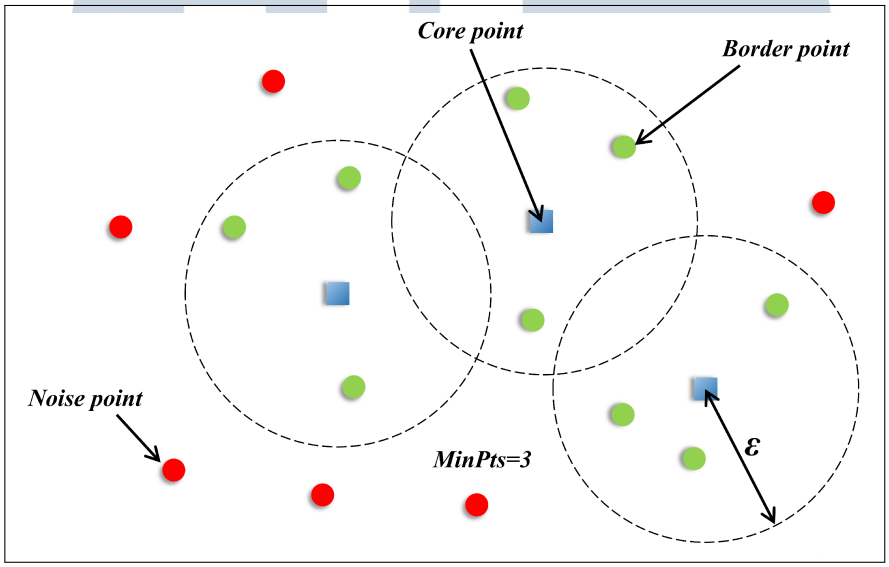
Pada penelitian ini, pendekatan *pseudo-labeling* diimplementasikan melalui kombinasi algoritma DBSCAN dan SVM. DBSCAN digunakan sebagai tahap awal untuk membentuk *pseudo-label* berdasarkan kepadatan data, di mana data yang teridentifikasi sebagai *noise* dianggap sebagai kandidat anomali, sedangkan data lainnya dianggap sebagai data normal. Berbeda dengan pendekatan *semi-supervised learning* klasik yang memanfaatkan kombinasi data berlabel dan data tidak berlabel, penelitian ini tidak menggunakan label aktual sama sekali. Seluruh label yang digunakan pada proses pelatihan diperoleh secara otomatis dari hasil *clustering*, sehingga pendekatan yang diterapkan lebih tepat dikategorikan sebagai pendekatan *hybrid* berbasis *pseudo-labeling* [11]. Selanjutnya, *pseudo-label* tersebut digunakan sebagai data latih bagi SVM untuk mempelajari pola pemisahan antara data normal dan kandidat anomali. Untuk memastikan bahwa *pseudo-label* yang dihasilkan cukup representatif, penelitian ini tidak hanya menggunakan hasil DBSCAN secara langsung, tetapi juga mengevaluasi performa model hasil pembelajaran SVM melalui data uji menggunakan *confusion matrix*, *precision*, *recall*, dan *F1-score*. Dengan demikian, penelitian ini tidak mengasumsikan bahwa seluruh *pseudo-label* yang dihasilkan DBSCAN telah benar, melainkan mengevaluasi sejauh mana label tersebut mampu menghasilkan model klasifikasi yang memiliki kemampuan generalisasi terhadap data yang belum pernah dilihat sebelumnya.

2.6 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

Clustering merupakan salah satu teknik dalam *unsupervised learning* yang bertujuan mengelompokkan data ke dalam beberapa *cluster* berdasarkan tingkat kemiripan tertentu tanpa memerlukan label kelas sebelumnya [3]. Dalam analisis proses bisnis, *clustering* digunakan untuk mempelajari pola aktivitas yang dominan serta mengidentifikasi pola yang jarang muncul atau menyimpang sehingga berpotensi menjadi anomali [6]. Karakteristik tersebut menjadikan *clustering* sesuai untuk data proses bisnis yang umumnya memiliki keterbatasan label.

Salah satu algoritma *clustering* yang banyak digunakan untuk deteksi anomali adalah *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN). DBSCAN merupakan algoritma berbasis kepadatan (*density-based*) yang membentuk *cluster* berdasarkan kerapatan titik dalam ruang fitur

menggunakan dua parameter utama, yaitu ε (*epsilon*) sebagai radius pencarian tetangga dan *MinPts* sebagai jumlah minimum titik untuk membentuk area padat [3, 6]. Berbeda dengan banyak algoritma *clustering* lainnya, DBSCAN tidak memerlukan jumlah *cluster* sejak awal dan mampu mengidentifikasi data yang tidak termasuk ke dalam *cluster* mana pun sebagai *noise*. Dalam konteks deteksi anomali, *noise* tersebut dapat diinterpretasikan sebagai kandidat anomali karena berada pada area dengan kepadatan yang rendah [9]. Ilustrasi konsep dasar DBSCAN ditunjukkan pada Gambar 2.1.



Gambar 2.1. Visualisasi konsep DBSCAN

Sumber: [19]

Berdasarkan Gambar 2.1, proses pembentukan cluster pada DBSCAN diawali dengan menentukan nilai parameter ε sebagai radius pencarian tetangga dan MinPts sebagai jumlah minimum titik yang harus berada di dalam radius tersebut. Untuk setiap titik data, algoritma menghitung jumlah titik lain yang berada pada ε-neighborhood. Apabila jumlah tetangga memenuhi atau melebihi nilai MinPts, maka titik tersebut dikategorikan sebagai core point. Selanjutnya, cluster dibentuk dengan menghubungkan core point yang saling density-reachable beserta titik-titik di sekitarnya. Titik yang berada dalam radius ε dari suatu core point tetapi tidak memiliki jumlah tetangga yang memenuhi nilai MinPts dikategorikan sebagai border point, sedangkan titik yang tidak termasuk dalam ε-neighborhood dari core point mana pun dikategorikan sebagai noise point. Dalam konteks deteksi anomali, noise point tersebut dapat diinterpretasikan sebagai kandidat anomali karena berada

pada area dengan kepadatan yang rendah [19].

DBSCAN mengelompokkan data berdasarkan konsep kepadatan lokal (*local density*) menggunakan dua parameter utama, yaitu ε sebagai radius pencarian tetangga dan *MinPts* sebagai jumlah minimum titik yang diperlukan untuk membentuk suatu area padat. Proses DBSCAN diawali dengan menghitung jarak antara suatu titik data dengan seluruh titik lainnya menggunakan fungsi jarak, yang umumnya berupa *Euclidean distance* pada Rumus 2.1. Nilai jarak tersebut digunakan untuk menentukan apakah suatu titik berada di dalam radius ε dari titik yang sedang diamati.

$$dist(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{2.1}$$

Pada Rumus 2.1, p_i dan q_i merupakan nilai atribut ke-i dari dua titik data yang dibandingkan, sedangkan n menyatakan jumlah atribut atau dimensi data. Semakin kecil nilai *Euclidean distance*, semakin dekat posisi kedua titik di dalam ruang fitur. Berdasarkan nilai jarak tersebut, himpunan tetangga suatu titik dalam radius ε didefinisikan menggunakan ε-*neighborhood* pada Rumus 2.2.

$$N_\varepsilon(p) = \{q \in D \mid dist(p,q) \leq \varepsilon\} \tag{2.2}$$

Pada Rumus 2.2, D merupakan himpunan seluruh titik data, sedangkan $N_\varepsilon(p)$ menyatakan seluruh titik yang berada dalam radius ε dari titik p. Suatu titik dikategorikan sebagai *core point* apabila jumlah tetangganya memenuhi atau melebihi nilai *MinPts*, sebagaimana ditunjukkan pada Rumus 2.3.

$$|N_\varepsilon(p)| \geq MinPts \tag{2.3}$$

Titik yang memenuhi kondisi pada Rumus 2.3 dikategorikan sebagai *core point*, yaitu titik yang berada pada area dengan kepadatan tinggi dan berperan sebagai pusat pembentukan *cluster*. Sebaliknya, titik yang tidak memenuhi syarat sebagai *core point* tetapi masih berada di dalam ε-*neighborhood* dari suatu *core point* dikategorikan sebagai *border point*. Kriteria *border point* ditunjukkan pada Rumus 2.4.

$$|N_\varepsilon(p)| < MinPts \wedge \exists q \in D : p \in N_\varepsilon(q) \wedge |N_\varepsilon(q)| \geq MinPts \tag{2.4}$$

Pada Rumus 2.4, suatu titik p dikategorikan sebagai *border point* apabila jumlah titik di dalam ε-*neighborhood*-nya kurang dari nilai *MinPts*, namun titik tersebut masih berada di dalam ε-*neighborhood* dari suatu *core point* q. Dengan demikian, *border point* menjadi bagian dari suatu *cluster*, tetapi tidak memiliki kepadatan yang cukup untuk membentuk atau memperluas *cluster* secara mandiri.

Sementara itu, titik yang tidak berada dalam jangkauan *core point* mana pun dikategorikan sebagai *noise*. Hubungan antar *core point* dalam pembentukan *cluster* dijelaskan melalui konsep *density-reachable*, sebagaimana ditunjukkan pada Rumus 2.5.

$$p_1 = p, \quad p_n = q \tag{2.5}$$

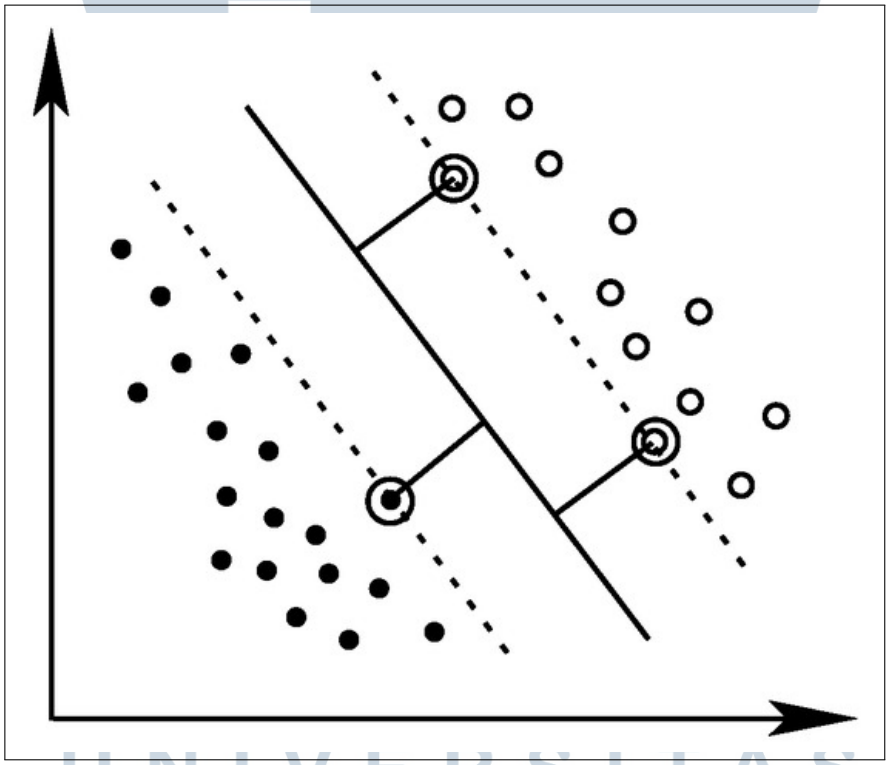
Suatu titik q dikatakan *density-reachable* dari titik p pada Rumus 2.5 apabila terdapat rangkaian *core point* yang saling *directly density-reachable* hingga menghubungkan kedua titik tersebut. Melalui mekanisme ini, DBSCAN membentuk *cluster* berdasarkan kepadatan lokal, sedangkan titik yang tidak memenuhi kriteria tersebut dikategorikan sebagai *noise*, sebagaimana ditunjukkan pada Rumus 2.6.

$$|N_\varepsilon(p)| < MinPts \wedge q \in D : p \in N_\varepsilon(q) \wedge |N_\varepsilon(q)| \geq MinPts \tag{2.6}$$

Pada Rumus 2.6, suatu titik dikategorikan sebagai *noise* apabila tidak memenuhi syarat sebagai *core point* maupun *border point*, serta tidak berada dalam ε-*neighborhood* dari *core point* mana pun. Dalam penelitian ini, *noise* diinterpretasikan sebagai kandidat anomali karena berada di luar area dengan kepadatan tinggi dan tidak mengikuti pola mayoritas data.

2.7 Support Vector Machine (SVM)

Support Vector Machine (SVM) merupakan salah satu algoritma *machine learning* yang digunakan untuk tugas klasifikasi dengan membangun batas keputusan (*hyperplane*) yang mampu memisahkan dua atau lebih kelas secara optimal [4]. Tujuan utama SVM adalah memperoleh *hyperplane* optimal yang menghasilkan margin maksimum, yaitu jarak terjauh antara batas keputusan dengan titik data terdekat dari masing-masing kelas. Margin yang lebih besar memungkinkan model memiliki kemampuan generalisasi yang lebih baik sehingga proses klasifikasi menjadi lebih robust terhadap data baru [20]. Ilustrasi pembentukan *hyperplane*, *margin*, dan *support vectors* pada SVM ditunjukkan pada Gambar 2.2.



Gambar 2.2. Visualisasi konsep *Support Vector Machine* (SVM)

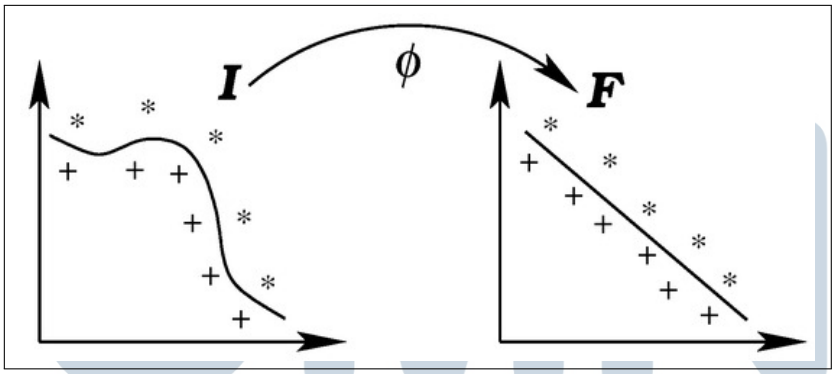
Sumber: [20]

Berdasarkan Gambar 2.2, *hyperplane* berfungsi sebagai batas keputusan yang memisahkan dua kelas data. Margin merupakan jarak antara *hyperplane* dengan titik data terdekat dari masing-masing kelas. Titik-titik tersebut disebut sebagai *support vectors* karena memiliki pengaruh terbesar dalam menentukan posisi *hyperplane*. Selama proses pelatihan, SVM mencari *hyperplane* yang

menghasilkan margin maksimum sehingga diperoleh batas keputusan yang optimal. Setelah model terbentuk, setiap data baru diklasifikasikan berdasarkan posisinya terhadap *hyperplane* tersebut [20]. Secara matematis, proses pengambilan keputusan pada SVM dinyatakan menggunakan fungsi keputusan pada Rumus 2.7.

$$f(x) = w \cdot x + b \tag{2.7}$$

Pada Rumus 2.7, x merupakan vektor fitur, w adalah vektor bobot yang menentukan orientasi *hyperplane*, sedangkan b merupakan bias yang menggeser posisi *hyperplane*. Nilai fungsi keputusan digunakan untuk menentukan kelas suatu data berdasarkan posisinya terhadap batas keputusan yang telah dibentuk. Pada SVM linear, pemisahan dilakukan secara langsung pada ruang fitur asli. Namun, apabila data tidak dapat dipisahkan secara linear, SVM memanfaatkan fungsi kernel melalui pendekatan *kernel trick* untuk memetakan data ke ruang fitur berdimensi lebih tinggi tanpa menghitung transformasi tersebut secara eksplisit [20]. Ilustrasi konsep *kernel trick* yang memetakan data dari *input space* ke *feature space* ditunjukkan pada Gambar 2.3.



Gambar 2.3. Visualisasi konsep *kernel trick*

Sumber: [20]

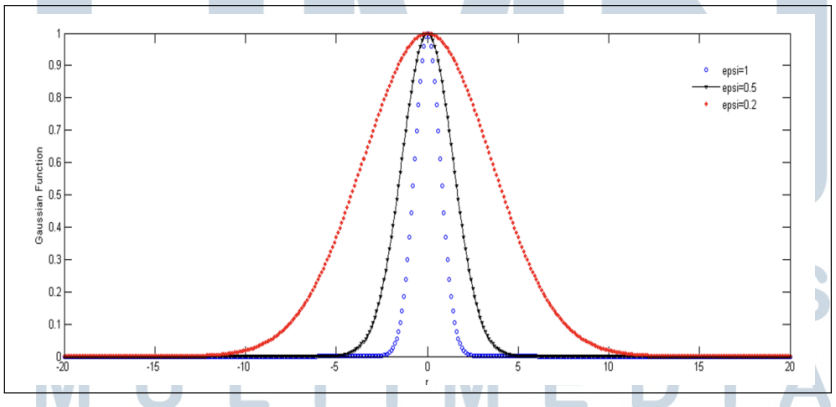
Berdasarkan Gambar 2.3, distribusi data pada *input space* (I) awalnya tidak dapat dipisahkan secara linear. Melalui fungsi pemetaan ϕ, data diproyeksikan ke *feature space* (F) yang memiliki dimensi lebih tinggi sehingga kedua kelas menjadi lebih mudah dipisahkan menggunakan *hyperplane* linear. Pendekatan ini dikenal sebagai *kernel trick*, yaitu mekanisme yang memungkinkan data diperlakukan seolah-olah telah dipetakan ke ruang fitur berdimensi lebih tinggi tanpa menghitung proses pemetaan tersebut secara eksplisit. Dengan representasi

tersebut, pola data yang sebelumnya sulit dipisahkan secara linear menjadi lebih mudah diklasifikasikan menggunakan SVM [20].

Salah satu fungsi kernel yang umum digunakan untuk menerapkan *kernel trick* adalah *Radial Basis Function* (RBF). Kernel RBF menghitung tingkat kemiripan antar pasangan data berdasarkan jarak Euclidean antar sampel, kemudian secara implisit memetakan data ke ruang fitur berdimensi lebih tinggi. Dengan representasi tersebut, data yang semula tidak dapat dipisahkan secara linear menjadi lebih mudah dipisahkan menggunakan *hyperplane*. Oleh karena itu, kernel RBF dipilih pada penelitian ini karena sesuai untuk menangani karakteristik data proses bisnis yang memiliki pola kompleks dan hubungan non-linear [20, 21]. Fungsi kernel RBF ditunjukkan pada Rumus 2.8.

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2) \tag{2.8}$$

Pada Rumus 2.8, x_i dan x_j merupakan dua vektor fitur yang dibandingkan, γ merupakan parameter yang mengatur tingkat pengaruh jarak antar data terhadap nilai fungsi kernel, sedangkan $\|x_i - x_j\|^2$ menyatakan jarak kuadrat antara kedua vektor fitur. Nilai fungsi kernel akan semakin besar apabila dua data memiliki jarak yang semakin dekat, dan sebaliknya akan semakin kecil apabila jarak antar data semakin jauh [21]. Pengaruh parameter γ terhadap bentuk fungsi RBF diilustrasikan pada Gambar 2.4.



Gambar 2.4. Visualisasi fungsi kernel Radial Basis Function (RBF)

Sumber: [21]

Berdasarkan Gambar 2.4, semakin besar nilai γ, kurva Gaussian menjadi semakin sempit sehingga hanya data yang sangat berdekatan yang memiliki nilai

kemiripan tinggi. Sebaliknya, nilai γ yang lebih kecil menghasilkan kurva yang lebih lebar sehingga data yang memiliki jarak relatif lebih jauh masih dianggap memiliki tingkat kemiripan yang cukup tinggi. Dengan demikian, parameter γ menentukan tingkat kompleksitas batas keputusan yang dibentuk oleh SVM [21]. Pada penelitian ini, parameter γ menggunakan nilai *scale* sehingga nilainya ditentukan secara otomatis berdasarkan karakteristik data. Pengaturan tersebut dipilih agar model mampu membentuk batas keputusan yang fleksibel tanpa memerlukan penentuan nilai γ secara manual.

Selain parameter γ, SVM juga menggunakan parameter regularisasi C untuk mengatur keseimbangan antara memaksimalkan margin dan meminimalkan kesalahan klasifikasi. Nilai C yang lebih besar mendorong model untuk meminimalkan kesalahan klasifikasi pada data latih, sedangkan nilai C yang lebih kecil menghasilkan margin yang lebih lebar dengan toleransi terhadap beberapa kesalahan klasifikasi [20]. Pada penelitian ini digunakan nilai $C = 10$ untuk memperoleh batas keputusan yang tetap mampu melakukan generalisasi terhadap data.

Berdasarkan uraian tersebut, SVM dengan kernel RBF dipilih pada penelitian ini karena mampu menangani pola data yang kompleks dan tidak dapat dipisahkan secara linear. Melalui kombinasi parameter kernel RBF, γ, dan parameter regularisasi C, model SVM digunakan untuk mempelajari pola data normal maupun anomali berdasarkan *pseudo-label* yang dihasilkan oleh DBSCAN sehingga dapat menghasilkan model klasifikasi yang memiliki kemampuan generalisasi yang baik.

2.8 Standarisasi Data

Standardisasi data merupakan salah satu tahap *preprocessing* yang bertujuan untuk menyamakan skala setiap fitur sehingga memiliki rata-rata (*mean*) sebesar 0 dan simpangan baku (*standard deviation*) sebesar 1. Pada penelitian ini, proses standardisasi dilakukan menggunakan metode *StandardScaler* yang tersedia pada pustaka *scikit-learn*. Tahap ini bertujuan untuk mengurangi pengaruh perbedaan rentang nilai antar fitur yang dapat menyebabkan fitur tertentu mendominasi proses pembelajaran model. Standardisasi menjadi penting terutama pada algoritma yang bergantung pada perhitungan jarak, seperti DBSCAN, maupun algoritma yang sensitif terhadap skala fitur, seperti Support Vector Machine (SVM) [22]. Sebelum proses standardisasi dilakukan, terlebih dahulu dihitung nilai rata-rata (*mean*) dari

setiap fitur menggunakan Rumus 2.9.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.9}$$

Pada Rumus 2.9, x_i merupakan nilai data ke-i, sedangkan N menunjukkan jumlah seluruh data pada suatu fitur. Nilai μ digunakan sebagai pusat distribusi data yang akan menjadi acuan dalam proses standardisasi. Selanjutnya dihitung nilai simpangan baku (*standard deviation*) yang menunjukkan tingkat penyebaran data terhadap nilai rata-ratanya menggunakan Rumus 2.10.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{2.10}$$

Pada Rumus 2.10, σ menyatakan simpangan baku, x_i merupakan nilai data ke-i, μ adalah nilai rata-rata, dan N merupakan jumlah data. Nilai simpangan baku digunakan untuk mengukur seberapa jauh penyebaran data terhadap rata-ratanya. Semakin besar nilai σ, semakin besar variasi data pada fitur tersebut. Setelah nilai rata-rata dan simpangan baku diperoleh, setiap nilai fitur distandardisasi menggunakan StandardScaler Rumus 2.11.

$$z = \frac{x - \mu}{\sigma} \tag{2.11}$$

Pada Rumus 2.11, x merupakan nilai asli suatu fitur, sedangkan μ dan σ berturut-turut merupakan nilai rata-rata dan simpangan baku dari fitur tersebut. Hasil standardisasi berupa nilai z (z-*score*) yang menunjukkan posisi suatu data relatif terhadap rata-ratanya. Setelah proses standardisasi, setiap fitur memiliki rata-rata mendekati 0 dan simpangan baku sebesar 1 sehingga seluruh fitur berada pada skala yang sebanding.

Pada penelitian ini, standardisasi dilakukan setelah proses ekstraksi fitur dan sebelum penerapan algoritma DBSCAN serta SVM. Tahap ini bertujuan agar fitur-fitur dengan rentang nilai yang berbeda tidak mendominasi perhitungan jarak pada DBSCAN maupun proses pembentukan *hyperplane* pada SVM, sehingga kedua algoritma dapat bekerja secara lebih optimal [22].

2.9 Evaluasi Model

Evaluasi model merupakan tahap penting untuk mengukur performa model dalam melakukan klasifikasi maupun deteksi anomali. Pada penelitian ini, evaluasi dilakukan menggunakan *confusion matrix* beserta metrik turunannya, yaitu *accuracy*, *precision*, *recall*, dan *F1-score* [23]. Mengingat karakteristik data yang tidak seimbang (*imbalanced*), di mana jumlah data normal lebih banyak dibandingkan data anomali, evaluasi tidak hanya mengacu pada nilai *accuracy*. Metrik *precision*, *recall*, dan *F1-score* juga digunakan agar kemampuan model dalam mendeteksi kelas anomali dapat dinilai secara lebih representatif.

Confusion matrix menyajikan perbandingan antara hasil prediksi model dengan label aktual dalam bentuk matriks dua dimensi yang terdiri atas empat komponen utama, yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). *True Positive* menunjukkan jumlah data anomali yang berhasil dideteksi dengan benar, sedangkan *True Negative* merupakan jumlah data normal yang diklasifikasikan dengan benar. *False Positive* terjadi ketika data normal diklasifikasikan sebagai anomali (*false alarm*), sedangkan *False Negative* menunjukkan data anomali yang gagal dideteksi dan berpotensi menimbulkan risiko.

Berdasarkan *confusion matrix*, beberapa metrik evaluasi dapat dihitung menggunakan rumus sebagai berikut. Metrik pertama yang digunakan adalah *accuracy*, yang dirumuskan pada Rumus 2.12.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.12}$$

Accuracy mengukur proporsi prediksi yang benar terhadap seluruh data. Namun, metrik ini kurang representatif pada dataset yang tidak seimbang. Oleh karena itu, diperlukan metrik lain seperti *precision* yang dirumuskan pada Rumus 2.13.

$$Precision = \frac{TP}{TP + FP} \tag{2.13}$$

Precision mengukur ketepatan model dalam mengidentifikasi data anomali serta menunjukkan kemampuan model dalam mengurangi *false positive*. Selain itu, *recall* juga digunakan untuk mengukur kemampuan model dalam mendeteksi

seluruh data anomali, yang dirumuskan pada Rumus 2.14.

$$Recall = \frac{TP}{TP + FN} \tag{2.14}$$

Nilai *recall* yang tinggi menunjukkan bahwa model mampu meminimalkan *false negative*. Untuk menyeimbangkan kedua metrik tersebut, digunakan *F1-score* sebagai rata-rata harmonis dari *precision* dan *recall*, yang dirumuskan pada Rumus 2.15.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.15}$$

F1-score merupakan rata-rata harmonis antara *precision* dan *recall*, yang memberikan evaluasi yang lebih seimbang, terutama pada kondisi data yang tidak seimbang. Oleh karena itu, metrik ini sering digunakan dalam deteksi anomali untuk menilai performa model secara lebih komprehensif.

2.10 Splitting Dataset

Splitting dataset merupakan tahapan dalam pengembangan model *machine learning* untuk membagi data menjadi *training set* dan *testing set* [13]. *Training set* digunakan untuk melatih model, sedangkan *testing set* digunakan untuk mengevaluasi performa model pada data yang tidak digunakan selama proses pelatihan, sehingga dapat mengurangi risiko *overfitting*.

Dalam konteks penelitian ini, pembagian data tetap diperlukan agar proses evaluasi tidak dipengaruhi langsung oleh data yang digunakan untuk melatih model [3]. Pada penelitian ini, pembagian data dilakukan setelah pembentukan *pseudo-label* dari hasil DBSCAN. Data kemudian dibagi menggunakan skema *train-test split* dengan proporsi 70:30, yaitu 70% data digunakan untuk pelatihan SVM dan 30% digunakan untuk pengujian. Pembagian dilakukan dengan mempertahankan proporsi kelas normal dan anomali agar evaluasi model tetap representatif pada kondisi data yang tidak seimbang.