BAB 2
KAJIAN PUSTAKA

Pada bab ini dijelaskan tentang teori-teori yang mendasari permasalahan dan penyelesaian tesis, baik perangkat keras, perangkat lunak, dan penelitian terdahulu yang digunakan untuk perancangan tesis ini. Beberapa topik pembahasan yang digunakan dalam penelitian ini meliputi: Suku Cadang, *Machine Learning*, Tahapan *Machine Learning*, *Synthetic Minority Oversampling Technique (SMOTE), Principal Component Analysis (PCA), K-Nearest Neighbor* (*KNN*), Evaluasi Kinerja Model dan *Dashboard*. Bab ini juga membahas penelitian terdahulu dengan beberapa jumlah penelitian.

2.1 TEORI PENUNJANG

2.1.1 Suku Cadang

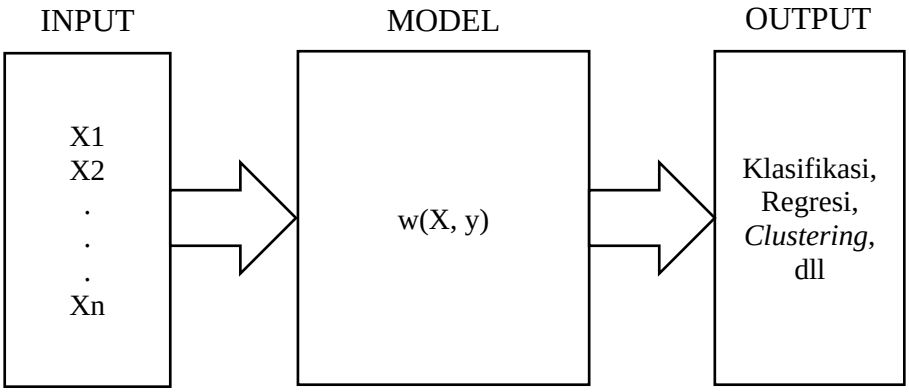
Suku cadang pemeliharaan (*MRO*) merupakan komponen yang diperlukan untuk menjaga keandalan peralata proses seperti pompa, kompresor, mesin, motor, dan lainnya sehingga gangguan tidak menjadi *downtime* berkepanjangan. Dalam lingkup industri migas, suku cadang memiliki karakter persediaan yang beragam dengan item atau suku cadang yang sangat banyak, pola permintaan sporadis dan konsekuensi stok kritikal kosong jauh lebih mahal daripada stok non kritikal *overstock* sehingga klasifikasi atau penetapan suku cadang kritikal dan non kritikal dan peramalan permintaan merupakan fondasi pengendalian persediaan. Pada *literature review* menunjukkan adanya celah praktik riset pada klasifikasi dan peramalan suku cadang dan menekankan kebutuhan kerangka keputusan yang konsisten di industri proses kontinu. (Bacchetti & Saccani, 2012)

Pada sisi permintaan, tinjauan terbaru juga menegaskan teknik peramalan untuk *intermittent demand* yang menjadi isu sentral pada suku cadang karena data yang jarang menunjukkan kualitas peramalan berpengaruh langsung pada level

stok, *service level*, dan biaya. Pembaruan kaji tahun 2021 sampai dengan 2024 menekankan kebutuhan klasifikasi *criticality*, peramalan, dan kebijakan persediaan agar risiko opreasional dapat ditekan tanpa menumpuk biaya simpan. Oleh karena itu, pentingnya integrasi data dan kebijakan persediaan berbasis *criticality* untuk menjaga keandalan operasi. (Zakizadeh, 2024)

2.1.2 *Machine Learning*

Machine Learning merupakan salah satu cabang dari kecerdasan buatan yang membahas mengenai pembangunan sistem yang didapat berdasarkan pada pembelajaran data, atau sebuah studi yang mempelajari cara untuk memprogram sebuah komputer untuk belajar. Inti dari *machine learning* adalah representasi dan generalisasi (Christopher M. Bishop, 2007).



Gambar 2. 1 Machine Learning secara umum (Christopher M. Bishop, 2007)

Pada Gambar 2.1 menunjukkan pembelajaran mesin secara umum, dimana *Input* merupakan representasi data dan pemilihan ekstraksi fitur-fitur tertentu, dimana sekumpulan fitur-fitur yang memberikan deskripsi sebuah objek dinyatakan dengan $X = [X1, X2, X3, \dots, Xn]$. Dalam mengenali sebuah objek sehingga mendapatkan informasi data dari pembagian area fitur-fitur tersebut, maka diperlukan model $W = [X, y]$ tertentu yang digunakan dan berfungsi sebagai model *machine learning* yang menggunakan algoritma-algoritma tertentu sehingga diperoleh *output* yang berupa informasi dari data yang telah diproses, sehingga menghasilkan klasifikasi, regresi, ataupun pengklasteran data.

Salah satu bagian dari *artificial intelligence* adalah *Machine Learning* yang merupakan mencakup perancangan dan pengembangan dengan disiplinnya

algoritma dari suatu komputer dalam pengembangan perilaku berdasarkan data yang nyata (Jackson, 1988), terdapat hal dalam pembelajaran *Machine Learning*, seperti:

1. *Supervised Learning*

   Menggunakan masukan yang diberi tanda dengan label, yang setelahnya akan membuat perencanaan dari data yang telah diberi tanda

2. *Unsupervised Learning*

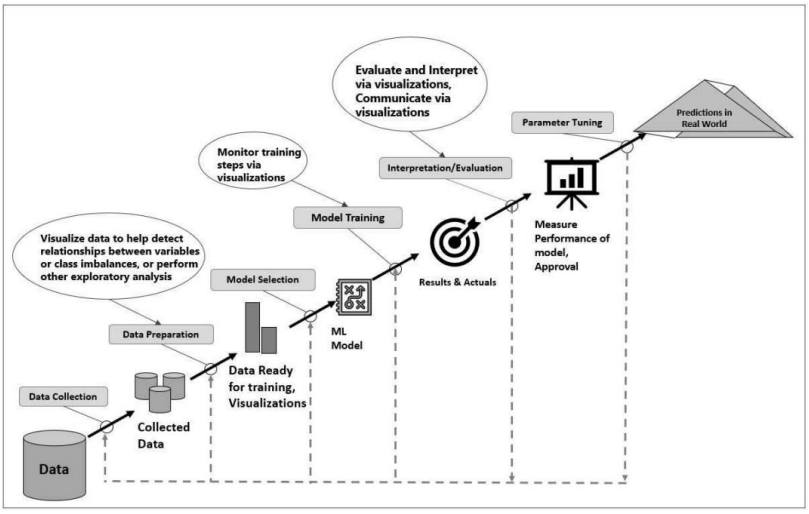
   Pelajaran yang di gunakan untuk memasukan data yang tidak menggunakan label, namun akan menyatukan data yang dijumpai

3. *Reinforcement learning*

   Tujuannya adalah untuk mengabungkan berbagai berita pelajaran secara aktif yang bekerja sama terhadap lingkungan untuk mendapatkan balasan kegiatan dari pembelajaran tersebut.

2.1.3 Tahapan *Machine Learning*

Aplikasi dan sistem *Machine Learning* membutuhkan manusia untuk berpartisipasi di hamper semua proses analisis. Keterlibatan ini bergantung pada faktor-faktor seperti model data, domain data, dan Langkah *input machine learning* (Eisler & Meyer, 2020). Tahapan *Machine Learning* dapat digambarkan pada Gambar 2.2.



Gambar 2. 2 Tahapan Machine Learning (Eisler & Meyer, 2020)

2.1.3.1 Data Collection

Data Collection adalah proses pengumpulan dan pengukuran berbagai informasi dari berbagai sumber. Tujuan dari tahapan ini adalah untuk mengumpulkan data yang relevan dan akurat untuk digunakan dalam proses *machine learning*. Data yang dikumpulkan dapat berasal dari berbagai sumber, seperti *database*, file, atau bahkan pengumpulan data secara manual. Data yang dikumpulkan haruslah relevan dan akurat untuk memastikan kualitas data yang digunakan dalam proses *machine learning*.

2.1.3.2 Data Preparation

Data Preparation adalah proses untuk mengubah data mentah menjadi format yang dapat digunakan oleh algoritma *machine learning*. Tahapan ini melibatkan beberapa langkah, seperti:

1. *Data Cleaning*: Menghapus data yang tidak relevan, tidak akurat, atau tidak lengkap.
2. *Data Transformation*: Mengubah format data menjadi format yang lebih sesuai untuk digunakan oleh algoritma *machine learning*.
3. *Data Normalization*: Mengubah skala data menjadi skala yang sama untuk memastikan bahwa data yang berbeda skala dapat digunakan dengan baik.

Tahapan Data *Preparation* sangat penting karena data yang tidak siap dapat berdampak pada kualitas hasil yang dihasilkan oleh algoritma *machine learning*.

2.1.3.3 Exploratory Visualization dan *Monitoring*

Visualisasi eksploratori digunakan sejak awal untuk membaca data dan mengantisipasi masalah umum seperti ketidakseimbangan kelas dengan *bar chart*, sebaran dan *outlier* dengan histogram atau *boxplot*, serta korelasi atau redundansi antar fitur dengan *heatmap korelasi*. Pada tahap evaluasi ini, divisualisasikan dengan *confusion matrix*, kurva *ROC* dan khususnya *Precision Recall* (*PR*) untuk membantu menilai performa pada data tidak seimbang, *PR* kerap lebih informatif daripada *ROC* ketika kelas minoritas yang jarang. Untuk reduksi dimensi, *scree plot* dan *loadings PCA* menjelaskan berapa banyak variasi yang ditangkap dan fitur apa yang dominan. Setelah model berjalan, monitoring berkala seperti tren metrik,

stabilitas skor atau kalibrasi, dan deteksi *drift* pergeseran data atau konsep penting agar degradasi kinerja cepat terdeteksi dan ditindaklanjuti dengan *retrain* atau penyesuaian *threshold*. Hal tersebut sejalan dengan rekomendasi riset tentang visualisasi untuk interpretabilitas, evaluasi pada data imbang dan tak imbang, serta keandalan sistem *machine learning* di produksi. (Bayram & Ahmed, 2025)

2.1.3.4 Model Selection

Model Selection adalah proses pemilihan satu model akhir *machine learning* dari kumpulan model kandidat *machine learning* untuk pelatihan dataset. Tahapan ini melibatkan beberapa langkah, seperti:

1. Mengidentifikasi Tujuan: Mengidentifikasi tujuan dari proses *machine learning* seperti klasifikasi, regresi, atau clustering.
2. Mengidentifikasi Algoritma: Mengidentifikasi algoritma *machine learning* yang paling sesuai untuk tujuan yang diidentifikasi.
3. Menguji Model: Menguji beberapa model *machine learning* yang dipilih untuk membandingkan performanya.

Tahapan Model Selection sangat penting karena memungkinkan untuk memilih model yang paling sesuai untuk tujuan yang diidentifikasi sehingga hasil yang dihasilkan lebih akurat dan efektif.

2.1.3.5 Model Training

Model *Training* adalah bagian utama dalam proses *machine learning*. Pada tahapan ini, model *machine learning* yang dipilih sebelumnya dilatih menggunakan data pelatihan. Data pelatihan digunakan untuk mengajarkan model melakukan klasifikasi, regresi, atau *clustering*. Tahapan ini melibatkan beberapa langkah, seperti:

1. Mengumpulkan Data Pelatihan: Mengumpulkan data pelatihan yang relevan dan akurat.
2. Mengajarkan Model: Mengajarkan model *machine learning* menggunakan data pelatihan.
3. Mengoptimalkan Model: Mengoptimalkan model *machine learning* untuk memastikan kualitas hasil yang dihasilkan.

Tahapan Model *Training* sangat penting karena memungkinkan untuk mengajarkan model *machine learning* untuk melakukan klasifikasi, regresi, atau *clustering* dengan akurat.

2.1.3.6 Evaluating Model

Setelah *model training* selesai, selanjutnya yaitu melihat model yang digunakan apakah bagus dengan menggunakan *Evaluating Model*. *Evaluating Model* memungkinkan untuk menguji model terhadap data yang belum pernah digunakan untuk pelatihan. Metrik ini memungkinkan untuk melihat bagaiamana performa model pada data yang belum terlihat. Tahapan ini melibatkan beberapa langkah, seperti:

1. Mengumpulkan Data Uji: Mengumpulkan data uji yang relevan dan akurat.
2. Menguji Model: Menguji model *machine learning* menggunakan data uji.
3. Menghitung Metrik: Menghitung metrik yang relevan, seperti akurasi, presisi, dan *recall*.

2.1.3.7 Parameter Tuning

Penalaan hiperparameter bertujuan memilih kombinasi pengaturan model misal pada *k*, metrik jarak, pembobotan tetangga pada *kNN* yang memaksimalkan kinerja generalisasi tanpa bias. Hal ini adalah melakukan pencarian *grid* atau *random* di dalam kerangka *cross validation* agar proses seleksi tidak melihat data uji untuk mencegah optimisme berlebih. Sehingga banyak studi menganjurkan *nested cross validation* ketika pemilihan *hyperparameter* karena dapat menyebabkan bias estimasi performa pada data kecil. (Vabalas et al., 2019)

Dibanding *grid search* yang menyapu titik yang seragam pada ruang *hyperarameter*, *random search* jauh lebih efisien di ruang berdimensi tinggi karena menemukan nilai penting dengan biaya evaluasi yang lebih sedikit (Bergstra & Bengio, 2012). Selain itu, ada algoritma *resource-aware* seperti *Hyperband* yang mengalokasikan anggaran komputasi secara adaptif sehingga kombinasi *hyperparameter* yang menjanjikan mendapat sumber daya lebih banyak dan mempercepat proses *tuning* pada skenario biaya latih yang mahal.

Untuk ruang *hyperparameter* yang kontinu atau campuran dan biaya evaluasi, *Bayesian Optimization* memberikan pendekeatan yang lebih *sample efficient* dengan menyeimbangkan eksplorasi dan eksploitasi melalui fungsi akuisisi. (Shahriari et al., 2016)

Pada akhirnya, pemilihan fungsi skoring harus selaras dengan tujuan bisnis karakter data di mana pada klasifikasi tidak seimbang, metrik seperti *F1* atau *AUC-PR* lebih informatif dari akurasi murni. Hal ini didukung oleh studi yang menunjukkan *PR curve* atau *AUC-PR* lebih sensitif terhadap perubahan preforma pada kelas minoritas dibanding *ROC* di pengaturan *imbalance*. (Saito & Rehmsmeier, 2015)

2.1.4 *SMOTE*

SMOTE membangkitkan contoh sintetis di sekitar sampel minoritas lewat interpolasi ke tetangga terdekat untuk menaikkan kerapatan kelas minoritas di area batas keputusan. Teknik ini dirancang untuk mengatasi masalah ketidakseimbangan kelas (*imbalanced data*) yang dapat menurunkan kemampuan model dalam mengenali pola pada kelas minoritas. Dalam pembelajaran mesin, kelas merujuk pada label target dalam masalah klasifikasi seperti kelas mayoritas dan minoritas. Ketidakseimbangan kelas menyebabkan algoritma berbasis jarak seperti *KNN* cenderung bias terhadap kelas mayoritas karena sebagian besar tetangga terdekat berasal dari kelas tersebut. Kondisi ini menggeser *decision boundary* dan membuat model lebih sulit mempelajari pola kelas minoritas. *SMOTE* menyeimbangkan kelas dengan meningkatkan kepadatan sampel minoritas di ruang fitur sehingga representasi antar kelas menjadi lebih setara dan peluang model untuk mempelajari pola kelas minoritas meningkat.

Meskipun demikian, penyeimbangan kelas tidak selalu menjamin peningkatan performa terutama pada ketidakseimbangan yang tidak ekstrem karena interpolasi sintetis dapat menambah *class overlap*. Secara metodologis, *SMOTE* tetap menjadi teknik standar untuk mengurangi bias akibat ketidakseimbangan kelas dalam algoritma klasifikasi berbasis jarak. *SMOTE* dapat efektif pada ketidakseimbangan moderat tinggi, tetapi dapat menambah *class overlap* dan menurunkan *precision* bila *k_neighbors* terlalu besar, data sangat *noisy* atau

berdimensi tinggi, atau tumpang tindih antar kelas besar. Karena itu implementasi yang disarankan adalah menempatkan *SMOTE* hanya pada data latih di setiap *fold* (*Pipeline* dan *Stratified K-Fold*) guna mencegah *leakage*, menggunakan standardisasi numerik (serta *winsorizing* ringan) sebelum *oversampling*, dan apabila *PCA* digunakan maka *SMOTE* diterapkan setelah transformasi *PCA* agar interpolasi berlangsung di ruang fitur yang telah terdekorrelasi. Untuk pengaturan awal, disarankan menggunakan *sampling_strategy='auto'* dan nilai *k_neighbors* kecil (1–3). Bila *precision* menurun, varian *SMOTE* seperti *Borderline-SMOTE* atau kombinasi pembersihan seperti *SMOTE–Tomek* dan *SMOTE–ENN* dapat dipertimbangkan. Untuk data campuran numerik–kategorikal digunakan *SMOTE-NC* (Aguiar et al., 2024; Kaur et al., 2020).

Dari sisi evaluasi, pada data timpang *accuracy* dan bahkan *ROC-AUC* dapat menyesatkan sehingga fokus pada *F1* dan *PR-AUC* (*average precision*) lebih disarankan. Prevalensi kelas perlu dilaporkan agar *precision* tidak bias dan perbandingan performa dengan dan tanpa *SMOTE* dianalisis menggunakan *PR curve*, *confusion matrix*, serta *F1/PR-AUC*. Seluruh pemilihan model dilakukan melalui validasi silang terstratifikasi agar estimasi performa tidak optimistis. Langkah implementasi ini konsisten dengan tinjauan ketidakseimbangan kelas oleh Luque et al. (2019), survei *imbalance learning* oleh Kaur et al. (2020), serta *review* komprehensif *SMOTE modern* oleh Aguiar et al. (2024).

2.1.5 Principal Component Analysis (PCA)

Principal Component Analysis (*PCA*) memproyeksikan fitur numerik ke sekumpulan komponen ortogonal yang menangkap varians terbesar secara berurutan sehingga efektif untuk mengurangi dimensi, kolinearitas, dan *noise* terutama bermanfaat bagi metode berbasis jarak seperti *kNN* yang sensitif terhadap skala dan *curse of dimensionality*. Praktik yang dianjurkan adalah menempatkan *PCA* di dalam *pipeline* dan hanya di-*fit* pada data latih di setiap *fold* (*anti-leakage*), setelah standardisasi fitur numerik. Karena *PCA* bersifat tidak terawasi (*unsupervised*), memilih target varians kumulatif (mis. 70–95%) menjadi aturan praktis untuk menyeimbangkan bias *variance* dan biaya komputasi dimana target terlalu rendah bisa membuang sinyal diskriminatif, terlalu tinggi kurang memberi

penghematan dimensi. Evaluasi beberapa target (mis. 0.70/0.80/0.90/0.95) dan bandingkan metrik *F1* serta *PR-AUC*, dibantu *scree plot* dan kurva *cumulative explained variance* untuk melihat titik lutut.

Pada konteks *kNN*, *PCA* dapat merapikan ruang jarak dengan menekan redundansi dan *noise* sehingga tetangga terdekat lebih representatif. Rancang *pipeline* blok-spesifik, terapkan *PCA* hanya pada fitur numerik, sementara fitur kategorikal di-*One-Hot Encoding* dibiarkan apa adanya (*PCA* kurang tepat untuk dummy spars). Setelah *OHE*, lakukan *densification* bila algoritma *downstream* membutuhkannya, tetapi hindari mem-*fit PCA* pada gabungan numerik dan kategorikal agar komponen tidak didominasi frekuensi dummy. Untuk skenario *imbalanced* yang memakai *SMOTE*, terapkan urutan yang konsisten yaitu scale, *PCA* pada numerik, *SMOTE* pada ruang hasil transformasi, kemudian *kNN* agar interpolasi sintetis terjadi pada representasi yang lebih terdekorrelasi. Keseluruhan praktik ini sejalan dengan survei reduksi dimensi modern dan tinjauan *kNN* dan variannya yang menekankan pentingnya *preprocessing* skala atau dimensi bagi kualitas tetangga dan stabilitas metrik.

2.1.6 *K-Nearest Neighbor (KNN)*

KNN merupakan metode melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. Metode ini bertujuan untuk mengklasifikasikan objek baru berdasarkan atribut dan *training sample* (Pangestu & Noris, 2023). *KNN* akan mengelempokkan hasil perhitungan dengan *data training* yang mempunyai kerabat terbanyak dalam nilai jangkauan yang ditentukan. Jarak antara data latih dan data uji dihitung menggunakan persamaan *Euclidean*. Pengklasifikasian tidak menggunakan model apapun untuk dicocokkan dan hanya berdasarkan pada memori. *KNN* bisa digunakan untuk memasukkan data baru (*data testing*) ke dalam kelompok data yang jaraknya berdekatan dengan *data training*, sehingga metode ini bisa digunakan untuk mengklasifikasi *data testing* sesuai dengan kelompok data yang seharusnya. Prinsip dari *KNN* adalah menemukan *k* objek dari *data training* yang paling dekat dengan *data testing*. Algoritma *KNN* sangat sederhana, bekerja berdasarkan pada jarak terdekat dari *data testing* dengan *data training* untuk menentukan *k*-tetangga

terdekat (*KNN*), kemudian diambil mayoritas dari *KNN* untuk dijadikan klasifikasi dari *data testing* (M.A. et al., 2020).

Pada proses pengklasifikasian, algoritma ini tidak menggunakan model apapun untuk dicocokkan dan hanya berdasarkan pada memori. Algoritma *KNN* menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari sampel uji yang baru. Menurut J. Sreemathy & P.S Balamurugan (2012), peringkat untuk *k* tetangga terdekat berdasarkan nilai kesamaan dihitung menggunakan jarak *Euclidean* (*Eucledean Distance*) yang didefinisikan sebagai berikut:

$$D\ (X,Y) = \sqrt{\sum_{i=1}^{p} (Xi - Yi)^2}$$

Dengan

$D\ (X,Y)$	:	Jarak *Euclidean* (*Euclidean Distance*)
Xi	:	*Sample* data
Yi	:	Data uji
p	:	Dimensi data
i	:	Variable data

Menurut Karegowda et al. (2012) Algoritma *K-Nearest Neighbor* dapat dituliskan sebagai berikut :

1. Tentukan : *k*, tetangga terdekat
2. Hitung jarak data baru (*input*) dengan data *training*, Ukuran jarak yang digunakan adalah jarak *Euclidean*:

$$D\ (X,Y) = \sqrt{\sum_{i=1}^{p} (Xi - Yi)^2}$$

3. Urutkan jarak dari yang terdekat
4. Periksa kelas *k* tetangga terdekat
5. Kelas data baru = kelas mayoritas tetangga terdekat.

2.1.7 *Performance Measurement* (Klasifikasi)

Pengukuran performa atau keakuratan hasil klasifikasi dilakukan untuk melihat hasil yang didapatkan dari klasifikasi yaitu dengan menggunakan

Confusion Matrix. *Confusion Matrix* mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan oleh sistem dengan hasil klasifikasi yang seharusnya (Imron & Prasetiyo, 2020).

Tabel 2. 1 Tabel Confusion Matrix (Asiyah, 2016)

	Prediksi		
Aktual		**Kelas Kritikal**	**Kelas Non Kritikal**
	Kelas Kritikal	F_5	F_6
	Kelas Non Kritikal	F_7	F_8

Keterangan:

F5 = TP (Aktual: Kritikal, Prediksi: Kritikal)

F6 = FN (Aktual: Kritikal, Prediksi: Non Kritikal)

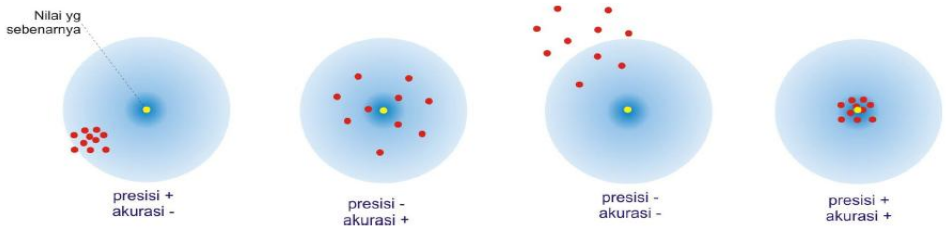
F7 = FP (Aktual: Non Kritikal, Prediksi: Kritikal)

F8 = TN (Aktual: Non Kritikal, Prediksi: Non Kritikal)

Merangkum prediksi dan label dengan *True Positive (TP)*, *False Positive (FP)*, *True Negative (TN)*, dan *False Negative (FN)*. Versi ternormalisasi (per baris atau kelas aktual) memudahkan membaca proporsi benar atau salah per kelas. Terdapat beberapa cara untuk mengukur performa dengan menggunakan *Confusion Matrix* yaitu:

1. Akurasi (*Accuracy*)

Akurasi menggambarkan seberapa akurat model dapat mengklasifikasikan antara nilai prediksi hasil klasifikasi dengan nilai yang sebenarnya.



Gambar 2. 3 *Confusion Matrix – Accuracy* (Asiyah, 2016)

$$accuracy = \frac{F_5+F_8}{F_5+F_6+F_7+F_8}$$

2. Recall

Recall menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Dari tabel 2.1 ukuran performansi *recall* untuk kelas 1 dapat dihitung menggunakan persamaan sebagai berikut,

$$recall = \frac{F_{aa}}{\sum_{b=1}^{B} F_{ab}}$$

Dimana

a = 1, 2, …., A
b = 1, 2, …., B

$$recall\ kategori\ 1 = \frac{F_5}{F_5 + F_6}$$

Persamaan tersebut dapat diartikan proporsi ketepatan klasifikasi data aktual yang berasal dari data prediksi kelas 1. Perhitungan untuk *recall* pada kelas 2 disesuaikan dengan Tabel 2.1.

3. Precision

Precision menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Berikut merupakan persamaan untuk kelas 1,

$$precision = \frac{F_{aa}}{\sum_{a=1}^{A} F_{ab}}$$

$$precision\ kategori\ 1 = \frac{F_5}{F_5 + F_7}$$

Persamaan tersebut dapat diartikan proporsi ketepatan klasifikasi data yang diprediksi pada kelas 1 yang berasal dari kelas data aktual pada seluruh kelas. Sama seperti *recall*, apabila ingin mencari kelas 2 maka disesuaikan dengan Tabel 2.1.

4. *F1-Score* (*F-measure*)

F1-Score merupakan kompromi dari *recall* dan *precision* untuk mengukur kinerja keseluruhan pengklasifikasi. Berikut merupakan cara perhitungan *F1-Score*.

$$F1 - Score = \frac{2\ x\ recall\ x\ precision}{recall + precision}$$

5. *Area Under the ROC Curve (AUC-ROC)*

AUC (*Area Under the ROC Curve*) merupakan metrik evaluasi yang digunakan untuk mengukur kinerja model klasifikasi, termasuk *K-Nearest Neighbors* (*KNN*), dengan menilai kemampuan model dalam membedakan antara kelas positif dan negatif. *ROC Curve* (*Receiver Operating Characteristic*) memvisualisasikan *trade-off* antara *True Positive Rate* (*TPR*) dan *False Positive Rate* (*FPR*) pada berbagai *threshold* klasifikasi. *AUC* mengkuantifikasi keseluruhan performa model dalam bentuk nilai antara 0.5 hingga 1.0, di mana 0.5 menunjukkan performa acak (tidak lebih baik daripada tebakan *random*) dan 1.0 menunjukkan klasifikasi sempurna. Keunggulan *AUC* terletak pada ketahanannya terhadap ketidakseimbangan kelas (López et al., 2013).

a. *True Positive Rate* (*TPR/Recall/Sensitivity*), proporsi *instance* positif yang benar diklasifikasikan

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

b. *False Positive Rate* (*FPR*), proporsi instance negatif yang salah diklasifikasikan sebagai positif.

$$FPR = \frac{FP}{FP + TN}$$

c. *ROC Curve*, memplot *TPR* (sumbu Y) dan *FPR* (sumbu X) untuk berbagai *threshold* klasifikasi

d. *AUC* dihitung dengan mengintegrasikan area di bawah kurva *ROC*,

$$AUC = \int_0^1 TPR(FPR)\, d(FPR)$$

6. Precision-Recall (PR) Curve dan Average Precision (AUC-PR/AP)

PR Curve menyorot trade off *Precision-Recall* untuk kelas positif, *Average Precision* luas di bawah *PR curve* lebih sensitif terhadap perubahan performa kelas minoritas dibanding *ROC* pada skenario *imbalance*. (Saito & Rehmsmeier, 2015)

7. Pemilihan Skor untuk Tuning dan Pelaporan

Pada *tuning hyperparameter* dilakukan dalam Stratified K-Fold CV (anti *leakage*). Skor *CV* dipilih selaras dengan tujuan ketika prioritasnya meminimalkan *false negative* pada kelas 1, gunakan *F1* atau *AP* sebagai skor utama yang lebih

cocok daripada *Accuracy* pada *imbalance*. Model akhir hanya dievaluasi sekali di set uji terpisah dengan metriknya yaitu seperti *Accuracy*, *Precision, Recall, F1, AUC-ROC,* dan *AP* serta *Confusion Matrix* (Tharwat, 2018).

2.1.8 *Dashboard*

Dashboard merupakan alat untuk menyajikan informasi secara singkat, memberikan tampilan antarmuka dengan berbagai bentuk seperti *diagram*, laporan, indikator visual, mekanisme alert yang dipadukan dengan informasi yang dinamis dan relevan, dan memberikan solusi bagi kebutuhan informasi organisasi dan sehingga kinerja organisasi dapat di-monitor secara sekilas .

Dengan kata lain, *dashboard* menampilkan data ke dalam bentuk yang berbeda sehingga mempermudah memahami data yang dimiliki. Dengan adanya *dashboard* dapat memberikan beberapa manfaat kepada perusahaan. Berikut adalah manfaat yang diberikan *dashboard* (Shalabh Aggarwal, 2014),

Dashboard dapat meningkatkan pengambilan keputusan dengan memperkuat pemahaman dan memanfaatkan kemampuan persepsi manusia. *Dashboard* dapat mengumpulkan, meringkas, dan mempresentasikan informasi dari berbagai sumber agar pengguna dapat melihat langsung bagaimana performa dari suatu data.

2.1.9 *Perangkat Lunak yang Digunakan*

2.1.9.1 Google Colab (Google Collaboratory)

Google Colab adalah aplikasi berbasis *cloud* yang digunakan untuk pengembangan *machine learning* dan *artificial intelligence* (*AI*). Aplikasi ini memiliki fitur *built-in-library machine learning* yang lengkap, seperti Keras, *PyTorch*, dan *TensorFlow* sehingga memudahkan pengguna untuk mengimpor dan meng-*install* berbagai pustaka *Python* populer dengan mudah. Selain itu, *Google Colab* juga dapat diakses melalui *smartphone* karena bersifat *cloud*. *Google Colab* juga dapat diintegrasikan dengan *Jupyter Notebook* di komputer dengan *local runtime*, serta dihubungkan dengan *Google Drive* atau *Github*. Kelebihan lainnya adalah dapat melakukan running program *deep learning* melalui *smartphone*, karena *Google Colab* bisa dibuka di *browser*. *Google Colab* juga memberikan akses gratis ke infrastruktur komputasi seperti penyimpanan, memori, kapasitas

pemrosesan, *Graphics Processing Unit (GPU)*, dan *Tensor Processing Unit (TPU)*. Dengan fitur-fitur tersebut, *Google Colab* menjadi alat yang populer di kalangan peneliti dan praktisi *machine learning* (Naik et al., 2023).

2.1.9.2 Bahasa Pemrograman Python

Bahasa pemrograman *Python* merupakan salah satu bahasa pemrograman yang banyak digunakan oleh perusahaaan besar maupun para *developer* untuk mengembangkan berbagai macam aplikasi berbasis *desktop, web* dan *mobile* (Copperwaite & Leifer, 2015). *Python* juga memiliki *library* yang lengkap sehingga memungkinkan *programmer* untuk membuat aplikasi yang mutakhir dengan menggunakan *source code* yang tampak sederhana. *Python* dapat digunakan pada berbagai sistem operasi seperti *Linux, Microsoft Windows, Mac OS, Android, Symbian OS, Amiga, Palm,* dan lain-lain. Python juga mendukung multi *platform* dan multi *system* serta memiliki sistem pengelolaan memori otomatis. *Python* banyak digunakan dalam pengembangan *machine learning* karena mudah dipelajari dan memiliki *library* yang lengkap (Naik et al., 2023).

2.2 TINJAUAN PENELITIAN TERDAHULU

Berikut ini mengenai penelitian terkait yang mencoba untuk menyelesaikan permasalahan dalam penelitian yang dilakukan antara lain:

Masalah utama pada penelitian Yang & Shami (2020) adalah implementasi penentuan hiperparameter yang sering *ad-hoc* yang menghasilkan model jadi tidak stabil dan sulit direplikasi sehingga pada penelitian ini ditekankan untuk memilih skema pencarian (*grid vs random vs Bayesian*) dan desain validasi silang yang selaras dengan tujuan metrik. Penelitian ini bertujuan untuk menyintesis teknik *Hyperparameter Optimization* (*HPO*) *modern* dan memberi pedoman kapan tiap pendekatan tepat digunakan dengan meliputi *Grid*/*Randomized Search, Bayesian optimization, meta-heuristics* serta skema *CV* (*k-fold, repeated CV*) dan strategi *scoring*. Dan diperoleh hasil utamanya:

(i) pemilihan metrik harus sesuai tujuan bisnis;

(ii) *random*/*Bayesian* sering lebih efisien daripada *grid* murni pada ruang hiperparameter luas;
(iii) replikasi menuntut penetapan *seed*, skema *CV* konsisten, dan pelaporan ruang pencarian.

Pada penelitian Luque et al (2019) mengenai akurasi dan metrik berbasis *confusion matrix* mudah menyesatkan pada data tidak seimbang yang dimana banyak penelitian sebelumnya yang melaporkan akurasi atau *ROC* saja tanpa membahas *PR-AUC* atau *average precision*. Sehingga penelitian ini menganalisis bagaimana ketidakseimbangan memengaruhi interpretasi metrik dan kapan metrik tertentu layak digunakan. Kajian teoretis dan empiris atas metrik (*precision*, *recall*, *F1*, *ROC-AUC*, *PR-AUC*) di berbagai tingkat *imbalance*. Hasil penelitian menemukan bahwa *PR-AUC* atau *average precision* dan *F1* lebih informatif untuk kelas minoritas di mana akurasi dan bahkan *ROC-AUC* bisa terlihat baik padahal performa di kelas minoritas buruk. Selain menekankan pentingnya *PR-AUC*/*F1* pada data tak seimbang, penelitian ini juga menyoroti ketergantungan *precision* pada prevalensi (proporsi kelas positif) yang artinya dua model dengan kualitas deteksi serupa bisa tampak berbeda hanya karena perubahan distribusi kelas sehingga peneliti melaporkan komposisi kelas dan atau *average precision* bersama kurva *PR*. Peneliti juga membahas *macro* atau *weighted averaging* pada skenario multi-kelas serta risiko *over-optimism* saat menambal *threshold* bila evaluasi tidak dilakukan di lintas-*fold* yang sama.

Pada penelitian Kaur et al (2019), ketidakseimbangan kelas memicu bias pembelajaran dan penurunan generalisasi di mana strategi penanganan sering hanya satu sisi, misal hanya melakukan *oversampling* tanpa menimbang interaksi dengan pemilihan model dan validasi. Sehingga penelitian ini meninjau sampling (*SMOTE* dan turunannya), penyesuaian *loss*, *ensemble*, dan evaluasi dengan tinjauan sistematis lintas domain. Dan diperoleh hasil penelitian:

(i) *SMOTE baseline* yang kuat, namun parameter seperti *k_neighbors* dan strategi *sampling* perlu disesuaikan ukuran minoritas;
(ii) teknik evaluasi harus menyorot kelas minoritas;
(iii) *pipeline* harus mencegah *leakage*, *SMOTE* diterapkan di dalam *CV*.

Pada penelitian Anowar et al (2021), dimensi tinggi dan korelasi antarfitur melemahkan model berbasis jarak seperti *kNN* (*curse of dimensionality*) di mana implementasi memilih target varian *PCA* sering arbitrer. Sehingga penelitian ini memberi konseptualisasi dan komparasi reduksi dimensi serta panduan pemilihan dengan survei komprehensif *PCA* (linier) dan *t-SNE*/*UMAP* (nonlinier) pada berbagai tugas. Dan hasil penelitian ini, *PCA* efektif mengontrol korelasi atau *noise* dan sering cukup untuk *pipeline* klasifikasi tabular dengan penentuan komponen berbasis varian kumulatif merupakan pendekatan yang dilakukan untuk menyeimbangkan bias *variance*.

Pada penelitan Krawczyk et al (2024), SMOTE klasik kadang memperhalus batas kelas secara berlebihan (*over-generalization*) atau menghasilkan sampel sintetis di wilayah tumpang tindih di mana implmentasi pemilihan varian *SMOTE* dan setelan *k_neighbors* sering tidak dibahas rinci. Sehingga penelitian merangkum perkembangan *SMOTE* (*Borderline-SMOTE, SMOTE-Tomek*, *ADASYN*, dsb.) dan panduan imeplementasi penggunaannya. *Survey* mendalam dengan model *oversampling* sintetis dan evaluasinya. Dan diperoleh hasil penelitian:

(i) *k_neighbors* kecil dapat lebih aman pada minoritas sangat kecil;
(ii) integrasi *oversampling* dengan validasi silang wajib untuk mencegah *leakage*;
(iii) varian *borderline* dapat dicoba saat kelas tumpang tindih.

Di luar rekomendasi *k_neighbors* kecil untuk minoritas sangat kecil, penelitian ini menekankan protokol evaluasi yang bebas kebocoran (*leakage*): *SMOTE* harus diterapkan di dalam tiap *fold CV* bukan sebelum *split*. Peneliti juga membedakan kapan *Borderline-SMOTE* lebih tepat (batas kelas tumpang-tindih atau rapuh) dibanding *ADASYN* (agresif di area sulit dan berpotensi menambah *noise*). Kombinasi *SMOTE* dan *cleaning* dapat merapikan tumpang-tindih setelah *oversampling*. Untuk data berdimensi menengah-tinggi, peneliti mengingatkan bahwa *oversampling* di ruang sangat berdimensi dapat memperburuk *curse of dimensionality*. Peneliti juga menyarankan membedakan *k SMOTE vs k KNN* (*k_SMOTE* biasanya lebih kecil daripada *k_KNN*) dan memantau kelas *borderline* lewat inspeksi kedekatan tetangga *pasca-SMOTE*.

Pada penelitan Hancock & Khoshgoftaar (2023), masalah evaluasi model pada ketidakseimbangan ekstrem di konteks data *modern* (*big data*), di mana *ROC-AUC* sering mendominasi pelaporan sementara ketepatan di kelas minoritas tidak tercermin memadai. Penelitian ini mengatasi kurangnya implementasi standar pelaporan kurva *PR*, *AP*, dan laporan multi-*threshold* untuk memetakan *trade-off precision–recall* sesuai kebijakan risiko. Penelitian ini bertujuan menyintesis kerangka evaluasi praktis pada skenario *imbalance* tinggi, termasuk rekomendasi pelaporan multi-metrik, kurva *PR*, dan pemilihan *threshold* berbasis tujuan (mis. *Fβ*) dengan menggunakan model panduan evaluasi dan contoh perhitungan komparatif, bukan algoritme klasifikasi baru. Dan hasil yang ditekankan adalah *PR-AUC/F1* lebih representatif untuk minoritas, keputusan operasional sebaiknya disertai evaluasi di beberapa ambang dan atau *tuning threshold* di dalam *CV*.

Tabel 2. 2 Penelitian Terdahulu

Penulis	Model/Metode yang Diulas/Digunakan	Keterkaitan dengan Penelitian Saat Ini	Keterbatasan yang Dicatat
Yang & Shami (2020)	Analisis evaluasi metrik (*accuracy*, *precision*, *recall*, *F1*, *ROC-AUC*, *PR-AUC*/*AP*), pengaruh prevalensi, isu *averaging* & *threshold*.	Menguatkan penggunaan *GridSearchCV* 5-*fold* ber-*F1* serta pelaporan ruang pencarian dan *seed*	Tidak spesifik ke kasus *kNN* suku cadang; rekomendasi implementatif perlu disesuaikan *resource*
Luque et al. (2019)	Analisis metrik (*Accuracy, Precision, Recall, F1, ROC-AUC, PR-AUC*)	Menguatkan pemilihan *F1* dan *AP* sebagai metrik utama/pelengkap; anjuran melaporkan rasio kelas; kesadaran *tuning threshold* di dalam *CV* yang selaras dengan eksperimen dan *dashboard*.	Fokus pada kerangka evaluasi, bukan perbandingan model atau domain spesifik; tidak menguji *pipeline end-to-end* seperti *KNN* dan *PCA* dan *SMOTE*.
Kaur et al. (2019)	*SMOTE* dan varian, *cost-sensitive*, evaluasi	Mendukung pemakaian *SMOTE* (di dalam CV, *anti-leakage*) dan pemilihan metrik berbasis minoritas	Survei luas; tidak memberi parameter spesifik untuk tiap domain
Anowar et al. (2021)	*PCA*, *t-SNE*, *UMAP*; kriteria pemilihan komponen	Landasan pemilihan target varian *PCA* (80–95%) untuk menjaga bias-variance *kNN*	Dominan konseptual; tidak menyajikan angka baku terbaik untuk semua dataset
Krawczyk et al. (2024)	Review SMOTE & variannya; rekomendasi *k_neighbors* kecil untuk minoritas sangat kecil; integrasi *oversampling* di *CV*; opsi varian *borderline*.	Menjustifikasi setelan *SMOTE* *k*=1/3 dan integrasi di Stratified 5-fold *CV* pada *pipeline*; sejalan dengan fokus *F1/AP*.	Kajian luas, namun tidak mengevaluasi dataset; beberapa rekomendasi bersifat kontekstual (perlu validasi empiris per domain).
Hancock, J. T.; Khoshgoftaar, T. M. (2023).	Panduan evaluasi untuk data sangat tidak seimbang: kurva *PR*, *AP*, pelaporan multi-*threshold*, dan pemilihan *Fβ* sesuai kebijakan.	Relevan dengan pelaporan *PR-AUC*/*AP* dan opsi *thresholding* di penelitian (menyajikan skor pada beberapa ambang, dan bila dioptimasi, di dalam *CV*).	Artikel tinjauan atau panduan; tidak menawarkan algoritma baru.