# BAB II
# LANDASAN TEORI

## **2.1 Tinjauan** Pustaka

Beberapa penelitian terkait dengan penggunaan *machine learning* dan analisis data pada LMS/ALMS adalah sebagai berikut; Penelitian yang dilakukan oleh Rawat dan Malhan (2019), memfokuskan untuk membandingkan model klasifikasi *hybrid machine learning* berdasarkan algoritma *decision tree, clustering, artificial neural network, naive bayes* dan lain-lain. Model klasifikasi *hybrid* digunakan untuk menganalisis kinerja siswa dengan temuan bahwa metode klasifikasi *hybrid* lebih efisien untuk prediksi kinerja siswa berdasarkan data terkait dengan akurasi sebesar 98% (Rawat dan Malhan, 2019).

Penelitian yang dilakukan oleh Gomathy Ramaswami dkk (2019), melakukan penelitian menggunakan teknik EDM untuk meningkatkan akurasi pada prediksi kinerja akademik siswa. Algoritma *machine learning* yang digunakan adalah *naive bayes* (NB), *logistic regression* (LR), *k-nearest neighbor* (KNN) dan *random forest* (RF), digunakan untuk menghasilkan model prediktif. Data yang digunakan dari aktivitas penggunaan *Xorro-Q* untuk mengevaluasi efektivitas metode EDM dalam memprediksi kinerja akademik siswa dengan berfokus pada siswa yang beresiko gagal dalam kursus, dari empat algoritma yang diterapkan menemukan bahwa RF lebih akurat daripada tiga algoritma lainnya (Ramaswami dkk, 2019).

Penelitian yang dilakukan oleh A. Ahmed dan M. Malik (2020), menganalisis *output* yang dihasilkan menggunakan algoritma *machine learning* yang dapat membantu dalam prediksi apakah kebijakan pembelajaran dalam *e-learning* merugikan atau tidak selama COVID-19. Penelitian ini menyelidiki kinerja algoritma *machine learning* untuk pengambilan keputusan strategis di lembaga pendidikan tinggi. Hasil penelitian menunjukkan bahwa kinerja algoritma *Random Forest* (RF) lebih tinggi dibandingkan dengan SVM, DT dan *Naive Bayes* dengan akurasi sebesar 99% (Ahmed dan Malik, 2020).

Penggunaan *machine learning* dan *data mining* pada lingkungan pendidikan untuk memprediksi kinerja siswa dikenal dengan *educational data mining* (EDM), yang menjadi penelitian yang penting. Memprediksi awal kinerja siswa dapat membantu entitas terkait untuk memberikan solusi pada kinerja siswa yang rendah. Penelitian yang dilakukan oleh Alamri dkk (2020), memprediksi kinerja akademik siswa untuk meningkat hasil akademik yang lebih baik. Penelitian ini menggunakan algoritma dan teknik klasifikasi *Support vector machine* (SVM) dan *Random Forest* (RF). *Dataset* terdiri dari 369 dari pelajaran matematika dan 649 dari pelajaran bahasa portugis. Hasilnya percobaan untuk algoritma SVM dan RF yang diterapkan pada kedua *dataset* menunjukkan bahwa akurasi dalam kasus klasifikasi biner mencapai prediksi akurat mencapai 93%, sedangkan dalam regresi, RMSE terendah adalah 1,13 dalam kasus RF (Alamri dkk, 2020).

Muhammad Adnan (2021), mengusulkan model prediktif yang dapat menganalisis masalah yang dihadapi oleh siswa selama menggunakan *platform* pembelajaran *online.* Model prediktif tersebut dapat membantu instruktur mengidentifikasi siswa yang beresiko di awal untuk dapat menghindari kasus putus sekolah. Menggunakan algoritma *machine learning* (ML) dan *deep learning* (DL) untuk mengklasifikasi perilaku belajar siswa sesuai dengan variabel belajarnya. Selanjutnya, performa dari berbagai algoritma ML tersebut dibandingkan dengan menggunakan akurasi, presisi, *recall* dan *f1 score*. Hasil menunjukkan bahwa nilai siswa, waktu menggunakan *platform* dan banyaknya keterlibatan siswa merupakan faktor penting dalam pembelajaran *online.* Hasil penelitian menunjukkan bahwa model prediksi yang dilatih menggunakan *random forest* (RF) memberikan hasil terbaik dengan rata-rata *precision* 0,92%, *recall* 0,91%, *f1-score* = 0,91% dan akurasi sebesar 80% (Adnan dkk, 2021).

Shu-tong Xie dkk (2021), mengumpulkan data *log* perilaku siswa selama proses pengajaran di masa pandemi, untuk melakukan analisis perilaku belajar dan hasil prediksi belajar. Algoritma *k-means* digunakan untuk analisis *cluster* dan *multiclass classification*, yang menggabungkan pemilihan ciri berdasarkan *genetic algorithm* (GA) dengan metode *error correcting output code* (ECOC). Penelitian

ini mengklasifikasikan dalam empat label yaitu *excellent, good, pass,* dan *fail,* yang dapat digunakan untuk membedakan tingkat pembelajaran siswa dan personalisasi pengajaran. Hasil penelitian menunjukkan bahwa metode *multiclass classification* yang diusulkan dapat secara efektif memprediksi tingkat kinerja, dengan tingkat akurasi rata-rata lebih dari 75% (Xie dkk, 2021).

Randhir Singh dan Saurabh Pal (2020), mengimplementasikan empat algoritma *machine learning : Decision Tree (DT), Naive Bayes (NB), KNN* dan *Extra Tree (ET)* kemudian membangun model untuk mengkombinasikan hasil dari tiap-tiap algoritma menggunakan *ensemble technique* yaitu *bagging* dan *boosting* untuk meningkatkan performa dari algoritma *machine learning*. Nilai akurasi terbaik diantara algoritma klasifikasi *machine learning* sebesar 86,83% dari *Naive Bayesian* dan 91,76% dari *ensemble technique boosting* (Singh dan Pal, 2020).

Jing Yu (2021), melakukan penelitian tentang model prediksi kinerja akademik pada pembelajaran *online* menggunakan algoritma *random forest* dan metode *artificial intelligence.* Penelitian tersebut menggunakan data pembelajaran *online* dari *platform online course*, yang didalamnya terdapat *records* dari penggunaan *online course* dengan 12 atribut. Hasil dari penelitian ini menunjukkan bahwa algoritma RF memiliki nilai akurasi lebih dari 90% dibandingkan dengan algoritma *decision tree* (Yu, 2021b).

Eluwumi Buraimoh dkk (2021), melakukan penelitian enam model *machine learning* untuk memprediksi keberhasilan siswa dalam lingkungan *e-learning*. Studi ini mencoba memberikan model dengan akurasi yang optimal untuk menentukan siswa yang memerlukan bantuan atau tidak dalam meningkatkan kinerja belajar. Model *decision tree* dan *linear regression* memiliki skor akurasi terbaik sebesar 0,86 setelah dilakukan *cross-validation*. Atribut perilaku siswa merupakan prediktor yang berguna untuk keberhasilan siswa (Buraimoh dkk, 2021).

Suad Almutairi dkk (2019), menganalisis hubungan antara kinerja siswa pada *e-learning* menggunakan *data mining*. Faktor utama yang mempengaruhi kinerja

siswa dipilih dengan menggunakan metode seleksi ciri. Metode *machine learning* yang digunakan dan diuji untuk memprediksi kinerja siswa yaitu *random forest, logistic regression,* XGBoost, MLP dan *ensemble learning* menggunakan *bagging* dan *voting*. Dari semua metode tersebut *random forest* mendapatkan akurasi tertinggi mencapai 77%. Hasilnya menunjukkan bahwa *data mining* dapat secara akurat memprediksi tingkat kinerja siswa (Almutairi dkk, 2019). Beberapa penelitian yang menggunakan *machine learning* dan *ensemble* l*earning* pada data aktivitas LMS dapat dilihat pada tabel 2.1.

Tabel 2.1 Penelitian terkait

| **Penulis** | **Masalah Penelitian** | **Dataset** | **Pengukuran** | **Hasil** |
|---|---|---|---|---|
| Ramaswami dkk (2019) | Melakukan penelitian menggunakan teknik EDM untuk meningkatkan akurasi pada prediksi kinerja siswa. | *Student engagement dataset* yang dikumpulkan secara manual. | Akurasi pada prediksi kinerja siswa dengan algoritma ML : NB, LR, KNN dan RF. | Dari empat algoritma yang diterapkan menemukan bahwa, RF lebih akurat dari pada tiga algoritma lainnya. |
| Alamri dkk (2020) | Memprediksi kinerja akademik siswa untuk meningkatkan hasil akademik yang lebih baik. | *Dataset* terdiri dari 369 dari pelajaran matematika dan 649 dari pelajaran bahasa portugis. | Akurasi untuk prediksi kinerja siswa pada algoritma SVM dan RF. | Algoritma SVM dan RF untuk klasifikasi biner mencapai prediksi akurat mencapai 93%. Regresi, RMSE terendah adalah 1,13 dalam kasus RF. |
| Adnan dkk (2020) | Mengklasifikasi perilaku belajar siswa sesuai dengan variabel belajarnya. | *Open University Learning Analytics Dataset* (OULAD) | Akurasi pada model ML dan DL. | *Random forest* (RF) memberikan hasil terbaik dengan rata-rata precision 0.92%, *recall* 0.91%, *f1-score* = 0.91% dan akurasi sebesar 80%. |

| Penulis | Masalah Penelitian | Dataset | Pengukuran | Hasil |
|---|---|---|---|---|
| Rawat dan Malhan (2019) | Model klasifikasi *hybrid* digunakan untuk menganalisis kinerja siswa. | *Student dataset of department computer science* | Akurasi pada model klasifikasi *hybrid* ML berdasarkan algoritma DT, CNN dan NB. | Metode klasifikasi *hybrid* lebih efisien dengan akurasi sebesar 98%. |
| Randhir Singh dan Saurabh Pal (2020) | Implementasi empat algoritma ML dan *ensemble technique* untuk meningkatkan performa dari algoritma ML. | *Student dataset of bachelor of computer application program* | Akurasi prediksi pada 4 algoritma ML dan teknik *ensemble learning (bagging* dan *boosting.* | Nilai akurasi terbaik diantara algoritma klasifikasi ML sebesar 86.83% dari NB dan 91.76% dari *ensemble technique boosting.* |
| Jing Yu (2021) | Penelitian tentang model prediksi kinerja akademik pada pembelajaran *online* menggunakan algoritma RF dan metode AI. | Data pembelajaran *online* dari *platform online course.* | Memverifikasi penggunaan algoritma ML pada data pembelajaran *online.* | Algoritma RF memiliki nilai akurasi lebih dari 90% dibandingkan dengan algoritma DT. |
| Eluwumi Buraimoh dkk (2021) | Penelitian enam model ML untuk memprediksi keberhasilan siswa dalam lingkungan *e-learning.* | *Students performance dataset* | Akurasi yang optimal untuk menentukan siswa yang memerlukan bantuan atau tidak dalam meningkatkan kinerja belajar. | Model DT dan LR memiliki skor akurasi terbaik sebesar 0,86 setelah dilakukan *cross-validation.* |
| Suad Almutairi dkk (2019) | Menganalisis hubungan antara kinerja siswa pada *e-learning* menggunakan *data mining.* | *Students performance dataset* | Prediksi kinerja siswa menggunakan algoritma ML dan *ensemble.* | RF mendapatkan akurasi tertinggi mencapai 77%. |

Berdasarkan referensi tersebut fokus penelitian yang akan dilakukan adalah membangun model prediksi kinerja siswa menggunakan teknik *ensemble machine*

*learning* yaitu *bagging, boosting* dan *voting* berdasarkan data aktivitas siswa selama menggunakan ALMS, kemudian untuk menganalisis hubungan kinerja siswa dan perilaku belajar dalam menggunakan ALMS menggunakan analisis korelasi *pearson.*

## 2.2 Dasar Teori

### 2.2.1 Kecerdasan Buatan (*Artificial Intelligence)*

Kecerdasan buatan atau *Artificial Intelligence* (AI) membahas bagaimana sebuah komputer dapat meniru cara berpikir seperti manusia. Komputer dapat mengambil kesimpulan dan memutuskan suatu permasalahan seperti layaknya manusia. *Artificial Intelligence* merupakan studi komprehensif yang mengintegrasikan pengetahuan dari banyak studi. Melibatkan ilmu komputer, psikologi, matematika, statistik, linguistik, dan sebagainya. *Artificial Intelligence* mengacu pada mesin cerdas yang dapat membuat penalaran nyata dan memecahkan masalah secara mandiri (Yu, 2021a). *Artificial Intelligence* sebagai suatu sistem komputer yang dirancang untuk dapat berinteraksi dengan dunia melalui kemampuan-kemampuan tertentu dan perilaku intelijen yang seperti manusia pada umumnya. *Artificial Intelligence* juga sering dianggap sebagai cara komputer untuk melakukan serangkaian tes berpikir yang dimiliki manusia dan hewan (Pratikno, 2018). Menurut Russel dan Norvig pengertian AI dapat dibedakan menjadi 4 kategori, yaitu (Russell dan Norvig, 2021) :

1. *Thinking Humanly*

   *Thinking Humanly* dilakukan dengan dua cara yaitu; pertama melalui introspeksi, mencoba menangkap pemikiran sendiri saat berpikir. Kedua melalui penelitian-penelitian dari segi psikologi.

2. *Acting Humanly*

   *Acting Humanly* merupakan pendekatan untuk menguji apakah komputer mengelabui seorang manusia/ interogator melalui komunikasi berbasis teks jarak jauh. Komputer tersebut harus memiliki kemampuan, *Natural language Processing, Knowledge Representation, Automated Reasoning, Machine Learning, Computer Vision,* dan *Robotics.*

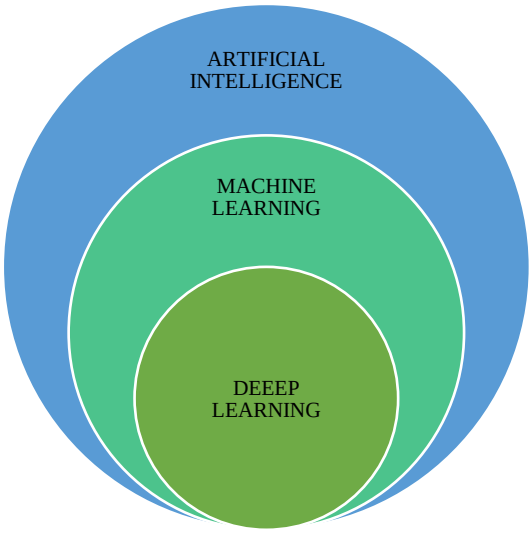
3. *Thinking rationally*

   *Thinking rationally* adalah pendekatan dengan dua masalah yaitu; pertama tidak mudah membuat pengetahuan informal, lalu menyatakan dalam formal t*erm* dengan notasi-notasi logika. Kedua terdapat perbedaan besar antara dapat memecahkan masalah secara prinsip dan memecahkannya dalam dunia nyata.

4. *Acting Rationally*

   *Acting Rationally* membuat intervensi logis yang merupakan bagian dari suatu rational agen. Karena untuk melakukan aksi secara rasional adalah menalar secara logis, sehingga dapat diambil kesimpulan aksi yang dilakukan akan mencapai tujuan atau tidak.

Hubungan antara *artificial intelligence, machine learning dan deep learning* ditunjukkan oleh gambar 2.1. *Artificial intelligence* dapat dikatakan sebagai payung yang lebih luas di mana *machine learning* (ML) dan *deep learning* (DL) berada dalam lingkupnya. ML merupakan bagian dari AI, dan DL bagian dari ML (Cholissodin dan Soebroto, 2020).




Gambar 2.1 Ilustrasi hubungan AI, ML dan DL

*Artificial intelligence* adalah program komputer yang memiliki kemampuan untuk belajar seperti manusia, *machine learning* adalah sebuah algoritma yang

memiliki kemampuan untuk belajar tanpa diprogram secara eksplisit, sedangkan *deep learning* adalah bagian dari ML di mana ANN beradaptasi dan belajar dari sejumlah data besar. Bidang penelitian mengenai AI meliputi *machine learning*, *artificial neural network, expert system*, dan *pattern recognition* tujuan dari penelitian adalah untuk membuat komputer memiliki kemampuan belajar seperti manusia (Turgeon dan Lanovaz, 2020).

*Artificial intelligence* dapat memberikan kemudahan bagi manusia jika dimaksimalkan untuk hal-hal positif. Penelitian tentang penggunaan AI pada dunia pendidikan telah banyak digunakan dengan melibatkan jenjang pendidikan dari sekolah dasar hingga perguruan tinggi. Pemanfaatan AI pada dunia pendidikan antara lain AI sebagai asisten digital dalam menampilkan tutorial pembelajaran, sebagai sistem dalam evaluasi pembelajaran siswa, sebagai sistem layanan komunikasi dengan siswa, dan masih banyak lagi (Pratikno, 2018).

### 2.2.2 Pembelajaran Mesin *(Machine Learning)*

Pembelajaran mesin atau *machine learning* merupakan sub bidang kecerdasan buatan spesialisasi dalam menggunakan data untuk membuat prediksi atau mendukung pengambilan keputusan. Pembelajaran mesin atau *machine learning* adalah disiplin multi-bidang yang muncul pada abad ke-21. Teori pembelajaran mesin untuk merancang dan menganalisis beberapa algoritma yang memungkinkan komputer untuk belajar secara otomatis, *machine learning* mampu dimanfaatkan untuk memprediksi data yang tidak diketahui (Guo dkk, 2021).

*Machine learning* menyatukan semua teknik yang memungkinkan untuk belajar dan membuat prediksi yang akurat dari pengamatan masa lalu. Proses *machine learning* memiliki tiga langkah dasar *initialization*, *learning* dan, *testing* yang menyediakan data keluaran yang efisien dan akurat. *Machine learning* menawarkan cara yang lebih efisien untuk menangkap pengetahuan dalam data untuk meningkatkan kinerja model prediktif secara bertahap dan membuat keputusan berdasarkan data (Raschka dan Mirjalili, 2017). *Machine learning* dapat diterapkan pada banyak area seperti aplikasi identifikasi wajah, prediksi cuaca,

*speech reorganization*, klasifikasi, deteksi penipuan, penyaringan *spam* dan sebagainya (Rawat dan Malhan, 2019).

Secara garis besar *machine learning* mengekstraksi pengetahuan dari data. *Machine learning* merupakan bidang penelitian di persimpangan statistik, AI, dan ilmu komputer yang juga dikenal dengan analitik prediktif atau pembelajaran statistik. Algoritma *machine learning* dibagi menjadi tiga, yaitu algoritma *supervised learning*, *unsupervised learning*, dan *reinforcement learning*. Definisi dari tiga jenis algoritma *machine learning* tersebut adalah (Qazdar dkk, 2019) :

1. Algoritma *Supervised Learning*

   Algoritma *supervised learning* adalah algoritma *machine learning* paling populer. Data-data yang digunakan dalam algoritma ini adalah data berlabel atau klasifikasi yang sesuai. Beberapa algoritma yang termasuk ke dalam algoritma *supervised learning* adalah *Regresi Linier, Random Forest, Super Vector Machine, Naive Bayes, Decision Tree, dan K-Nearest Neighbor* (KNN).

2. Algoritma *Unsupervised Learning*

   Algoritma *unsupervised learning* tidak menggunakan data label karena algoritma ini mampu belajar dari data dengan menemukan pola implisit. Algoritma ini mengidentifikasikan data berdasarkan kepadatan, struktur, segmen serupa, dan ciri serupa lainnya. Salah satu teknik *unsupervised learning* yang paling populer adalah *clustering. Clustering* adalah teknik pengelompokkan kumpulan objek serupa dalam grup yang sama dan berbeda dari objek grup lainnya.

3. Algoritma *Reinforcement Learning*

   *Reinforcement learning* adalah algoritma yang baru untuk *machine learning* dan sangat berbeda dari jenis algoritma *machine learning* sebelumnya. Algoritma *reinforcement learning* memungkinkan mesin untuk berinteraksi dengan lingkungan dinamis untuk mencapai target atau tujuan. Contoh implementasi algoritma *reinforcement learning* adalah *Active Query Answering* (AQA) pada *Google.* Sistem ini akan meneruskan ulang pertanyaan yang diajukan oleh *user google*.

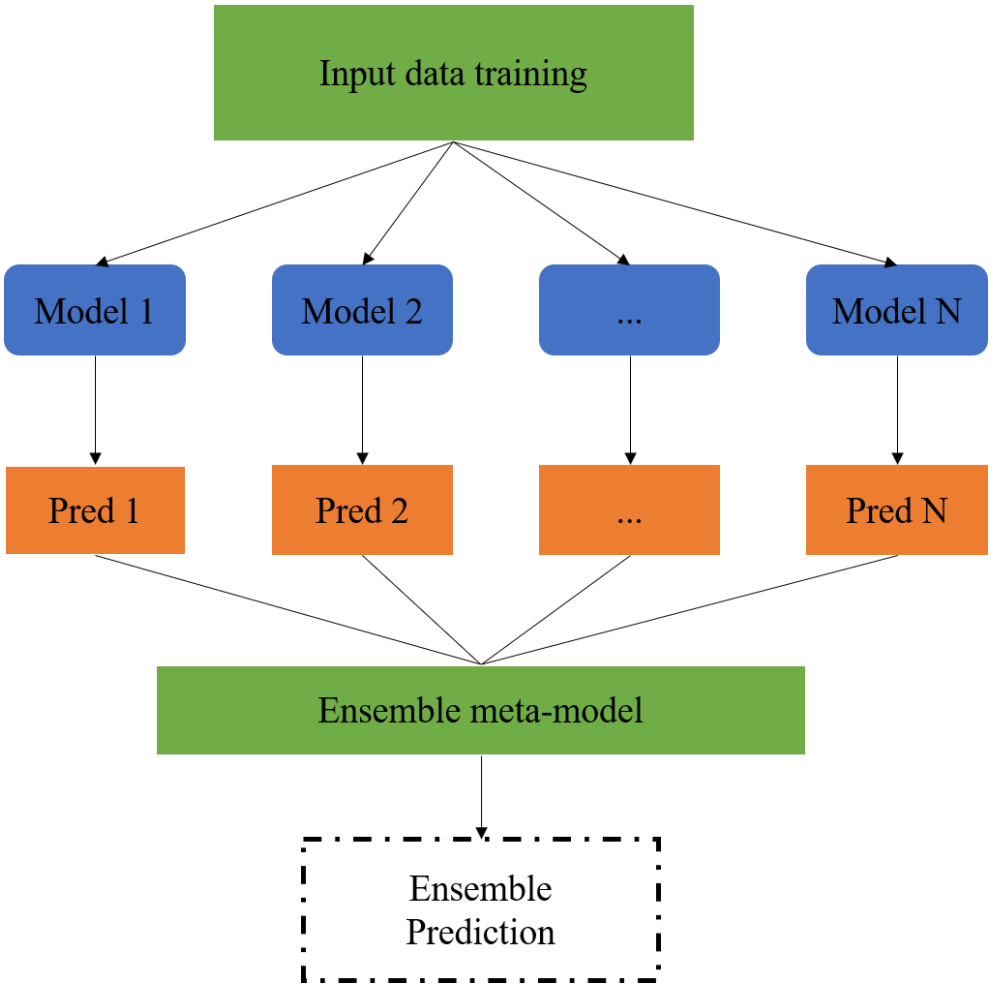
*Data mining* menggunakan statistik untuk mengekstrak informasi dari data mentah. Sedangkan *machine learning* menggunakan teknik-teknik *data mining* dan algoritma pembelajaran lainnya untuk membuat model dari hal-hal yang terjadi pada data untuk memprediksi kejadian selanjutnya (Faisal dan Nugrahadi, 2019). Algoritma *machine learning* yang penulis gunakan pada penelitian ini adalah *algoritma random forest, support vector machine, AdaBoost,* dan *logistic regression.* Algoritma tersebut diimplementasikan pada data aktivitas siswa dalam

menggunakan ALMS dan selanjutnya diterapkan *ensemble* model untuk meningkatkan akurasi.

### *2.2.3 Ensemble Machine Learning*

Mengaplikasikan *machine learning* pada sebuah *project* seringkali memerlukan evaluasi pada model yang dibangun, untuk mengetahui akurasi terbaik sehingga memerlukan banyak percobaan sampai menemukan model yang tepat untuk *project* tersebut. Metode *ensemble* adalah teknik pembelajaran mesin yang menggabungkan beberapa model dasar untuk menghasilkan satu model prediksi yang optimal, karena *ensemble model* menggunakan lebih dari satu model untuk dilatih menggunakan teknik tertentu sehingga mengurangi tingkat kesalahan dan meningkatkan kinerja model. *Ensemble models* dapat meningkatkan akurasi prediksi dibandingkan dengan pengklasifikasian tunggal pada *dataset*. *Ensemble models* dapat diterapkan pada kasus regresi maupun klasifikasi, model ini dapat diterapkan pada algoritma yang sejenis atau dapat memilih algoritma dari jenis yang berbeda. Jika beberapa model dibangun dengan menggunakan *dataset* yang sama dan hanya menggunakan *neural networks*, maka *ensemble model* itu disebut *homogeneous ensemble model.* Sedangkan model yang dibangun menggunakan algoritma yang berbeda seperti *support vector machine, neural network,* dan *random forest* maka, model tersebut dikenal dengan *heterogeneous ensemble model* (Kamal dan Ahuja, 2019).

*Ensemble model* terdiri dari dua langkah, pada langkah pertama harus mempelajari model dasar yang membentuk *ensemble*. Pada langkah kedua harus mencari cara untuk menggabungkan model-model ini (atau prediksinya) menjadi satu model (atau prediksi) yang koheren dengan menggunakan teknik *ensemble* khusus seperti *max-voting, averaging,* dan *weighted averaging.* Struktur *ensembel model* ditunjukkan pada gambar 2.2 (Sarkar dan Natarajan, 2019).




Gambar 2.2 Struktur *ensemble model*

Gambar 2.2 menunjukkan beberapa model akan diterapkan pada data *training* dengan menghasilkan prediksinya masing-masing kemudian hasil prediksi tersebut akan digabungkan menggunakan *ensemble model* yaitu *voting averaging* dan menghasilkan *final prediction*. *Ensemble machine learning* memprediksi label kelas melalui *voting* sederhana, dengan menggabungkan label kelas yang diprediksi dari masing-masing pengklasifikasi individu, $C_j$, dan label kelas, $\hat{y}$, yang menerima suara terbanyak. Model matematika dari *ensemble machine learning* adalah sebagai berikut :

$$\hat{y} = model\{C_1(x), C_2(x), \dots, C_m(x)\} \quad (2.1)$$

Dengan :

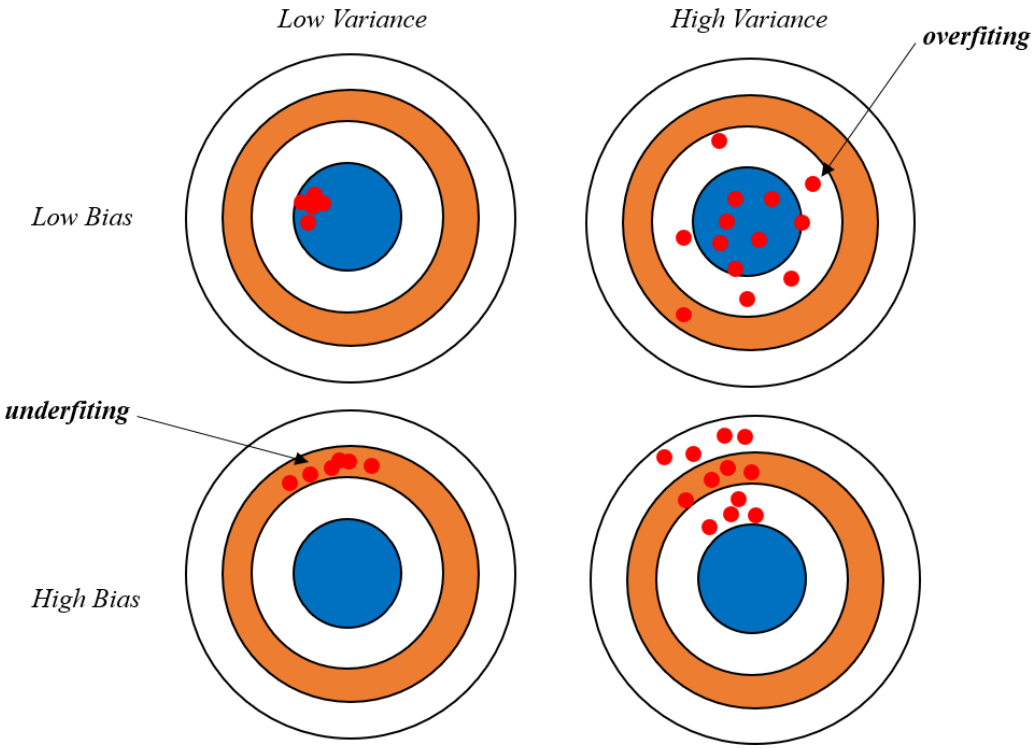
$\hat{y}$ adalah label kelas (target).

$C_1 \ldots . C_m$ adalah pengklasifikasi individu.

Contoh pada, klasifikasi biner di mana class 1 = -1 dan class 2 = +1, dengan menuliskan persamaan prediksi menggunakan *voting* adalah sebagai berikut :

$$C(x) = sign\left[\sum_j^m C_j(x)\right] = \begin{Bmatrix} 1\ if\ \sum_i C_j(x) \geq 0 \\ -1\ lainnya \end{Bmatrix} \qquad (2.2)$$

*Ensemble model* membutuhkan jumlah data yang cukup untuk menghasilkan kinerja dengan baik, karena terbukti lebih berguna ketika diterapkan pada *dataset* yang besar dan *non-linear*. *Ensemble learning* bertujuan untuk memecahkan masalah bias dan varians, dengan menggabungkan banyak model dapat mengurangi kesalahan, dan tetap mempertahankan kompleksitas dari masing-masing model. Model ini menggabungkan model yang berbeda menjadi satu model prediktif untuk mengurangi *variance* menggunakan *bagging,* bias menggunakan *boosting* dan meningkatkan prediksi menggunakan *stacking.* Bias adalah perbedaan antara rata-rata hasil prediksi dari model *machine learning* yang dibangun dengan data nilai yang sebenarnya. Bias yang tinggi dikarenakan dalam pembangunan model ML, dilakukan terlalu sederhana *(oversimplified).* Faktor penyebab lain dikarenakan model ML yang dibangun tidak terlalu berinteraksi dengan *training data*. Sedangkan *variance,* adalah variabel dari prediksi model untuk data tertentu dimana memberikan informasi persebaran data. Model yang memiliki *variance* tinggi sangat memperhatikan hanya pada *training data.* Jika, diberikan data baru yang belum pernah ditemukan training data model tersebut tidak dapat menggeneralisasikan secara baik dari identifikasi data baru tersebut, sehingga model memprediksi dengan keliru. Ilustrasi mengenai bias dan *variance* dijelaskan gambar 2.3.




Gambar 2.3 *Bias* dan *variance*

Gambar 2.3 diilustrasikan data yang memiliki bias yang tinggi *(high bias)* dengan varians yang rendah *(low variance)* akan menjadi *underfitting*. Semetara jika dengan bias yang tinggi *(high bias)* dan varians yang tinggi *(high variance)* menjadikan prediksi sangat tidak tepat. Jika biasnya rendah *(low bias)* dan variansnya tinggi *(high variance)* akan menimbulkan *overfitting* di mana dengan *data train*, performanya baik tapi ketika diberikan data baru, tidak dapat memprediksi. Sehingga yang paling baik adalah jika bias rendah *(low bias)* dan varians rendah *(low variance).* Pada *Python* untuk mengimplementasikan metode *ensemble* menggunakan pustaka *sklearn.ensemble. Sklearn* atau *Scikit-learn* adalah pustaka *machine learning* dari *Python*. Ini menampilkan berbagai macam algoritma seperti *support vector machine*, *random forest*, dan KNN pustaka ini juga mendukung pustaka numerik seperti *NumPy* dan *SciPy*. Adapun kelebihan dan kekurangan pada *ensemble model* adalah sebagai berikut :

Kelebihan

a. *Ensemble model* terbukti dapat meningkatkan akurasi model dan bekerja disebagian besar kasus.
b. Model ini lebih kuat dan stabil sehingga memastikan kinerja yang layak pada uji kasus disebagian besar skenario.
c. Dapat menangkap hubungan kompleks linier dan sederhana serta non-linier dalam data. Hal ini dapat dilakukan dengan menggunakan dua model yang berbeda dan membentuk dua *model ensemble.*
d. Terdapat beberapa teknik *ensemble,* yang bertujuan untuk menggabungkan *weak learners* seperti *bagging, boosting.*

Kekurangan

a. *Ensemble model* memakan waktu dan bukan ide terbaik untuk aplikasi *real time.*
b. Pemilihan model untuk menciptakan *ensemble model* adalah seni yang benar-benar sulit untuk dikuasai.

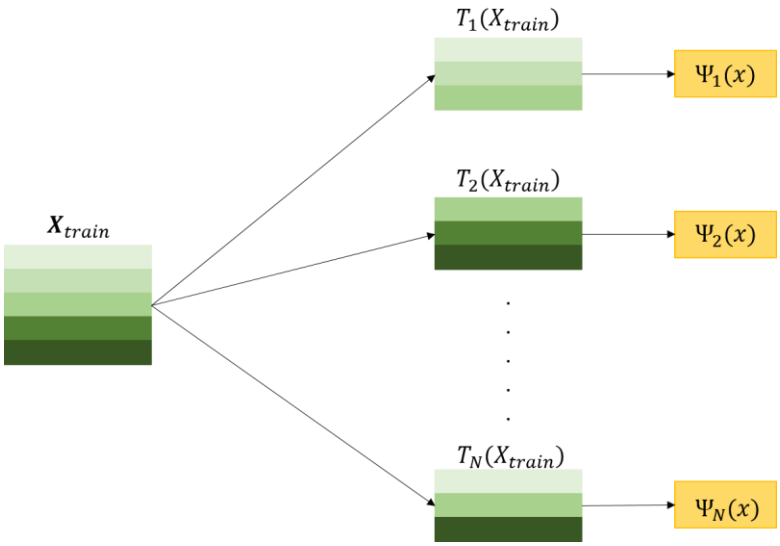
### *2.2.4 Bagging*

*Bagging* adalah gabungan dari *bootstrap* dan *aggregating,* digunakan dengan tujuan untuk mengurangi varians dari *classifier decision tree.* Metode *bagging* membagi *dataset* menjadi berbagai subset untuk *training* yang dipilih secara acak dengan substitusi. Selanjutnya subset data ini dilatih menggunakan *decision tree*. Rata-rata hasil yang diperoleh dari setiap subset data diambil yang memberikan hasil terbaik dibandingkan dengan *classifier* tunggal (Singh dan Pal, 2020). *Bagging* harus digunakan dengan algoritma *machine learning* yang tidak stabil seperti, *decision tree* atau *neural networks. Bagging* membangun *classification trees* menggunakan *bootstrap sampling* dari data *training* dan kemudian menggabungkan prediksi untuk menghasilkan meta-prediksi akhir. Model *bagging* yang digunakan pada penelitian ini adalah *random forest.*

Langkah-langkah dalam penerapan *bagging classifier :*

1. Misalkan terdapat N observasi dan M ciri dalam *dataset training*. Sampel dari kumpulan *data training* diambil secara acak dengan bergantian.
2. Subset ciri M dipilih secara acak dan ciri manapun yang memberikan pemisahan terbaik digunakan untuk membagi node secara iteratif.
3. Langkah-langkah diatas diulangi sebanyak n kali dan prediksi diberikan berdasarkan agregasi prediksi dari n jumlah *trees.*

Gambar 2.4 menunjukkan proses pengulangan N, dengan N yang memproduksi data *training* baru dan N model yang berpotensi sebagai independen (Narassiguin, 2019).




Gambar 2.4 Tahap *training bagging*

*N* bootstrap *($T_n$(Xtrain)) 1≤N* dipilih dari data pelatihan yaitu *Xtrain* dan digunakan untuk menghasilkan *ensemble model*. *Bagging* menggunakan *majority voting* atau perhitungan rata-rata, dengan menyesuaikan apakah klasifikasi atau regresi). *Bagging classifier* merupakan *ensemble meta-estimator* yang sesuai dengan pengklasifikasi dasar masing-masing pada *random subset* dari kumpulan data asli dan kemudian menggabungkan prediksi masing-masing pada untuk menentukan prediksi akhir. *Meta estimator* biasanya digunakan sebagai cara untuk

mengurangi varian dari *black-box estimator* (contohnya *decision tree*), dengan memperkenalkan pengacakan ke dalam prosedur konstruksi dan kemudian membuat *ensemble*.

Pada *jupyter notebook* untuk mengimplementasikan metode *bagging classifier* menggunakan pustaka *sklearn.ensemble.BaggingClassifier* dengan penulisan *code* sebagai berikut :

```
class sklearn.ensemble.BaggingClassifier(estimator=None, n_estimators=10, *, max_samples=1.0, max_features=1.0, bootstrap=True, bootstrap_features=False, oob_score=False, warm_start=False, n_jobs=None, random_state=None, verbose=0, base_estimator='deprecated')
```

Parameter :

a. *Estimator : object, default=None*
   *Estimator* dasar digunakan agar sesuai dengan *random subset* dari *dataset.* Jika tidak ada, maka *estimator* dasarnya adalah *DecisionTreeClassifier.*
b. *n_estimators : int, default=10*
   Jumlah *estimator* dasar dalam *ensemble*.
c. *max_samples : int or float, default=1.0*
   Jumlah sampel yang diambil dari X untuk melatih setiap *estimator* dasar.
d. *max_features : int or float, default=1.0*
   Jumlah ciri yang akan diambil dari X untuk *training* setiap *estimator* dasar.
***e.*** *bootstrap : bool, default=True*
   Apakah sampel diambil dengan *replacement.*
f. *boostrap_features : bool, default=False*
   Apakah ciri diambil dengan *replacement.*
g. *oob_score : bool, default=False*
   Apakah akan menggunakan sampel *out-of-bag* untuk memperkirakan kesalahan generalisasi, dapat dijalankan jika *bootstrap=True*.
h. *warm_start : bool, default=False*

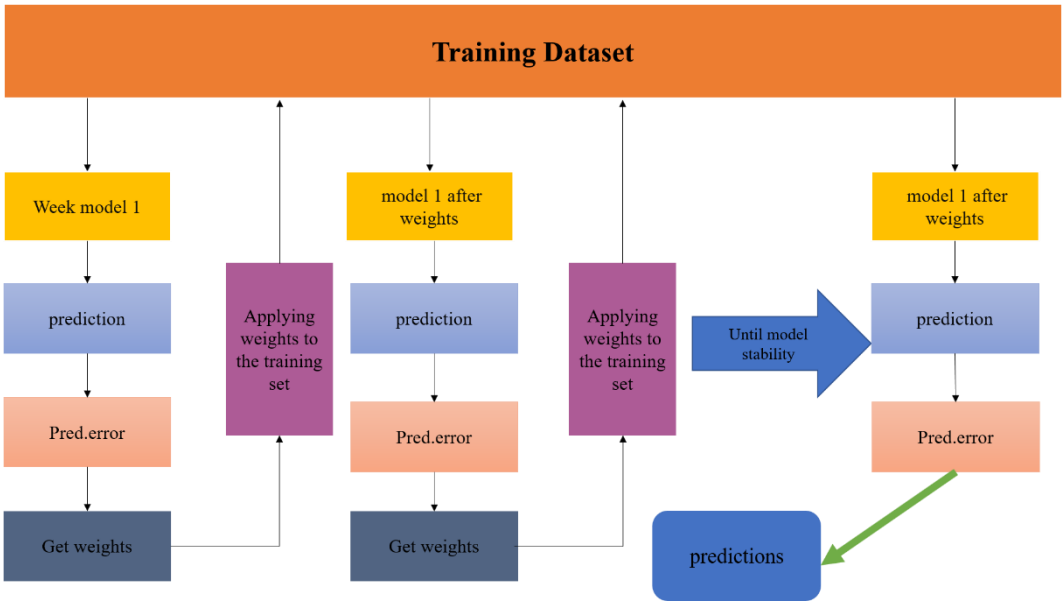
Saat bernilai *True*, solusi sebelumnya akan dipanggil untuk menyesuaikan dan menambahkan lebih banyak *estimator* ke *ensemble*, jika tidak digunakan *ensemble* yang baru.

i. *n_jobs : int, default=None*
   Jumlah tugas yang dijalankan secara paralel untuk kesesuaian dan prediksi.
j. *Random_state : int, RandomState instance or None, default=None*
   Mengontrol resampling acak dari *dataset* asli.

Parameter tersebut dapat berubah disesuaikan dengan versi *Python* yang digunakan. Jika sampel diambil dengan *replacement*, maka metode tersebut dikenal dengan *bagging*. Ketika *random subset* diambil dari *dataset* maka metode ini dikenal sebagai *random subspace*. Namun, ketika *estimator* dasar dibangun di atas himpunan bagian dari sampel dan ciri, maka metode tersebut dikenal sebagai *random patch* (Breiman, 2001).

### *2.2.5 Boosting*

*Boosting* adalah teknik *ensemble* yang diterapkan untuk membuat grup pengklasifikasi. Pada metode *boosting* pengklasifikasi dilatih secara serial oleh pengklasifikasi yang memasang data dan kemudian menganalisis kesalahan. *Boosting* digunakan untuk mengurangi bias dan varians dalam *supervised learning. Boosting* adalah salah satu metode *ensemble* untuk meningkatkan *performance* pada suatu algoritma dengan mengkombinasikan *classifier* yang lemah menjadi *classifier* yang kuat. Proses utama dari *boosting* adalah memilih sekumpulan data *training* dengan beberapa cara untuk kemudian dipelajari oleh suatu *base learner*, di mana *base learner* dipaksa menarik sesuatu yang baru tentang sampel tersebut setiap kali *base learner* itu dipanggil. Prinsip kerja *boosting* yaitu mempekerjakan sekumpulan *classifier* yang dilatih secara iteratif. Gambar 2.5 menunjukkan struktur teknik *boosting.*



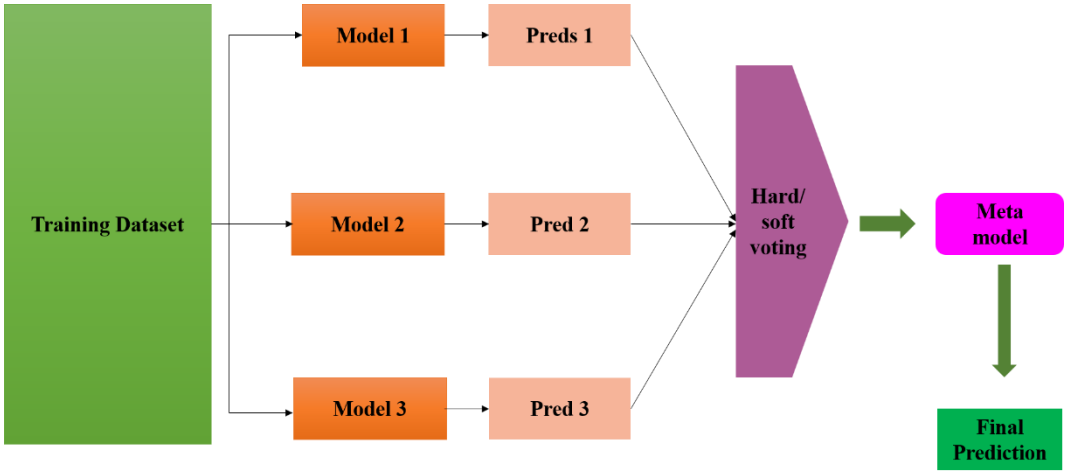

Gambar 2.5 Struktur teknik *boosting*

*Boosting* dapat membangun *estimator* dasar secara berurutan dari yang lemah, lalu mengurangi bias dari kombinasi estimator. *Boosting* sendiri pada dasarnya adalah *m* buah kombinasi linear dari $mk_m(x_i)classifier$ dengan fungsi *classifier* dimisalkan $k_m(x_i) \in \{-1,1\}$ (Zulhanif, 2016). Terdapat tiga tipe algoritma *boosting* yang sering digunakan yaitu *AdaBoost, Gradient descent, Xtreme gradient descent* dan penelitian ini menggunakan algoritma *AdaBoost.*

### *2.2.6 Voting*

*Voting* merupakan salah satu cara paling sederhana untuk menggabungkan prediksi dari beberapa algoritma *machine learning. Voting ensemble learning* adalah metodologi *ensemble machine learning* yang menggunakan banyak metode sebagai pengganti metode klasifikasi tunggal untuk meningkatkan kinerja model. *Voting ensemble* dapat digunakan untuk klasifikasi atau regresi, pada regresi melibatkan perhitungan rata-rata prediksi dari setiap model dan pada klasifikasi untuk setiap label dijumlahkan dan label dengan suara terbanyak digunakan sebagai prediksi. Terdapat dua pendekatan pada *ensemble voting classifier* yaitu *hard voting* dan *soft voting*. *Hard voting* melibatkan penjumlahan prediksi untuk setiap label kelas dan memprediksi label kelas dengan suara terbanyak. Pendekatan *hard voting*

sesuai ketika model yang digunakan dalam *ensemble voting* memprediksi label kelas yang kuat. *Soft voting* melibatkan penjumlahan probabilitas yang diprediksi untuk setiap label kelas dan memprediksi label kelas dengan probabilitas terbesar. Pendekatan *soft voting* sesuai ketika model yang digunakan dalam *ensemble voting* memprediksi probabilitas pada setiap kelas. Gambar 2.6 menunjukkan struktur *ensembel voting.*




Gambar 2.6 Struktur teknik *voting*

*Ensemble voting* tidak menjamin memberikan kinerja yang lebih baik daripada model tunggal yang digunakan jika model tertentu yang digunakan dalam *ensemble* bekerja lebih baik dari pada *ensemble voting*, model tersebut dapat digunakan sebagai pengganti model *ensemble voting*. *Ensemble voting* sangat berguna untuk model *machine learning* yang menggunakan algoritma stokastik dan menghasilkan model akhir yang berbeda setiap kali dilatih pada kumpulan data yang sama. Salah satu contohnya pada algoritma *neural network* yang menggunakan *stochastic gradient descent.* Penelitian ini menggabungkan algoritma *logistic regression* dan *support vector machine* (SVM) yang digabungkan pada model *voting.*

Pada *jupyter notebook* untuk mengimplementasikan metode *voting classifier* menggunakan pustaka *sklearn.ensemble.VotingClassifier* dengan penulisan *code* sebagai berikut :

*class* sklearn.ensemble.**VotingClassifier***(estimators, *, voting='hard', weights =None, n_jobs=None, flatten_transform=True, verbose=False)*

Parameter :

a. *estimator : list of (str, estimator) tuples*
   Memanggil fit *method* pada *VotingClassifier* untuk menyesuaikan *estimator* yang akan disimpan dalam kelas atribut *self.estimators_. Estimator* dapat diatur ke *'drop'* menggunakan *set_params.*
b. *voting : {'hard', 'soft}, default = 'hard'*
   Jika *'hard'*, digunakan pada label kelas yang diprediksi untuk aturan *majority voting*. Jika *'soft'*, digunakan pada label kelas berdasarkan *argmax* dari jumlah probabilitas yang diprediksi, untuk *ensemble classifier*.
c. *weights : array-like of shape (n_classifiers,), default=None*
   Urutan bobot *(float atau int)* untuk memberi kemunculan bobot pada label kelas yang diprediksi *(hard voting)* atau probabilitas kelas sebelum rata-rata *(soft voting).* Menggunakan bobot yang seragam jika *None.*
d. *n_jobs : int, default = None*
   Jumlah pekerjaan yang dijalankan secara paralel. *None* berarti 1 kecuali dalam konteks *joblib.paralel_beckend*. -1 berarti menggunakan semua prosesor.
e. *flatten_transform : bool, default = True*
   Mempengaruhi bentuk transformasi output hanya ketika *voting ='soft'* dan *flatten_transform = True*, metode transform mengembalikan matriks dengan bentuk *(n_samples, n_classifiers*n_classes)*. Jika *flatten_transform = False, return (n_classifiers, n_samples, n_classes).*

f. *verbose : bool, default=False*

*g.* Jika True, akan dicetak selesai.

### 2.2.7 Algoritma *Random Forest*

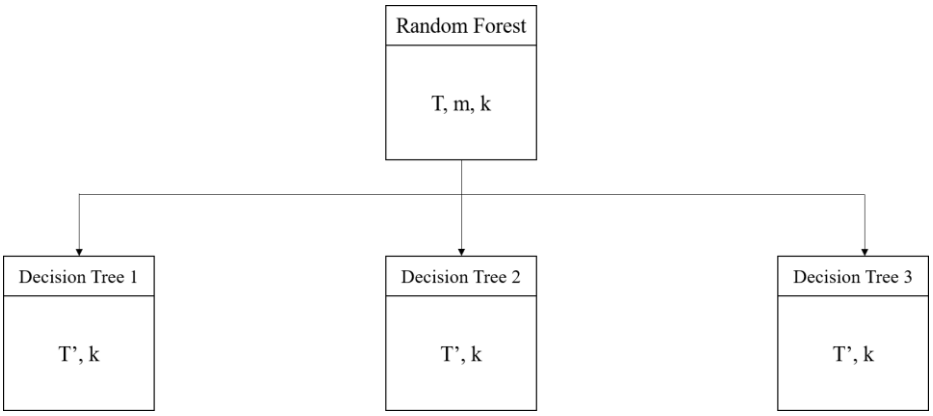
*Random forest* merupakan pengembangan dari *bagging*, di mana perbedaan utamanya dengan *bagging* adalah pemilihan ciri secara acak. Pada *Random forest* pertama-tama dilakukan pemilihan subset ciri secara acak, lalu menggunakan subset ciri tersebut dan menjalankan algoritma CART untuk membangun pohon keputusan. Banyaknya ciri yang dipilih secara acak pada RF dapat mempengaruhi tingkat kesalahan, tingkat kesalahan RF tergantung pada korelasi dan kekuatan. Memilih banyak ciri dapat meningkatkan kekuatan individu pohon keputusan sehingga tingkat kesalahan menurun. Namun, ketika terlalu banyak ciri yang dipilih, pohon-pohon yang terbentuk akan memiliki korelasi tinggi karena identik. Korelasi yang tinggi antara dua individu pohon dapat meningkatkan kesalahan hutan (gabungan pohon-pohon). Maka untuk menurunkan korelasi antar individu pohon, maka banyaknya ciri yang dipilih harus diturunkan (Breiman, 2001).

Konsep dasar dari metode ini adalah dengan membangun banyak *tree* untuk mengklasifikasikan suatu objek berdasarkan atribut. Setiap *tree* akan menghasilkan klasifikasi masing-masing, dari apa yang disebut dengan '*votes*' untuk kelas tertentu. Selanjutnya metode ini akan memilih *vote* yang terbanyak dari hasil klasifikasi *tree* (Tahyudin dkk, 2021). Konsep detail dari *random forest* adalah masing-masing *tree* ditanam dan tumbuh dengan proses sebagai berikut :

1. Asumsi jumlah kasus pada *training* set N. Kemudian, diambil sampel dari kasus N secara *random* tetapi dengan penggantian. Sample ini akan ditraining untuk menumbuhkan *tree*.
2. Jika terdapat variabel *input* M sebuah bilangan m<M yang ditentukan sedemikian rupa sehingga menjadi beberapa *node*, variabel m dipilih secara *random* pada M. Pilihan terbaik pada m digunakan untuk membagi *node*. Nilai pada m bersifat konstan ketika menumbuhkan *forest.*
3. Masing-masing *tree* ditumbuhkan menjadi lebih besar tanpa pemotongan.

4. Memprediksi data baru dengan cara mengagregasi prediksi pada n *tree* (contoh mengambil *majority votes* untuk klasifikasi dan rata-rata untuk regresi).

*Random forest* memiliki dua parameter utama, yaitu: m jumlah *tree* yang akan dipakai dan k yaitu maksimal banyaknya ciri yang dipertimbangkan ketika proses percabangan. Semakin banyak nilai m maka semakin bagus hasil klasifikasi, sedangkan untuk nilai k direkomendasikan sebesar akar kuadrat atau logaritma dari jumlah total ciri. Konsep *random forest* diilustrasikan oleh gambar 2.7 (Wibowo, 2016).




Gambar 2.7 Ilustrasi konsep *random forest*

Gambar 2.7 menunjukkan proses *training* untuk *random forest* menggunakan *dataset* T dengan sejumlah m *tree* sebagai *basic learner* dan k ciri yang dipilih secara acak dari total ciri yang ada untuk percabangan pada setiap tree. *Dataset* T yang digunakan pada proses *training tree* merupakan hasil dari *bootstrap* dari *dataset* yang dijadikan parameter untuk *random forest. Bootstrap* adalah proses memilih sampel dari *dataset* yang akan digunakan proses *training tree*. Metode *ensemble, bootstrap* merupakan proses *sampling* dengan *replacement*, sehingga sampel yang diambil untuk proses *training tree* yang satu masih bisa dipakai lagi untuk proses *training tree* lainnya (Wibowo, 2016). *Random forest* bergantung pada sebuah nilai *vector random* dengan distribusi yang sama pada semua pohon yang masing-masing *decision tree* memiliki kedalaman yang maksimal. *Random forest* merupakan *classifier* yang berbentuk pohon {h(x, ɵ k), k = 1, . . . } dimana ɵk adalah

*random vector* yang didistribusikan secara independen dan masing-masing *tree* pada sebuah unit akan memilih *class* yang paling populer pada *input* x.

*Random forest* menggunakan *gini measure* untuk memilih bagian terendah disetiap *node*. *Gini impurity* adalah ukuran distribusi label kelas di seluruh *node.* Secara umum untuk mengukur *gini impurity* pada variabel $X = \{x_1, x_2, \dots, x_j\}$ pada node *t*, dimana *j* adalah jumlah anak pada node *t, N* adalah jumlah sampel. Persamaan 2.3 menunjukkan formula *Gini Impurity* (Dhiyaussalam dkk, 2020).

$$I(t_{xi}) = 1 - \sum_{c=0}^{c} \left(\frac{n_{ci}}{m_i}\right)^2 \quad (2.3)$$

Dengan :

$n_{ci}$ adalah jumlah sampel dari nilai $x_i$ pada kelas *c.*

$m_i$ adalah jumlah sampel dengan nilai $x_i$ pada node *t.*

Persamaan 2.4 menunjukkan *gini index* dari nilai yang berbeda dari X.

$$G(r, X) = \sum_{i=1}^{j} \frac{m_i}{N} I(r_{xi}) \quad (2.4)$$

*Feature importance* dapat dihitung dari rata-rata *impurity reduction* dari semua *decision tree* dalam *random forest* tanpa mengasumsikan apakah data yang digunakan terpisah secara linier atau tidak. Persamaan 2.5 menunjukkan formula untuk menghitung *feature importance* dari setiap *decision tree.*

$$FI_i = \frac{\sum_j G_{ij}}{\sum_k G_{ik}} \quad (2.5)$$

Dengan :

$FI$ adalah feature-*i* pada *decision tree.*

*k* adalah representasi dari semua *node.*

Selanjutnya untuk menghitung nilai dari *feature importance* pada *random forest* ditunjukkan pada persamaan 2.6.

$$RFFI_i = \frac{\sum_j FI_{ij}}{T} \quad (2.6)$$

Dengan :

*RFFI* adalah feature-i pada random forest.

T adalah jumlah decision tree.

Algoritma *random forest* memiliki kelebihan, antara lain sebagai berikut (Breiman, 2001) :

1. Memiliki akurasi sama baiknya dengan *AdaBoost* dan terkadang lebih baik.
2. Tahan terhadap pencilan dan *noise.*
3. Memberikan estimasi dari error, kekuatan, korelasi dan peubah yang penting.
4. Sederhana dan mudah diparalelkan.

Kekurangan menggunakan random forest (Tahyudin dkk, 2021):

1. Metode ini bagus pada kasus klasifikasi tetapi tidak tepat untuk regresi karena metode ini memberikan prediksi secara kontinu. Pada kasus regresi metode ini memprediksi *data training* sehingga kemungkinan terjadi *over-fit* pada *dataset.*
2. Metode ini bersifat *black box* sehingga *user* tidak banyak memberikan intervensi pada penentuan model terbaik. *User* hanya dapat merubah parameter dan benih secara *random* untuk menghasilkan yang terbaik.

### 2.2.8 *AdaBoost*

*AdaBoost* adalah metode *ensemble* yang berulang. Algoritma *AdaBoost* atau *Adaptive Boosting,* diterapkan secara luas pada model prediksi dalam *data mining.* Inti dari algoritma *AdaBoost* memberikan suatu bobot lebih pada observasi yang tidak tetap *(weak classification).* Algoritma *AdaBoost* merupakan *meta-estimator* yang dimulai dengan memasang sebuah *classifier* pada *dataset* asli dan kemudian menyesuaikan salinan tambahan dari *classifier* pada *dataset* yang sama tetapi bobot *instance* sudah disesuaikan, sehingga *classifier* berikutnya lebih fokus pada kasus yang sulit. *Adaboost* membangun *classifier* yang kuat dengan menggabungkan *classifier* yang lemah, sehingga didapatkan *classifier* kuat dengan akurasi yang

tinggi. *Boosting* sendiri pada dasarnya adalah *m* buah kombinasi linear dari $mk_m(x_i)classifier$ dengan fungsi *classifier* dimisalkan $k_m(x_i) \in \{-1,1\}$ (Zulhanif, 2016).

Lagkah-langkah penerapan *AdaBoost* adalah sebagai berikut ;

1. *AdaBoost* memilih *subset training* secara acak.
2. *Subset training* pada *Adaboost* memilih *set training* berdasarkan prediksi akurat dari *data train* terakhir.
3. Pembobotan dengan memberikan bobot yang lebih tinggi untuk pengamatan data yang salah diklasifikasikan sehingga pada iterasi berikutnya pengamatan ini akan mendapatkan probabilitas yang tinggi untuk klasifikasi.
4. Pembobotan juga diberikan ke *classifier* terlatih di setiap iterasi sesuai dengan keakuratan *classifier*. *Classifier* yang lebih akurat akan mendapatkan bobot yang tinggi.
5. Proses ini berulang hingga *data training* lengkap sesuai tanpa kesalahan atau hingga mencapai jumlah estimator maksimum yang ditentukan.

*AdaBoost Classifier* adalah *meta-estimator* yang dimulai dengan memasang *classifier* pada *dataset* asli dan kemudian menyesuaikan salinan *classifier* tambahan pada *dataset* yang sama tetapi, bobot yang salah diklasifikasikan akan disesuaikan sedemikian rupa sehingga pengklasifikasi berikutnya lebih fokus pada kasus yang sulit. Implementasi *AdaBoost Classifier* pada *Python jupyter notebook* menggunakan pustaka *sklearn.ensemble.AdaBoostClassifier* dengan *code* sebagai berikut :

```
class sklearn.ensemble.AdaBoostClassifier(estimator=None, *, n_estimators=50, learning_rate=1.0, algorithm='SAMME.R', random_state=None, base_estimator='deprecated')
```

Parameter :

a. *estimator : object, default=None*

*Estimator* dasar untuk sampel pembobotan, serta atribut *classes_* dan *n_classes_atributes*. Jika tidak ada, maka *estimator* dasar adalah *DecisionTreeClassifier* yang diinisialisasi dengan *max_depth=1*.

b. *n_estimators : int, default=50*

   Jumlah maksimum *estimator* di mana *boosting* dihentikan. Jika tepat, maka prosedur pembelajaran dihentikan lebih awal. Nilai harus dalam rentang [1,inf].

c. *learning_rate : float, default=1.0*

   Bobot diterapkan ke setiap *classifier* pada setiap iterasi *boosting.* Tingkat pembelajaran yang lebih tinggi meningkatkan kontribusi setiap *classifier.*

d. *algorithm : {'SAMME', 'SAMME.R'}, default='SAMME.R'*

   Algoritma SAMME.R biasanya konvergen lebih cepat daripada algoritma SAMME, mencapai kesalahan pengujian yang lebih rendah dengan peningkatan iterasi yang lebih sedikit.

e. *random_state : int, RandomState instance or None, default=None*

   Mengontrol *random seed* yang diberikan pada setiap estimator pada iterasi *boosting.* Hal ini hanya digunakan ketika estimator menampilkan random_state.

f. *base_estimator : object, default=None*

   *Estimator* dasar untuk *ensemble* dengan sampel pembobotan yang diperlukan seperti *classes_* and *n_classes_ attributes*. Jika *None*, kemudian estimator dasarnya adalah *DecisionTreeClassifier* dengan inisialisasi *max_depth=1*.

### 2.2.9 *Logistic Regression*

*Logistic Regression* merupakan *supervised algorithm* yang melibatkan lebih banyak ciri dependen. *Logistic regression* adalah model linier yang digunakan untuk memperkirakan hubungan antar variabel dengan menggunakan fungsi logistik (Priya dkk, 2021). *Logistic regression* memperkirakan kemungkinan suatu peristiwa terjadi, seperti memilih atau tidak memilih, berdasarkan *dataset* variabel independen dibatasi antara 0 dan 1. *Logistic regression* adalah model statistik yang menggunakan fungsi logistik, atau fungsi logit, dalam matematika sebagai

persamaan antara *x* dan *y*. Fungsi logit memetakan *y* sebagai fungsi *sigmoid* dari *x*. Fungsi *logistic regression* dijelaskan pada persamaan berikut ;

$$f(x) = \frac{1}{1+ e^{-x}} \tag{2.7}$$

Model *logit* juga dapat menentukan rasio keberhasilan terhadap kegagalan atau *log odds.* Secara matematis, peluang dalam hal probabilitas adalah *p/(1-p)*, dan peluang *log odds* adalah *log(p/(1-p)).* Persamaan fungsi *logistic* sebagai *log odss* seperti ditunjukkan di bawah ini ;

$$Logit\ Function = log\left(\frac{p}{1-p}\right) \tag{2.8}$$

*Logistic Regression* diimplementasikan kedalam model linear untuk klasifikasi dari regresi pada *scikit-learn* atau ML. Pada kasus *multiclass,* algoritma *training* menggunakan skema *one-vs-rest (OvR)* jika opsi *multi_class* di set ke *'ovr'*, dan menggunakan *'cross-entropy'* jika *'multi-class* di set ke *'multinomial'*. Pada *Python* untuk mengimplementasikan *logistic regression* menggunakan modul *sklearn.linear_model.LogisticRgression*.

*class* sklearn.linear_model.**LogisticRegression**(*penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*)

Parameter :

a. *penalty : {'l1', 'l2', 'elasticnet', None}, default='l2'*
b. *dual : bool, default=False*

   Formulasi ganda atau prima. Formulasi ganda hanya diterapkan untuk penalti *l2* dengan pemecah *liblinear*. *dual=False when n_samples > n_features.*
c. *tol : float, default=1e-4*

   Toleransi untuk kriteria.
d. *c : float, default = 1.0*

Kebalikan dari *strength regularization*; sebuah *positive float*. Seperti pada SVM, nilai yang lebih kecil menentukan regulasi yang lebih kuat.

e. *fit_intercept : bool, default=True*

Menentukan apakah konstanta *(bias atau intercept)* harus ditambahkan ke fungsi keputusan.

f. *intercept_scaling : float, default=1*

Berguna hanya ketika pemecah *'liblinear'* digunakan dan *self.fit_intercept* di set ke *True.* Dalam hal ini, *x* menjadi *[x, self.intercept_scaling]*, yaitu ciri sintesis dengan nilai konstanta yang sama dengan *intercept_scaling* ditambahkan ke *instance vector*. Maka *intercept* menjadi *intercept_scaling*syinthetic_feature_weight.*

g. *class_weight : dict or 'balanced', default=None*

Bobot yang terikat dengan kelas dalam bentuk *{class_label:weight}*. Jika tidak diberikan, maka semua kelas memiliki bobot satu. Mode *'balanced'* menggunakan nilai y untuk secara otomatis menyesuaikan bobot berbanding terbalik dengan frekuensi kelas dalam data *input* sebagai *n_samples/(n_classes*np.bincunt(y)).*

h. *random_state : int, RandomState instance, default=None*

Digunakan saat *solver == 'sag', 'saga'* atau *'liblinear'* untuk mengacak data.

i. *solver : {'lbfgs', 'liblinear', 'newton-cg', 'newton-cholesky', 'sag', 'saga'}, default='lbfgs'*

Algoritma digunakan untuk optimasi dalam pemecahan masalah. Standarnya adalah *'lbfgs'.*

j. *max_iter : int, default=100*

Jumlah maksimum iterasi yang diambil oleh pemecah konvergen.

k. *multi_class : {'auto', 'ovr', 'multinomial'}, default='auto'*

Jika opsi yang dipilih adalah *'ovr'*, maka masalah biner cocok untuk setiap label. Untuk *'multinomial'*, yang diminimalkan adalah kecocokan multinomial diseluruh distribusi probabilitas.

l. *varbose : int, default=0*

Untuk *solver liblinear* dan *lbfgs, set varbose* ke angka positif apapun untuk *verbositas.*

m. *warm_start : bool, default=False*

Saat di set ke True, maka gunakan kembali solusi sebelumnya agar pas sebagai inisialisasi, jika tidak hapus saja.

n. *n_jobs : int, default=None*

Jumlah inti CPU yang digunakan saat memparalelkan kelas, jika *multi_class='ovr'*. Parameter ini diabaikan saat *solver* diset ke *'liblinear'* terlepas dari apakah *'multi_class'* ditentukan atau tidak.

o. *|1_ratio : float, default=None*

Parameter pencampuran *Elastic-Net*, dengan *0<=|1_ratio<=1*. Hanya digunakan jika *penalty='elasticnet'.*

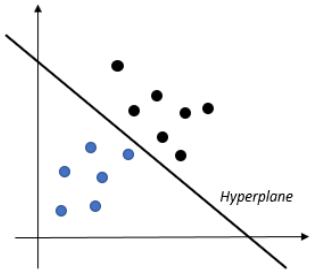
Kelas ini mengimplementasikan *logistic regression* menggunakan *'liblinear', solver 'newton-cg', 'sag', 'saga', dan 'lbfgs'*. Regulasi ini diterapkan secara *default.* Gunakan *C-ordered* atau matriks CSR yang berisis *float 64-bit* untuk performa optimal; format input lainnya akan dikonversi dan disalin (Hsiang dkk, 2011).

### 2.2.10 *Support Vector Machine (SVM)*

Menurut Y.Yin, Han dan Cai, (2019) *Support Vector Machine* (SVM) adalah seperangkat metode pembelajaran terkait analisis data dan pengenalan pola, yang kemudian digunakan untuk klasifikasi SVM secara bersamaan meminimalkan kesalahan klasifikasi empiris dan memaksimalkan *margin geometrik. Support Vector Machine* (SVM) adalah pendekatan pembelajaran mesin yang menggunakan pengklasifikasi *linear* untuk mengklasifikasi data ke dalam dua kategori (Peter dkk, 2019). *Support Vector Machine* adalah jenis algoritma *supervised learning* dimana SVM memiliki fungsi untuk mengklasifikasikan data pada variabel yang memiliki nilai prediktif, artinya variabel-variabel tersebut memiliki nilai yang dapat diprediksi untuk masa depan.

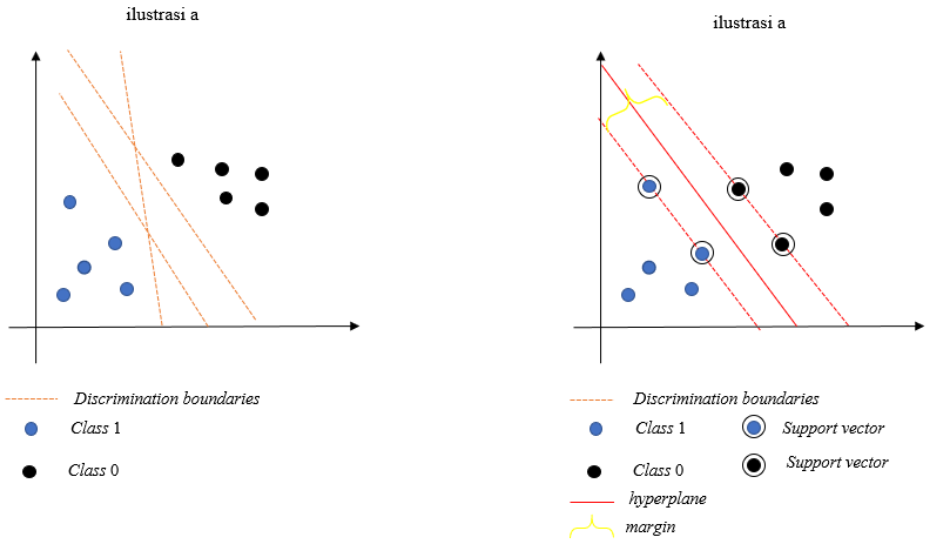
Terdapat dua jenis data dalam SVM, yaitu: data latih *(training)* dan data uji *(testing). Data training* digunakan untuk menentukan pemisah antar kelompok data.

Data *testing* adalah data yang akan ditentukan klasifikasinya. Konsep SVM dapat dijelaskan sebagai upaya untuk menemukan *hyperplane* terbaik yang berfungsi sebagai pemisah dua kelas di ruang *input. Hyperplane* pemisah terbaik antara dua kelas dapat ditemukan dengan mengukur margin *hyperplane* dan mencari titik maksimum. *Margin* adalah jarak antara *hyperplane* dari setiap kelas disebut *support vector* (Novenando dkk, 2019).




Gambar 2.8 *Hyperplane*

Gambar 2.8 mengilustrasikan data yang terbagi menjadi dua kelompok, yaitu kelompok lingkaran biru dan lingkaran hitam. Kedua kelompok tersebut dipisahkan oleh garis demarkasi atau *hyperplane* (Kusuma, 2020). Separator *hyperplane* terbaik dapat diperoleh dengan mengatur *margin hyperplane* dan titik maksimumnya.




Gambar 2.9 Model *support vector machine*

Gambar 2.9 ilustrasi a dan ilustrasi b memperlihatkan beberapa pola yang merupakan anggota dari dua kelas yaitu 1 dan 0. *Class* 1 digambarkan oleh lingkaran biru sedangkan *class* 0 digambarkan oleh lingkaran hitam. *Discrimination boundaries* sebagai alternatif garis pemisah, *hyperplane* pemisah yang terbaik diantara kedua kelas ditemukan dengan cara mengukur *margin hyperplane* dan mencari titik maksimumnya. *Margin* adalah jarak antara *hyperlink* dan data terdekat dari masing-masing kelas. Pada ilustrasi b, *margin* digambarkan dengan kurung kurawal warna kuning. Data yang paling dekat dengan *hyperplane* terbaik disebut *support vector*. Ilustrasi b menunjukkan *hyperplane* terbaik yang terletak tepat ditengah-tengah kedua kelas, sedangkan titik biru dan hitam yang berada dalam lingkaran hitam adalah *support vector* (Kasim dan Sudarsono, 2019).

*Support vector machine* (SVM) merupakan teknik *supervised classification* yang cukup rumit tetapi memiliki tingkat keakuratan yang cukup baik. SVM adalah metode yang ideal ketika batas kelas *nonlinier* memiliki terlalu sedikit data untuk mempelajari model *nonlinier* kompleks. Algoritma SVM akan menemukan *hyperplane* atau batas antara dua kelas lebih dengan memaksimalkan margin di antara kelas-kelas tersebut. Terdapat banyak batas yang bisa memisahkan kelas-kelas tersebut, tetapi hanya ada satu batas yang dapat memaksimalkan margin atau jarak antar kelas (Romero dan Ventura, 2020). Konsep SVM adalah mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dari dua data kelas, kelas positif (+1) dan kelas negatif (-1), seperti persamaan berikut (Parapat dan Furqon, 2018) ;

$$w..x + b = 0 \tag{2.9}$$

Dengan :

*w* adalah bobot vektor seperti $w = \{w_1, w_2, ..., w_n\}$.

*n* adalah jumlah atribut.

*b* adalah *scalar* yang disebut bias.

Persamaan untuk menghitung nilai *b* dan *w* adalah sebagai berikut ;

$$b = -\frac{1}{2}(w.x^{+} + w.x^{-}) \tag{2.10}$$

*b* adalah scalar atau nilai bias.

$w.x^+$ adalah nilai bobot untuk kelas data positif.

$w.x^-$ adalah nilai bobot untuk kelas data negatif.

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \quad (2.11)$$

W adalah bobot vektor

$\alpha_i$ adalah nilai bobot data ke-*i*

$y_i$ adalah kelas data ke-*i*

$x_i$ adalah data ke-*i*

Hyperplane ditandai dengan $H_1$ sebagai pendukung dari kelas +1 yang memiliki fungsi $wx1 + b = +1$.

$$Margin = | dH_1 - dH_2 | = \frac{2}{||w||} \quad (2.12)$$

Dengan :

$dH_1$ adalah jarak *hyperplane* pendukung kelas +1

$dH_2$ adalah jarak *hyperplane* pendukung kelas -1

Persamaan untuk menentukan optimal *hyperplane* pada kedua kelas menggunakan persamaan berikut ;

$$Minimize\ J1[w] = \frac{1}{2} \|w\|^2 \quad (2.13)$$

Dengan $yi(xi.w + b) - 1 \geq 0, i = 1, \ldots, n$

Meskipun *support vector machine* (SVM) merupakan model *linier* yang kuat yang dapat diperluas ke masalah *nonlinier* melalui titik kernel, algoritma ini memiliki banyak parameter yang harus disesuaikan untuk membuat prediksi yang baik. Sebaliknya, metode *ensemble* seperti *random forest* tidak memerlukan banyak setelan parameter dan tidak terlalu mudah dipasangkan seperti *decision tree*, yang membuatnya menjadi model domain praktis (Raschka dan Mirjalili, 2017). Pustaka

*Python* yang dapat digunakan untuk *support vector machine* diantaranya adalah *pandas* digunakan untuk manajemen data, *NumPy* digunakan untuk yang berhubungan angka, pustaka *Sklearn* atau *scikit learn* yang digunakan untuk memanggil algoritma *support vector machine*, dan juga *Matplotlib* dan *seaborn* untuk visualisasi.

### 2.2.11 *Python dan Jupyter Notebook*

*Python* adalah salah satu bahasa pemrograman yang populer untuk ilmu data dan memiliki pustaka tambahan untuk *data mining* dan bersifat *open source*. *Python* juga digunakan pada *machine learning* dan *data science*, memiliki peran penting karena dapat membantu analisis statistik, informasi visualisasi dan plot grafis. *Python* memiliki pustaka yang dapat digunakan untuk klasifikasi, pengelompokan data dan aplikasi desktop *machine learning* seperti, *scikit-learn, pandas*, dan *numpy* (Guleria dan Sood, 2018). *Python* mampu menampung *dataset* yang lebih besar, dibandingkan *tools* lainnya yang masih memiliki batasan ukuran (Slater dkk, 2017).

Ciri lain yang berguna dari *Python* adalah *jupyter notebook*, yang merupakan aplikasi klien-server yang memungkinkan untuk pembuatan dan modifikasi kode *Python* dan elemen teks seperti grafik dan tabel berbasis *web browser*. Proyek *jupyter notebook* dimulai pada tahun 2014 yang bertujuan untuk membuat seperangkat sumber yang konsisten untuk penelitian ilmiah, alur kerja yang dapat direproduksi, narasi, komputasi dan analitik data. Saat ini *jupyter notebook* telah menjadi bagian penting dari perangkat *data science. Jupyter Notebook* juga memungkinkan untuk membuat dan berbagi dokumen interaktif yang berisi kode dinamis dan dapat dijalankan. Protokol *jupyter notebook* menyediakan API standar untuk berkomunikasi dengan kernel yang bertindak sebagai mesin komputasi. Protokol memungkinkan arsitektur yang dapat disusun dengan memisahkan tempat untuk menulis kode dan tempat untuk mengeksekusi kode *(kernel).*

*Jupyter Notebook* sangat berguna untuk membuat prototipe algoritma atau menguji potongan kode untuk menganalisis hasil dan menambahkannya ke proyek utama. Proses implementasi dari model *ensemble machine learning* dengan aplikasi berbasis web menggunakan *framework* berbasis *Python* dan bersifat *open-source*

yang disebut *streamlit. Streamlit* dibuat untuk memudahkan dalam membangun aplikasi web pada sains data *dan machine learning* yang interaktif. *Framework* ini dapat menerima *input*, membuat grafik dengan visualisasi yang menarik dan menampilkannya dengan cepat (Richards, 2021). Pustaka *Python* yang digunakan dalam penelitian ini adalah sebagai berikut :

1. *Numpy (Numerical Python)*

   Numpy adalah pustaka *open-source Python* untuk komputasi ilmiah dengan memproses *array* dan matriks. *Numpy* memiliki *high-performing N-dimensional array object*, artinya *Numpy* mendukung *array* satu dimensi dan multidimensi dengan objek *array homogen*. Pada *jupyter notebook* untuk menggunakan pustaka *Numpy* harus melakukan *install Numpy* kemudian *import.* Contohnya *import numpy as np,* pada contoh tersebut mengimpor *Numpy* dan menggunakan alias np. Setelah itu fungsi-fungsi pada *Numpy* dapat digunakan.

2. *Scikit-learn*

   *Scikit-learn* adalah pustaka *Python* pada *machine learning* yang bekerja dengan pustaka *Numpy, Matplotlib* dan *SciPy Python* yang berisi *support vector machine, random forest, gradient boosting, k-means* dan DBSCAN. *Scikit-learn* dapat melakukan pekerjaan dalam *data science*, seperti regresi *(regression),* klasifikasi (*classification),* pengelompokkan *(clustering),* data *preprocessing* dan pemilihan model (Garreta dan Moncecchi, 2013). Contoh penggunaan *scikit-learn* di *jupyter notebook* untuk *preprocessing* adalah *from sklearn.preprocessing import LabelEncoder,* untuk pemilihan model *from sklearn.model_selection import train_test_split* dan untuk evaluasi menggunakan *from sklearn.metrics import classification report.*

3. *Pandas*

   *Pandas* adalah salah satu pustaka *Python* yang *open source*, menyediakan struktur data dan analisis data yang mudah. *Pandas* dapat melakukan tugas seperti menyelaraskan data untuk perbandingan dan penggabungan *dataset*, penanganan pada *missing data* dan lain sebagainya. *Pandas* merupakan turunan dari pustaka *Numpy*, penyajian data pada *Numpy* umumnya dalam

bentuk *array* dimensi. *Pandas* melakukan adopsi pada proses numerik yang dimiliki *numpy* pada penyajian data. Terdapat tiga bentuk penyajian pada *pandas Python* yaitu *series, dataframe* dan data panel. *Series* merupakan bagian data yang menjadi satu atribut dan kesatuan *record*. *Dataframe* adalah gabungan beberapa series yang saling terhubung. Susunan *dataframe* dalam baris lapisan yang saling berhubungan, sehingga terbentuk data panel (Albanna dkk, 2022).

4. *Seaborn*

   *Seaborn* adalah pustaka untuk membuat grafik statistik dengan *Python.* Diintegrasikan dengan *Matplotlib* dan struktur data menggunakan pustaka *pandas. Seaborn* membantu dalam memahami data dan fungsi *plotting* yang beroperasi pada kerangka data dan *array* yang berisi kumpulan data utuh dan secara internal melakukan pemetaan semantik dan agregasi statistik yang diperlukan untuk menghasilkan plot yang informatif (Waskom, 2022). Penggunaan *Seaborn* pada *Python* dengan melakukan *import seaborn as sns*, namun sebelumnya harus menginstal pustaka *pip instal seaborn.*

5. *Matplotlib*

   *Matplotlib* merupakan pustaka *Python* untuk menampilkan data atau grafik pada *array* 2D. *Matplotlib* berawal dari grafis MATLAB namun, ini tidak bergantung pada MATLAB dan dapat digunakan dalam *Python object oriented.* Meskipun *Matplotlib* ditulis dengan *Python, Matplotlib* juga banyak menggunakan *NumPy* dan ekstensi kode lainnya untuk memberikan kinerja yang baik (Hunter dkk, 2017). Penggunaan *Matplotlib* pada *Python* dengan perintah *pip* dari *prompt* berikut ini *pip install Matplotlib.* Sedangkan untuk menggunakannya menggunakan perintah berikut : *import Matplotlib.pyplot as plt.*

### 2.2.12 *Confusion Matrix* (Pengukuran Kinerja)

*Confusion Matrix* adalah salah satu metode pengukuran kinerja yang terdapat pada algoritma klasifikasi. Pengukuran kinerja sistem klasifikasi memperhatikan seberapa baik sistem dalam melakukan proses klasifikasi data. *Confusion matrix* merupakan sebuah tabel yang memberikan informasi yang membandingkan hasil

klasifikasi yang dilakukan oleh sistem dengan hasil klasifikasi nyata. Hal ini menggambarkan seberapa banyak data yang diprediksi secara benar atau salah oleh sistem. Perhitungan kinerja model dalam *confusion matrix* berdasarkan pada nilai *true positif* (TP), *true negative* (TN), *false positive* (FP) dan *false negative* (FN) (El Aissaoui dkk, 2019). Bentuk dari *confusion matrix* ditampilkan pada tabel 2.1.

Tabel 2.2 *Confusion matrix*

| **Aktual** | **Prediksi** | |
|---|---|---|
| | ***True*** | ***False*** |
| *True* | *TP (True Positive)* | *FP (False Positive)* |
| *False* | *FN (False Negative)* | *TN (True Negative)* |

Tabel 2.2 terkait *confusion matrix* nilai *true negative* (TN), *false positive* (FP), *false negative* (FN), dan *true negative* (TN) dapat diperoleh nilai akurasi. Nilai akurasi menggambarkan seberapa akurat sistem dapat mengklasifikasikan data secara benar. Perhitungan akurasi dengan persamaan 2.14 berikut ini :

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN}\ x\ 100\% \tag{2.14}$$

Nilai *precision* adalah nilai perbandingan dari data yang diprediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif. Perhitungan nilai *precision* dapat dilakukan dengan persamaan 2.15 berikut ini :

$$Precision = \frac{(TP)}{(TP+FP)}\ x\ 100\% \tag{2.15}$$

Nilai *recall* (sensitifitas) adalah nilai perbandingan dari data yang diprediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. Perhitungan nilai *recall* dapat dilakukan dengan persamaan 2.16.

$$Recall = \frac{(TP)}{(TP+FN)}\ x\ 100\% \tag{2.16}$$

Nilai *F-1 Score* adalah nilai timbal balik yang diperoleh dari perhitungan *recall* dan presisi. Perhitungan *F-1 Score* dapat dilakukan dengan persamaan 2.17 berikut ini:

$$F_{1\ Score} = 2\ x\ \frac{presisi\ x\ recall}{presisi + recall}\ x\ 100\% \quad (2.17)$$

### 2.2.13 Aplikasi *Data mining* pada Data Pendidikan

*Data mining*, juga dikenal dengan *Knowledge Discovery in Data (KDD)*, sebuah proses yang dapat menemukan informasi tersembunyi dan berguna dari sejumlah kumpulan data besar. Banyak bidang telah memanfaatkan teknik data mining seperti, *e-commerce, bioinformatic*, dan *e-learning* untuk membantu dalam pengambilan keputusan (El Aissaoui dkk, 2020). Proses EDM mengubah data mentah yang dikumpulkan dari repositori pendidikan yang berbeda menjadi informasi yang bermakna dan berguna sehingga dapat dilakukan penelitian tentang prestasi akademik, meningkatkan kinerja siswa. Ada beberapa teknik EDM seperti *association, clustering, classification, regression* dan masih banyak lagi dapat diterapkan pada data pendidikan untuk mengekstrak informasi yang tersembunyi (Shrestha dan Pokharel, 2019)

*Educational Data Mining* (EDM) adalah disiplin ilmu yang menggunakan teknik *data mining* di lingkungan pendidikan, yang mampu memprediksi informasi yang berguna dari *database* pendidikan. Teknik *data mining* yang sering digunakan seperti *k-nearest neighbor, neural networks, decision tree, support vector machines, naive bayes* dan masih banyak lagi (Jalota dan Agrawal, 2019).

### 2.2.14 Perilaku Belajar dan Kinerja Akademik Siswa

Perilaku belajar adalah suatu aktivitas mental/psikis yang berlangsung dalam interaksi aktif dengan lingkungan yang menghasilkan perubahan-perubahan dalam pengetahuan, pemahaman, keterampilan, dan nilai sikap. Perilaku belajar memiliki dua penilaian kualitatif yakni baik dan buruk tergantung kepada individu yang mengalaminya, untuk merespon dengan baik atau bahkan acuh tak acuh. Perilaku belajar melibatkan beberapa faktor seperti motivasi, minat, persepsi, memori, perhatian, dan pengalaman belajar sebelumnya. Perilaku belajar juga dapat dipengaruhi oleh startegi belajar individu, seperti penggunaan teknik belajar tertentu atau penggunaan sumber daya yang tepat contohnya pada pemilihan *e-learning* yang tepat (Alshammari, 2020).

Keterlibatan siswa dalam penggunaan *e-learning* merupakan konsep multidimensi yang berkaitan dengan aspek psikologis, perilaku siswa, desain pedagogis, dan perlunya intervensi untuk memastikan hasil belajar (K. Lee dan Song, 2019). Memahami perilaku siswa sangat penting untuk menciptakan pembelajaran yang berpusat pada siswa atau personalisasi siswa. Interaksi siswa dengan *e-learning* adalah proses saling berhubungan antara siswa dan *paltform pembelajaran online*. Interaksi ini dapat terjadi melalui media seperti forums dikusi, chat, interaksi siswa dengan materi pembelajaran, seperti melalui modul pembelajaran atau latihan soal.

Penggunaan *e-learning* melalui komputer dan internet dari mana saja dan kapan saja menyebabkan sulit untuk memahami perilaku siswa secara langsung. Namun, dengan menggunakan *learning management system* (LMS), perilaku siswa dapat terekam dari aktivitas yang disimpan pada *log* LMS (Purwoningsih dkk, 2019). *Machine learning* dapat digunakan untuk memprediksi kinerja siswa dan model *machine learning* digunakan untuk "belajar" tentang setiap siswa, yang memungkinkannya untuk mengidentifikasi kekurangan mereka dan menentukan cara-cara yang dapat ditingkatkan (Awad dkk, 2020).

Masing-masing siswa memiliki kinerja yang berbeda-beda. Pengelompokan terhadap kinerja akademik siswa bertujuan untuk memberikan perlakuan kepada siswa berdasarkan kecakapan dalam bidang akademik serta membantu tiap siswa yang membutuhkan perhatian lebih (Huriah, 2018). Terdapat banyak faktor yang dapat mempengaruhi kinerja akademik yng telah dikaji oleh para ahli. Berikut adalah beberapa faktor yang mempengaruhi kinerja akademik siswa (Deary dkk, 2007):

1. Faktor personal, seperti kemampuan akademik, minat dan bakat, motivasi, tingkat kepercayaan diri, dan kesehatan fisik dan mental dapat mempengaruhi kinerja akademik siswa. Siswa yang memiliki kemampuan akademik yang lebih baik, memiliki minat dan bakat dalam bidang tertentu, memiliki motivasi yang tinggi, dan kesehatan fisik dan mental yang baik cenderung memiliki performa yang lebih baik.

2. Faktor lingkungan, seperti lingkungan keluarga, lingkungan sosial, dan lingkungan sekolah juga dapat mempengaruhi kinerja akademik siswa. Keluarga yang mendukung, teman sebaya yang positif, dan lingkungan sekolah yang kondusif untuk belajar dapat membantu meningkatkan performa siswa.
3. Faktor pembelajaran, seperti metode pengajaran, kualitas materi pelajaran, jumlah waktu yang dihabiskan untuk belajar, dan lingkungan belajar juga dapat mempengaruhi kinerja akademik siswa. Metode pengajaran yang efektif, materi pelajaran yang berkualitas, waktu belajar yang cukup, dan lingkungan belajar yang kondusif dapat membantu siswa dalam meningkatkan performa belajar mereka.
4. Faktor teknologi, seperti akses internet, penggunaan perangkat digital dalam pembelajaran, dan aplikasi pembelajaran online atau e-learning juga dapat mempengaruhi kinerja siswa. Akses internet yang baik dan penggunaan perangkat digital dalam pembelajaran dapat membantu siswa dalam meningkatkan keterampilan teknologi dan memperoleh informasi yang lebih banyak.
5. Faktor psikologis, seperti stres, kecemasan, depresi, dan gangguan emosional lainnya juga dapat mempengaruhi kinerja siswa. Siswa yang mengalami stres, kecemasan, atau gangguan emosional lainnya cenderung memiliki performa yang lebih rendah dibandingkan dengan siswa yang tidak mengalami hal tersebut.

Mengukur kinerja akademik siswa dalam penggunaan *e-learning*, ada beberapa cara yang dapat dilakukan (Kusuma dan Setyawan, 2020) :

1. Ujian online, dapat digunakan untuk mengukur pemahaman siswa terhadap materi yang dipelajari melalui e-learning.
2. Tugas online, dapat diberikan kepada siswa untuk menilai pemahaman terhadap materi yang dipelajari dan kemampuan dalam menerapkan konsep yang telah dipelajari. Tugas online juga dapat dijadikan sebagai bagian dari

penilaian akhir, sehinggga dapat membantu menentukan kinerja akademik siswa secara keseluruhan.

3. Observasi pengajar, pengajar dapay melakukan observasi terhadap kinerja siswa dalam penggunaan e-learning, seperti penggunaan platform e-learning dan interaksi siswa dengan sesama siswa atau pengajar. Observasi ini dapat memberikan infromasi tentang tingkat keterlibatan siswa dan kemampuan mereka dalam menggunakan teknologi untuk belajar.
4. Evaluasi diri siswa, siswa dapat diminta untuk melakukan evaluasi diri terhadap kemampuan mereka dalam menggunakan e-learning.
5. Analisis data pengguna *e-learning, platform e-learning* dapat dilengkapi dengan fitur analisis data pengguna, yang dapat memberikan informasi tentang kinerja siswa, seperti tingkat pertisipasi da penggunaan *platform e-learning.* Data ini dapat membantu pengajar dalam minilai kinerja akademik siswa secara keseluruhan.

Terdapat berbagai cara untuk mengukur kinerja akademik siswa dalam penggunaan e-learning. Pemilihan metode evaluasi yang tepat tergantu pada tujuan dan sasaran yang ingin dicapai, serta kondisi siswa dan lingkungan pebelajaran (Firman, F. dan Hasanah, 2020). Berdasarkan tujuan dari penelitan ini, dibatasi fokus menggunakan analisis data pengguna e-learning untuk mengukur kinerja akademik siswa, selama berinteraksi degan e-learning atau LMS.

### 2.2.15 *Adaptive Learning Management System* (ALMS)

Sistem pembelajaran adaptif (sering disebut lingkungan pembelajaran adaptif) bertujuan untuk mendukung pembelajar dalam memperoleh pengetahuan dan keterampilan dalam pembelajaran domain tertentu. Tujuannya adalah untuk meningkatkan proses pembelajaran individu sehubungan dengan kecepatan, ketepatan, kualitas dan kuantitas pembelajaran. Berbagai teknik adaptasi yang berbeda digunakan dalam lingkungan pembelajaran adaptif saat ini, penerapan teknik tersebut berdasarkan informasi tentang pembelajar tertentu yang disimpan dalam model pembelajaran individual. *Adaptive learning management system* mencakup berbagai adaptasi dari sistem sederhana yang mendukung beberapa

aspek adaptasi dan hanya dengan pengetahuan dasar tentang adaptasi pelajar di satu sisi untuk menguraikan lingkungan belajar seperti sistem bimbingan cerdas *(intelligent tutoring system).*

Menurut Brusilovsky (2001), metode adaptasi yang banyak digunakan dalam ALMS dapat dapat dibagi kedalam kategori *adaptive presentation, adaptive navigation support* dan *adaptive curriculum sequencing.* Ini dapat diselesaikan dengan metode *adaptive problem-solving support* (Weber dan Brusilovsky, 2001).

1. Metode *adaptive presentation* atau *content adaptation*, dengan menyesuaikan penyajian konten yang bertujuan untuk mengetahui gaya belajar berdasarkan model belajar. Sistem dengan metode *adaptive presentation*, menyajikan halaman pengguna *(user)* tidak statis, tetapi secara adaptif dihasilkan atau dirakit dari bagian-bagian yang berbeda. Misalnya, *expert user* dapat menerima informasi lebih rinci dan mendalam, sedangkan *general user* (siswa) menerima penjelasan tambahan atau berdasarkan gaya belajar pengguna dengan menyajikan lebih banyak teks atau lebih banyak gambar yang mungkin lebih disukai.
2. Metode *curriculum sequencing* yang juga disebut sebagai teknologi perencanaan instruksional. Sistem dengan metode *curriculum sequencing* menyesuaikan urutan unit pengetahuan berdasarkan tujuan pembelajarannya. Contohnya dapat merencanakan urutan tugas untuk dikerjakan. Urutan yang optimal dapat direncanakan sebelum memulai materi pembelajaran atau dapat dihitung dengan cepat selama proses belajar dengan sistem dan tergantung pada sistem dan hasil mengerjakan latihan, tes, atau tugas pemecahan masalah.
3. Metode *adaptive navigation support*, sistem dengan metode ini dapat secara adaptif mengurutkan keterangan atau menyembunyikan sebagian tautan halaman untuk menyederhanakan pilihan. *Adaptive navigation support* dapat dianggap sebagai perpanjangan dari urutan kurikulum ke dalam konteks *hypermedia*. Hal ini membantu pembelajar menemukan "jalur optimal" melalui mata pelajaran. Pada waktu bersamaan, *adaptive*

*navigation support* kurang direktif untuk membimbing siswa secara implisit dan pengetahuan berikutnya untuk dipelajari atau masalah berikutnya terpecahkan.

4. Metode *adaptive problem-solving support* ditemukan dalam *intelligent tutoring system.* Sistem ini menganalisis pemecahan masalah atau bahkan mengamati peserta didik dalam mencari solusi. Hasil analisis menggambarkan konsep atau keterampilan mana yang dipelajari oleh peserta didik yang digunakan untuk memperbarui model pembelajaran.

Sistem pembelajaran adaptif masa kini didominasi oleh hasil kognitif dalam psikologi dan pedagogi. Artinya, pengembangan sistem adaptif terutama *intelligent tutoring system* didominasi dengan penelitian tentang peran representasi mental dan proses kognitif dalam pembelajaran dengan mengabaikan pentingnya motivasi selama proses pembelajaran. Oleh karena itu, topik yang muncul dalam penelitian tentang ALMS adalah penyelidikan tentang peran platform e-learning dan motivasi dalam pembelajaran, bagaimana motivasi belajar dan perilaku belajar dapat dideteksi secara otomatis oleh sistem, dan bagaimana sistem pembelajaran dapat beradaptasi dengan psikologi keadaan peserta didik (Rumbaugh dkk, 2012).

### 2.2.16 *Learning Management System* Kalboard 360

Kalboard 360 adalah LMS multi-agen, yang telah dirancang untuk memfasilitasi pembelajaran online/daring melalui penggunaan teknologi terdepan. Sistem ini menyediakan akses sinkron ke sumber daya pendiidkan dari perangkat apa pun yang memiliki koneksi internet. Platform ini menawarkan berbagai fitur seperti manajemen kelas, manajemen tugas, penilaian online, dan kemampuan untuk membuat konten pembelajaran interaktif.

Pengajar dapat membuat dan mengatur kursus online, mengunggah materi pelajaran, membuat tugas dan ujian, serta mengirimkan umpan balik kepada siswa secara online. Sementara itu siswa dapat mengakses materi pembelajaran kapan saja dan di mana saja, mengirimkan tugas dan ujian serta berkomunkasi dengan pengajar dan siswa lain melalui fitur forum dan obrolan. Sebagai platform LMS, Kalboard

360 merekam berbagai jenis data terkai dengan pengguna dan penggunaan platform. Berikut adalah data yang terekam oleh Kalboard 360 :

1. Data pengguna : Nama pengguna, alamat email, password, dan informasi profil lainnya seperti data demografi.
2. Aktivitas pembelajaran : Data tentang interaksi pengguna dengan konten pembelajaran seperti video, gambar, dan teks, serta data tentang penyelesaian tugas dan ujian.
3. Aktivitas pengguna : Data tentang kapan pengguna masuk atau keluar dari platform, kapan dan bagaimana pengguna berinteraksi dengan pengguna lain, serta data aktivitas lainnya yang terjadi di dalam platform.

Data – data tersebut dapat diunduh dan digunakan untuk mengelol maupun mengingkatkan kinerja platform, memantau dan melaporakan akvitas pengguna, seta memberikan dukungan teknis kepada pengguna. Dataset yang digunakan pada penelitian ini adalah data kinerja akademik siswa, yang telah dikumpulkan oleh Amrieh, dkk yang diambil menggunakan xAPI pada LMS Kalboard 360. Validitas dari dataset yang digunakan atau hasil dari rekaman database Kalbord 360 perlu dilakukan untuk memastikan bahwa kumpulan data tersebut akurat dan dapat diandalkan untuk analisis lebih lanjut (Kim, Kim dan Bonk, 2006). Langkah – langkah validasi pada data hasil LMS adalah sebagai berikut :

1. Memeriksa kelengkapan dan konsistensi data, periksa apakah ada data yang hilang atau duplikat dalam kumpulan data.
2. Verivikasi keakuratan data, bandingkan data tersebut dengan sumber data lain, seperti buku nilai atau dokumen lainnya.
3. Lakukan analisis statistik, gunakan statistik deksriptf untuk mengidentiifkasi rata – rata, median, dan standar deviasi dari setiap variabel kinerja akademik siswa.
4. Uji reliabilitas dan validitas, pengujian reliabilitas dapat membandingkan konsistensi setiap variabel kinerja siswa. Untuk menguji validitas dapat membandingkan variabel-variabel tersebut dengan variabel lain atau data

lain yang relevan untuk memastikan bahwa data tersebut benar – benar mempresentasikan kinerja siswa.

5. Evaluasi kualitas data, mengevaluasi kualitas data dengan menilai relevansi, ketepatan waktu, keakuratan, kelengkapan, dan konsistensi data tersebut.

Langkah-langkah tersebut dapat memvalidasi kumpulan data kinerja akademik siswa dan memastikan bahwa data tersebut dapat digunakan untuk analisis (Nwachukwu, 2018). Tahapan ini dijelaskan lebih lanjut pada BAB 3 terakit pengumpulan data.