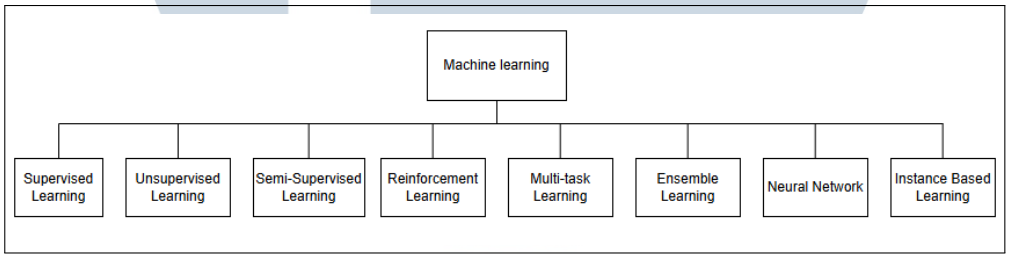
BAB 2
LANDASAN TEORI

2.1 *Machine Learning*

Machine learning (ML) adalah cabang *artificial intelligence* (AI) yang memungkinkan sistem komputer untuk mempelajari data dan dapat membuat prediksi atau melakukan instruksi spesifik tanpa diprogram secara eksplisit. ML memiliki beragam tipe algoritma yang digunakan untuk menyelesaikan suatu masalah dan pemilihan algoritma yang tepat bergantung dari jenis masalah, jumlah variabel, karakteristik data dan berbagai faktor lainya [22]. Gambar 2.1 menunjukan tipe-tipe algoritma ML.



Gambar 2.1. Tipe-tipe algoritma ML
Sumber: [22]

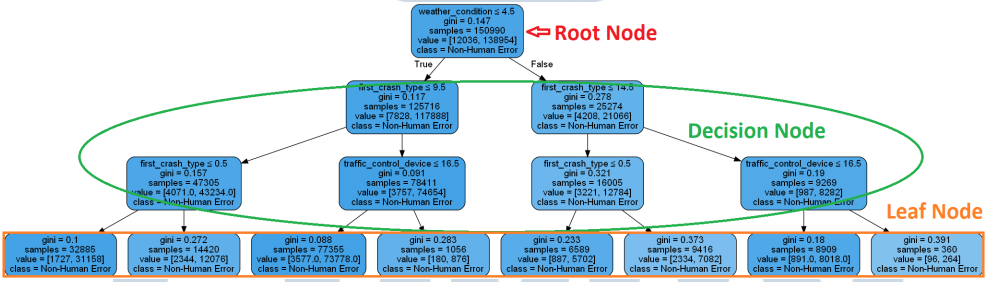
Pada penelitian ini digunakan algoritma *ensemble learning*, yaitu *random forest* (RF) dengan metode *bagging* dan XGB yang menerapkan metode *boosting*. Pendekatan *ensemble* dipilih karena mampu meningkatkan akurasi prediksi dibandingkan algoritma tunggal dengan cara menggabungkan beberapa pohon keputusan untuk mengurangi *variance* pada *bagging* maupun bias pada *boosting*, sehingga menghasilkan model yang lebih stabil dalam menangani data yang kompleks seperti pada kasus prediksi jenis kecelakaan lalu lintas.

2.2 *Decision Trees*

Decision tree (DT) merupakan alat klasifikasi yang digunakan untuk memisahkan data ke dalam kategori tertentu dengan menerapkan kondisi-kondisi spesifik dalam proses pengambilan keputusan [23]. Algoritma ini berbasis struktur pohon, di mana setiap jalur yang dimulai dari akar (*root node*) hingga ke daun (*leaf*

node) merepresentasikan urutan pemisahan data yang pada akhirnya menghasilkan keputusan *boolean*. Setiap proses pemisahan dalam pohon melibatkan pertanyaan logis yang memecah data berdasarkan titik pemisahan (*split point*), yang umumnya berupa nilai diskret.

Setiap *decision node* akan menampilkan kelas target dengan jumlah data terbanyak dalam distribusi pada *node* tersebut. Pembentukan struktur pohon melibatkan perhitungan nilai impuritas, yaitu ukuran ketidakhomogenan data dalam suatu *node*. Setelah proses pemisahan dilakukan, nilai impuritas dihitung untuk masing-masing *leaf node*. Total impuritas dari satu proses pemisahan kemudian ditentukan melalui rata-rata tertimbang dari nilai impuritas pada setiap *leaf node*. Nilai impuritas dari berbagai kemungkinan splitting akan dibandingkan, dan pemisahan yang menghasilkan nilai impuritas terendah akan dipilih sebagai kandidat terbaik. Proses ini berlangsung secara rekursif pada setiap *node* dalam pohon, hingga diperoleh *leaf node* yang semakin "murni" atau homogen terhadap variabel target. DT merupakan model *single classifier* yang digunakan untuk membangun model *ensemble* RF dan XGB. Gambar 2.2 menunjukan struktur dan visualisasi *decision tree* pada dataset yang digunakan *decision tree* sekaligus.



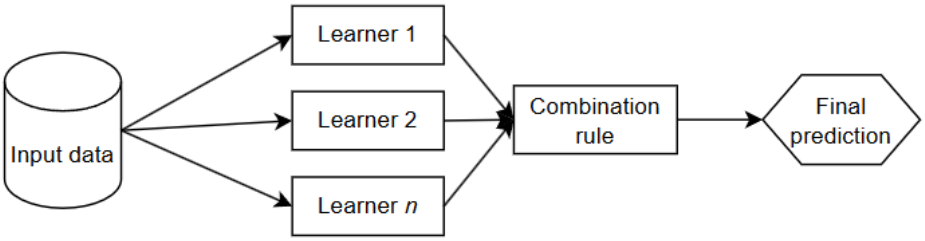
Gambar 2.2. Visualisasi DT

Sumber: [23]

2.3 *Ensemble Learning*

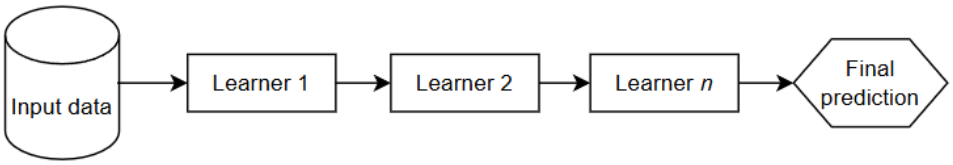
Ensemble learning adalah metode ML yang memanfaatkan beberapa model dasar atau model tunggal (*base models*) untuk meningkatkan performa dengan menggabungkan hasil prediksi masing-masing model [24]. Meskipun terdiri dari beberapa model tunggal, metode ensemble learning berfungsi sebagai satu kesatuan model dalam menghasilkan keputusan akhir [25]. *Ensemble learning* dibagi menjadi dua tipe utama yaitu [26]:

1. *Bagging (boostrap aggregating)*, merupakan metode di mana setiap model dilatih secara independen pada subset data yang berbeda, kemudian hasil prediksi dari masing-masing model digabungkan melalui rata-rata (untuk regresi) atau voting mayoritas (untuk klasifikasi) sehingga meningkatkan performa dan akurasi model. Metode *bagging* dapat dilihat pada Gambar 2.3.



Gambar 2.3. Metode bagging
Sumber: [24]

2. *Boosting*, merupakan metode di mana model dilatih secara berurutan dimana setiap model baru akan berusaha memperbaiki kesalahan yang dibuat oleh model sebelumnya [26]. Gambar 2.4 menunjukan proses metode boosting.



Gambar 2.4. Metode boosting
Sumber: [24]

2.4 *Random Forest*

Random forest (RF) merupakan algoritma *ensemble* dengan tipe *bootstrap aggregating* yang terdiri dari gabungan beberapa *decision tree* yang dibangun dari

subset data yang dipilih secara acak. Setiap *decision tree* bekerja secara independen dan hasil prediksi akhir merupakan hasil agregasi dari keseluruhan *decision tree* yang tersedia [27]. Pendekatan ini mengurangi *overfitting* dan variansi model sehingga meningkatkan akurasi dan kestabilan prediksi. Algoritma RF adalah sebagai berikut[28]:

1. untuk setiap b=1 hingga B:
   (a) ambil sampel *boostrap* Z^* berukuran N dari data latih
   (b) Bangun *decision tree* T_b menggunakan data *bootstrap* Z^* dengan aturan: Pada setiap *terminal node*,
      (i) Pilih secara acak variabel m dari total variabel p yang tersedia.
      (ii) Pilih fitur terbaik beserta titik *split* diantara m.
      (iii) Bagi *node* tersebut menjadi dua cabang.
      (iv) Ulangi sampai ukuran minimal *node* tercapai.
2. *Output* berupa gabungan dari *decision trees* $\{T_b\}_1^B$.
3. Untuk melakukan prediksi pada data uji x:
   (a) Setiap *decision tree* T_b memberikan prediksi kelas $\widehat{C}_b(x)$.
   (b) Kelas akhir ditentukan berdasarkan metode *majority vote*.

$$\widehat{C}_{rf}^B(x) = majority\ vote\{\widehat{C}_b(x)\}_1^B. \tag{2.1}$$

Pengertian notasi pada rumus tersebut adalah sebagai berikut [28]:

B = jumlah total *decision tree* yang akan dibangun.

b = index pohon

Z^* = sampel *bootstrap* dari data latih.

N = jumlah data latih.

p = variabel independen (fitur).

m = jumlah fitur yang dipilih acak saat *splitting*.

T_b = *decision tree* ke-b.

$\widehat{C}_b(x)$ = prediksi dari T_b untuk data uji x

$\widehat{C}_{rf}^B(x)$ = prediksi akhir dengan voting mayoritas.

2.5 *XGBoost*

XGB atau *extreme gradient boosting* merupakan algoritma ensemble kategori *boosting* yang menggunakan *decision tree* sebagai model dasarnya. Setiap *decision tree* akan dibangun berurutan dimana setiap pohon baru dilatih untuk memperbaiki kesalahan yang dibuat oleh pohon sebelumnya [29]. Untuk menentukan *split* terbaik dalam *decision tree*, model XGB menggunakan rumus sebagai berikut[29] [24]:

$$\frac{1}{2}[\frac{(\sum gL)^2}{\sum h_L+\lambda}+\frac{(\sum gR)^2}{\sum h_R+\lambda}-\frac{(\sum g)^2}{\sum h+\lambda}]-\gamma \tag{2.2}$$

gL dan gR secara berurutan adalah gradien dari sampel sisi kiri dan kanan *split*.

hL dan hR secara berurutan adalah hessian dari sampel sisi kiri dan kanan *split*

λ adalah parameter untuk mencegah model menjadi terlalu kompleks.

γ adalah parameter untuk memberi penalti pada *split* bernilai rendah.

2.6 *Confusion Matrix*

Confusion matrix adalah sebuah tabel yang digunakan untuk mengevaluasi kinerja algoritma klasifikasi [30]. Matrix ini menunjukan performa masing-masing model dengan mengukur nilai-nilai seperti *true positive, true negative, false positive* dan *false negative*. Tabel 2.1 menunjukan bentuk dari *confusion matrix*.

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

Tabel 2.1. Confusion Matrix

Pengertian dari masing masing nilai yang terdapat pada confusion matrix adalah sebagai berikut:

TP (*True Positive*) berarti hasil prediksi model positif dan hasil sebenarnya juga positif.

FP (*False Positive*) berarti hasil predisksi positif namun, hasil sebenarnya negatif.

TN (*True Negative*) berarti hasil prediksi dan hasil sebenarnya negatif.

FN (*False Negative*) berarti hasil prediksi negatif namun, hasil sebenarnya positif.

Performa algoritma dapat dihitung menggunakan nilai yang didapatkan dari *confusion matrix*. Metrik perhitungan yang digunakan untuk mengukur peforma antara lain, *Accuracy*, *precision*, *recall* dan *F1-score*. Berikut adalah rumus dari masing-masing metrik:

$$Accuracy = \frac{(TP+TN)}{(TP+FP+FN+TN)} \tag{2.3}$$

Accuracy mengukur tingkat keakuratan model dengan menghitung rasio antara jumlah prediksi yang benar terhadap keseluruhan data.

$$Precision = \frac{TP}{TP+FP} \tag{2.4}$$

Precision mengukur rasio prediksi positif yang benar terhadap seluruh prediksi positif.

$$Recall = \frac{TP}{TP+FN} \tag{2.5}$$

Recall Menunjukkan seberapa banyak data positif yang berhasil diidentifikasi dengan benar dari total data positif yang ada.

$$F1Score = \frac{2 * precision * recall}{precision + recall} \tag{2.6}$$

F1 score mengukur keseimbangan antara *precision* dan *recall*. Metrik ini digunakan untuk menilai menilai efektivitas model dalam mendeteksi data positif.