BAB 2
LANDASAN TEORI

2.1 Obesitas

Obesitas telah diperkirakan akan menjadi masalah kesehatan utama di dunia dunia dan masalah kesehatan serius di negara-negara berkembang [12]. Berdasarkan laporan dari *World Health Organization* (WHO) tentang Diet, Nutrisi, dan Pencegahan Penyakit Kronis, telah diketahui bahwa obesitas merupakan faktor risiko utama dari seluruh penyakit tidak menular [12].

Prevalensi obesitas di Indonesia terus meningkat dari tahun ke tahun [13]. Berdasarkan studi yang dilansir dari Badan Penelitian dan Pengembangan Kesehatan tahun 2019, diketahui bahwa, di Indonesia tingkat prevalensi obesitas pada orang dewasa terus meningkat, sebesar 8.6% pada tahun 2007, sebesar 11.5% pada 2013, dan sebesar 13.6% pada 2013 [13].

Pada akhirnya, obesitas telah menjadi masalah kesehatan utama di Indonesia sejak beban ganda penyakit mempengaruhi penduduk [12]. Hal ini dikarenakan, infeksi sendiri masih menjadi penyebab utama dari masalah-masalah kesehatan dan kematian. Di sisi lain, penyakit-penyakit tidak menular seperti obesitas, diabetes, dan kanker juga semakin meningkat [12] yang menuntut perhatian khusus dalam upaya pencegahan dan penanganannya.

2.2 Pembelajaran Mesin

Pembelajaran mesin merupakan salah satu cabang dari *Artificial Intelligence* (AI) yang memungkinkan sebuah sistem/ aplikasi untuk dapat belajar secara mandiri melalui data dan menjadi lebih handal dan akurat dalam memprediksi sesuatu tanpa campur tangan manusia [14]. Hal ini dikarenakan pembelajaran mesin dibuat untuk meniru cara kerja otak manusia [14]. Pembelajaran mesin sendiri terbagi menjadi beberapa jenis, yaitu pembelajaran terbimbing (*supervised learning*), pembelajaran tidak terbimbing (*unsupervised learning*), dan pembelajaran berbasis *trial* dan *error* (*reinforcement learning*) [15]. Berikut adalah penjelasan untuk masing-masing jenis pembelajaran mesin tersebut.

2.2.1 *Supervised Learning*

Supervised learning merupakan jenis pembelajaran mesin yang digunakan ketika terdapat *output*/ kelas dari sebuah *dataset* [15]. *Output* yang diberikan adalah kelas dan data-data lainnya adalah fitur. Salah satu prasyarat untuk model ini adalah bahwa kelas yang diberikan harus benar/ akurat [15]. Hal ini dikarenakan, model akan dilatih berdasarkan kelas dari data yang diberikan [15]. Oleh karena itu, performa model juga akan berkurang jika kelas yang diberikan salah.

Tahapan dasar pada *supervised learning* adalah (1) membagi *dataset* menjadi 2 bagian, yaitu data pelatihan dan data uji [16]; (2) melakukan pelatihan model menggunakan data latih untuk menginformasikan hubungan antara setiap fitur dengan kelas [16]; (3) melakukan evaluasi model menggunakan data uji untuk menentukan seberapa baik model melakukan tugasnya [16]. Salah satu metode yang populer digunakan dalam *supervised learning* adalah klasifikasi [16].

Klasifikasi sendiri merupakan proses melakukan pembelajaran algoritma melalui data-data yang diberikan lalu mengklasifikasikannya ke dalam kelas-kelas [14]. Metode klasifikasi ini juga terbagi lagi ke dalam 2 tipe, yaitu *dual* dan *multiple class classification* [14]. Pada *dual classification*, hasil klasifikasi akan berbentuk biner, seperti benar/ salah atau ya/ tidak [14]. Sementara itu, pada *multiple class classification*, hasil klasifikasinya akan menghasilkan lebih dari 2 kemungkinan, seperti tinggi/ sedang/ rendah atau baik/ sedang/ buruk [14]. Salah satu algoritma klasifikasi terbaik yang sering digunakan adalah *Random Forest* [14].

2.2.2 *Unsupervised Learning*

Unsupervised learning merupakan jenis pembelajaran mesin yang tidak menggunakan kelas/ *output* dalam melakukan pembelajaran [15]. Oleh karena itu, algoritma ini memiliki kemampuan untuk menyusun pola dari data tanpa kelas serta mengidentifikasi pola yang tidak umum dalam *dataset* [15].

Beberapa teknik yang umumnya digunakan dalam *unsupervised learning* adalah *clustering* dan *association* [16]. *Clustering* adalah proses pengelompokkan data ke dalam kelompok/ *cluster* [17]. Sementara itu, *association* adalah proses untuk mengetahui hubungan antar *variable* dalam sebuah *dataset* [17].

2.2.3 *Reinforcement Learning*

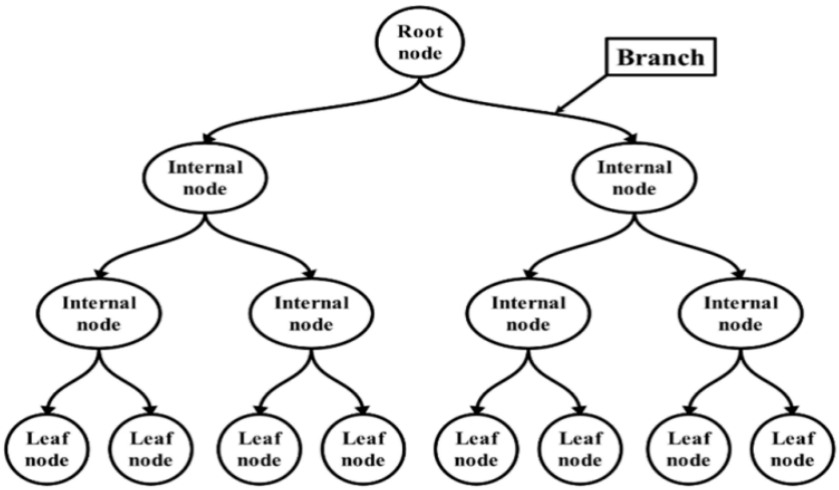
Reinforcement learning merupakan jenis pembelajaran mesin yang paling menyerupai cara manusia belajar karena tidak hanya mengandalkan pembelajaran dari data, tetapi juga melibatkan *trial* dan *error* dari pengalaman langsung dalam proses pembelajaran [16]. Dari proses *trial* dan *error* tersebut, sistem akan memperoleh *feedback* untuk dipelajari dan diterapkan dalam *trial* dan *error* yang berikutnya [15].

Reinforcement learning sendiri dapat dibedakan menjadi 2 jenis, yaitu *on-policy* dan *off-policy learning* [18]. Pada *on-policy learning*, terdapat SARSA (*state-action-reward-state-action*) yang melakukan pembelajaran dan evaluasi atas tindakan yang sedang diambil [18]. Sementara itu, pada *off-policy learning* terdapat *Q-learning* yang melakukan evaluasi tanpa bergantung pada tindakan saat ini [18]. Hal ini dikarenakan, *Q-learning* akan selalu mengevaluasi seluruh kemungkinan tindakan saat ini untuk mengetahui tindakan mana yang dapat memberikan *reward* yang paling maksimal pada langkah berikutnya [18].

2.3 *Decision Tree*

Decision tree learning merupakan pendekatan model prediktif yang biasanya digunakan dalam bidang statistik, *data mining*, dan pembelajaran mesin [19]. Pada pembelajaran mesin, algoritma ini sering digunakan untuk klasifikasi karena tidak memerlukan informasi yang besar [20]. Algoritma ini berbentuk seperti pohon yang terdiri dari *subdivision* yang membagi data kedalam *subset* yang lebih kecil berdasarkan kriteria tertentu secara berulang [1].

Decision tree ini dibangun dengan menggunakan *nodes* dan *branches* [1], dimana *nodes* tersebut terbagi lagi menjadi *internal* dan *leaf nodes*. Pembangunan pohon akan selalu diawali dengan *root/ parent node* yang terletak di puncak pohon [1]. Anak cabang dari *root* yang masih memiliki cabang anakan disebut dengan *internal nodes* yang dihubungkan kepada *root*, *internal nodes* lainnya, dan *leaf nodes* dengan *branch* [1]. *Leaf nodes* sendiri merupakan sebuah cabang yang tidak memiliki cabang lagi yang merepresentasikan hasil keputusan dari *decision tree*. Gambar 2.1 merepresentasikan struktur dari *decision tree* [1].



Gambar 2.1. Struktur *Decision Tree* [1]

2.4 *Random Forest*

Random Forest adalah salah satu algoritma *ensemble* [21] dari *supervised learning* yang merupakan pengembangan dari *decision tree*. Hal ini dikarenakan, algoritma ini bekerja dengan membangun beberapa *decision tree* untuk melakukan prediksi dan mengambil keputusan akhir berdasarkan hasil *voting* [22]. Prinsip dasar algoritma ini adalah menggabungkan sejumlah *estimator* yang lemah untuk membentuk satu *estimator* yang kuat dengan mengintegrasikan hasil prediksi dari seluruh *decision tree* yang terbentuk melalui hasil *voting* terbanyak [21]. Berikut adalah tahapan pembuatan model *random forest* [23].

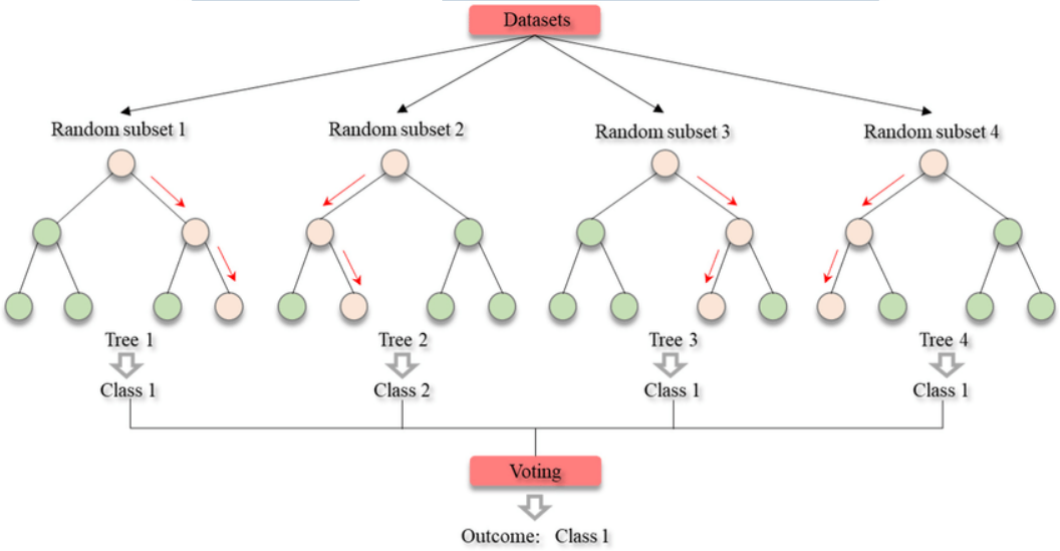
1. Memilih n sampel acak dari data *training* untuk digunakan dalam proses *bootstrap* yang bertujuan untuk membuat *dataset* yang lebih bervariasi.

2. Menggunakan data *training* yang dihasilkan dari proses *bootstrap* untuk membangun *decision tree* dengan memilih fitur secara acak dengan ukuran $m < p$, dimana p adalah jumlah total fitur pada *dataset*. Nantinya, pada setiap *nodes* dalam setiap *decision tree* yang terbentuk, akan dipilih fitur terbaik sebagai pemisah untuk membagi data menjadi 2 *child nodes* baru dengan kriteria pengoptimalan pemisahan data, seperti ”gini”, ”entropy”, atau ”log_loss”.

3. Proses pembangunan *decision tree* akan terus berlanjut hingga jumlah minimum observasi pada *node* tercapai.

4. Seluruh proses di atas akan diulang sebanyak k kali untuk menghasilkan k *decision tree*. Dari setiap *decision tree*, akan didapatkan 1 kelas yang akan digunakan dalam proses *voting* untuk menentukan kelas klasifikasi akhir.

Lalu, gambaran/ ilustrasi dari cara kerja algoritma *random forest* dapat dilihat pada Gambar 2.2.



Gambar 2.2. Cara Kerja Algoritma *Random Forest* [2]

Lalu, berikut adalah rumus menghitung algoritma *Random Forest* untuk mendapatkan hasil akhir klasifikasi. $\hat{C}_b(x)$ merupakan kelas prediksi dari pohon ke-b untuk data x [24], yang akan digunakan dalam proses *voting* dengan menggunakan hasil klasifikasi dari pohon ke-1 hingga pohon ke-B untuk mendapatkan hasil akhir klasifikasi berdasarkan hasil *majority vote*.

$$\hat{C}_{\text{rf}}^{B}(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B \tag{2.1}$$

Namun, algoritma ini pada dasarnya tidak dapat menghasilkan performa yang optimal jika hanya mengandalkan 1 *value* untuk setiap *hyperparameter* yang digunakan karena belum tentu merupakan nilai optimal [25]. Oleh karena itu, dapat dikatakan bahwa performa dari *random forest* sangat bergantung pada *hyperparameter* yang digunakan [25]. Untuk mengatasi kelemahan tersebut, dapat dilakukan *hyperparameter tuning* yang berfungsi untuk menemukan kombinasi *hyperparameter set* yang paling optimal yang didapatkan melalui pengujian *trial* dan *error* dari kombinasi seluruh *hyperparameter* [25]. Nantinya, akan dipilih

kombinasi *hyperparameter value* yang menghasilkan performa terbaik untuk digunakan pada pembangunan model.

2.5 *Randomized Search*

Randomized Search atau dapat disebut dengan algoritma stokastik merupakan salah satu teknik *hyperparameter tuning* yang menggunakan keacakan atau probabilitas [26]. Algoritma ini bekerja tanpa mengikuti pola/ urutan tertentu, melainkan menjelajahi ruang pencarian secara acak dengan hanya menggunakan beberapa pengaturan *hyperparameter* yang diambil secara acak [26]. Hal ini memungkinkan algoritma *Randomized Search* untuk melakukan pencarian ruang parameter secara efisien tanpa harus menguji semua nilai parameter. Dengan demikian, *randomized search* mampu untuk mengevaluasi beberapa kombinasi yang dianggap penting dengan cepat untuk mendapatkan hasil yang optimal/ mendekati optimal, sehingga waktu komputasi pun menjadi lebih efisien.

Algoritma *randomized search* ini akan dievaluasi menggunakan metode validasi silang [26] yang akan membagi data kedalam k *subset* yang disebut *fold*. Proses evaluasi dilakukan sebanyak k kali dengan menggunakan 1 *fold* sebagai data validasi untuk menguji model setelah dilatih menggunakan k-1 *fold* lainnya sebagai data *training*. Berikut adalah *pseudocode* untuk menjelaskan gambaran alur kerja dari *randomized search* [26].

Algorithm 1: Algoritma *Randomized Search*

Data: *Input data*
Result: *Output result*
Inisialisasi *hyperparameter* ”criterion”, ”max_depth”, ”n_estimators”, ”max_features”, ”min_samples_split”, ”max_leaf_nodes”;
Inisialisasi *estimator*, ruang pencarian, serta jumlah iterasi dan *k-fold*;
while *Kriteria berhenti belum terpenuhi* **do**
    Pilih parameter secara acak dalam ruang pencarian;
    Bagi dataset menjadi *K-Folds* secara merata;
    for *setiap fold k dalam K-Fold* **do**
        Tetapkan *fold* k sebagai data uji dan sisanya sebagai data latih;
        Latih data menggunakan *estimator* dengan kombinasi *hyperparameter*;
        Evaluasi kinerja model ;
        Hitung skor rata-rata yang diperoleh dari setiap *fold*;
    end
end
Kembalikan *hyperparameter* terbaik;

2.6 *Confusion Matrix*

Confusion matrix adalah sebuah metrik yang dapat digunakan untuk menampilkan performa dari hasil klasifikasi [27]. *Confusion matrix* juga memiliki kelebihan dalam kemudahan interpretasi [27]. Hal ini dikarenakan, baik atau buruknya performa model dapat langsung dilihat dari hasil distribusi *value* pada *cells* di *confusion matrix*. Model yang baik hanya akan memiliki *value* pada *cell* pada garis diagonal [27]. Sebaliknya, *value* akan terdistribusi/ tersebar pada seluruh *cell* [27]. Berdasarkan jumlah kelasnya, *confusion matrix* terbagi menjadi 2 jenis, yaitu *binary* dan *multi-class classification* [28].

Pada *binary classification*, *confusion matrix* berukuran 2x2 dengan label positif dan negatif untuk setiap kelas aktual dan hasil klasifikasi [19]. Setiap *cell* merepresentasikan hasil antara kelas aktual dengan hasil klasifikasi yang didapatkan [27]. Terdapat 4 istilah yang digunakan sebagai representasi hasil klasifikasi, yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN) [19]. *True positive* merepresentasikan jumlah hasil klasifikasi yang di prediksi positif oleh model sesuai dengan kenyataan [29].

True negative merepresentasikan jumlah hasil klasifikasi yang di prediksi negatif oleh model sesuai dengan kenyataan [29]. *False positive* merepresentasikan jumlah hasil klasifikasi yang di prediksi positif oleh model tidak sesuai dengan kenyataan karena seharusnya diklasifikasikan sebagai negatif [29]. *False negative* merepresentasikan jumlah klasifikasi yang di prediksi negatif oleh model tidak sesuai dengan kenyataan karena seharusnya diklasifikasikan sebagai positif [29]. Tabel 2.1 menampilkan representasi hasil *confusion matrix* untuk klasifikasi biner.

Tabel 2.1. *Confusion Matrix* untuk *Binary Classification*

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

Dari hasil *confusion matrix*, dapat dilakukan perhitungan akurasi dan beberapa metrik evaluasi lainnya, yaitu *precision*, *recall/ sensitivity*, dan *f1-score*. Berikut adalah rumus untuk menghitung akurasi yang berfungsi untuk mengukur seberapa baik model melakukan prediksi dengan benar [30].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.2}$$

Selanjutnya, berikut adalah rumus untuk menghitung *precision* yang berfungsi untuk mengukur kemampuan model dalam memprediksi kelas positif dari seluruh hasil prediksi positif [30]. Dengan kata lain, *precision* ini berfungsi untuk mengevaluasi seberapa sedikit kelas negatif yang salah diklasifikasikan sebagai kelas positif.

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

Lalu, berikut adalah rumus untuk menghitung *recall* yang berfungsi untuk mengukur kemampuan model dalam memprediksi kasus positif yang sebenarnya yang dinyatakan dengan perbandingan antara jumlah prediksi kelas positif dengan jumlah kasus positif yang sebenarnya [30]. Dengan kata lain, *recall* ini berfungsi untuk mengevaluasi seberapa sedikit kelas positif yang salah diklasifikasikan sebagai kelas negatif.

$$Recall = \frac{TP}{TP + FN} \tag{2.4}$$

Terakhir, berikut adalah rumus untuk menghitung *f1-score* yang berfungsi untuk mengukur keseimbangan antara *precision* dan *recall* untuk memberikan

gambaran lengkap tentang kinerja model dalam memprediksi kelas positif dengan memperhitungkan kedua metrik [30].

$$F1 - Score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.5}$$

Sementara itu, pada *multi-class classification*, dimensi dari *confusion matrix* adalah nxn dengan n adalah jumlah kelas dan n $>$ 2. Oleh karena itu, pada *multi-class classification confusion matrix* karakterisasi TP, TN, FP, dan FN tidak berlaku [19]. Akan tetapi, analisis untuk hasil klasifikasi masih dapat dilakukan dengan memfokuskan analisis pada sebuah kelas seperti pada Tabel 2.2 ini dengan kelas C_2 sebagai fokus utama analisis [19].

Tabel 2.2. *Confusion Matrix* untuk *Multi-class Classification*

	Predicted C_1	***Predicted*** C_2	**...**	***Predicted*** C_N
Actual C_1	$C_{1,1}$	FP	...	$C_{1,N}$
Actual C_2	FN	TP	...	FN
...	...	...	...	...
Actual C_N	$C_{N,1}$	FP	...	$C_{N,N}$

True positive (TP) menandakan kelas-kelas yang benar diklasifikasikan sebagai kelas C_2 [19]. *True negative* (TN) menandakan kelas-kelas yang benar diklasifikasikan sebagai bukan kelas C_2 [19]. *False positive* (FP) menandakan kelas-kelas yang termasuk dalam kelas C_2 pada *predicted class* dan tidak termasuk dalam *actual class* C_2 [19], atau yang salah diklasifikasikasikan sebagai kelas C_2. *False negative* (FN) menandakan kelas-kelas yang termasuk dalam kelas C_2 pada *actual class* dan tidak termasuk dalam *predicted class* C_2 [19], atau yang salah diklasifikasikan sebagai bukan kelas C_2.

Untuk menghitung akurasi dari hasil klasifikasi dengan *confusion matrix* untuk *multi-class classification*, dapat digunakan rumus berikut ini [19].

$$Akurasi = \frac{\sum_{i=1}^{N} TP(C_i)}{\sum_{i=1}^{N} \sum_{j=1}^{N} C_{i,j}} \tag{2.6}$$