BAB 2
LANDASAN TEORI

2.1 Covid-19

Coronavirus diseases atau yang lebih dikenal dengan Covid-19 merupakan virus yang ditemukan di kota Wuhan, China, pada tanggal 31 Desember 2019. Virus ini merupakan anggota Coronaviridae, yang juga memiliki kemiripan sebesar 79% dengan penyakit SARS atau *Severe Acute Respiratory Syndrome* (SARS) dan memiliki kemiripan sebesar 50% dengan penyakit MERS-CoV atau *Middle East Respiratory Syndrome coronavirus* [14]. Virus Corona umumnya menyerang area pernapasan, sehingga menyebabkan gangguan pernapasan ringan, sampai dapat menyebabkan kematian yang dikarenakan kegagalan fungsi organ tubuh.

Penyebaran Covid-19 berlangsung sangat cepat, sehingga dapat mengancam hidup tiap individu, yang tidak terbatas aspek geografi maupun usai. Hal tersebut didukung oleh mudahnya cara penyebaran virus Corona, yaitu, dapat menyebar melalui cairan tubuh manusia, udara, atau sentuhan dengan benda yang telah terkontaminasi dengan virus tersebut [4]. Bahkan virus tersebut bisa bertahan selama total waktu 72 jam pada benda-benda yang berbahan dasar plastik, 24 jam pada benda-benda yang berbahan dasar kardus, 4 jam pada benda-benda yang berbahan dasar tembaga, dan 3 jam di udara (Emma., 2020). Sampai saat ini (9 Februari 2022) jumlah kasus Covid-19 diseluruh dunia mencapai 396.558.014 kasus, dengan kasus total kematian berjumlah 5.745.032 jiwa [3].

Adapun gejala-gejala yang dialami oleh individu yang terpapar Covid-19 awalnya bersifat ringan yang kemudian muncul secara bertahap [15]. Bahkan, gejala-gejala yang dialami oleh penderita Covid-19 memiliki kemiripan dengan gejala-gejala penyakit lain seperti penyakit flu, pneumonia, demam berdarah, dan tipes [16]. Gejala-gejala umum yang dirasakan penderita Covid-19 berupa pilek, batuk, sakit kepala, demam, serta kehilangan kemampuan serta kehilangan kemampuan untuk mengenali bau dan rasa. Namun terdapat gejala-gejala yang tidak umum yang oleh penderita Covid-19, yaitu; ruam kulit, konjugtivitis, panas dingin, pusing, cepat marah, cemas, atau depresi, serta gangguan tidur [6].

2.2 Ensemble Learning

Ensemble Learning merupakan teknik yang mengambungkan (ensemble) beragam metode klasifikasi dengan tujuan untuk mendapatkan solusi terbaik dari permasalahan yang kompleks [7]. Perbedaan yang paling signifikan dari pendekatan pembelajaran individual adalah pembelajaran individual hanya mempelajari satu hipotesis dari data yang telah di latih, sedangkan pendekatan ensemble berusaha untuk menggabungkan beberapa hipotesis yang telah dibangun menjadi satu kesatuan. Terdapat 3 metode *Ensemble Learning*, yaitu [17]:

1. *Bagging*:

   Bagging juga dikenal sebagai *bootstrap aggregating*, yaitu proses yang menggunakan beberapa model dari algoritma yang sama dan melatih setiap model pada sampel yang berbeda dari data set yang sama. Prediksi yang telah dibuat oleh setiap model selanjutnya digabungkan dan divoting atau di rata-ratakan. Salah dua contoh algoritma yang menggunakan metode bagging adalah *Random Forest* dan *Extra Trees*

2. *Boosting*:

   Boosting merupakan teknik yang digunakan untuk mengurangi bias dan varian. Setiap model yang dibangun dapat menggunakan hasil dari model yang sebelumnya, yang mana jika terdapat kesalahan pada prediksi sebelumnya maka dapat diperbaiki dalam model selanjutnya. Salah satu contoh algoritma yang menggunakan metode *boosting* adalah *Gradient Boost Machine* (GBM)

3. *Stacking*:

   Stacking merupakan teknik yang melibatkan berbagai *fitting* dari banyaknya tipe model pada data yang sama dan kemudian menggunakan model yang lain untuk mempelajari cara terbaik untuk menggabungkan prediksi. Salah dua contoh algoritma yang menggunakan metode *stacking* adalah *Blending* dan *Super Ensemble*

2.3 Decision Tree

Decision Tree merupakan algoritma pada *machine learning* untuk membuat suatu prediksi dengan bentuk pohon. Implementasi *Decision Tree* seringkali untuk

klasifikasi berdasarkan atribut yang diberikan, dengan mempertimbangkan nilai pada atribut tersebut, yang kemudian merujuk pada class yang dijadikan hasil prediksi. *Decision Tree* bisa diibaratkan seperti if...then... pada pemrograman, dan dapat di gambarkan menjadi sebuah pohon keputusan. Terdapat struktur–struktur yang membangun *Decision Tree* [18], yaitu:

1. Node Akar atau *root node*, adalah node yang terdapat di paling atas (paling awal) dari tree, yang merepresentasikan keseluruhan populasi.
2. Node Internal atau *internal node*, adalah node yang memiliki percabangan sehingga menjadi dua atau lebih sub node.
3. Node Daun atau *leaf node*, adalah node terakhir dari percabangan, dimana tidak ada lagi cabang setelah node ini.
4. Lengan atau *branch* merupakan parameter atau rule dari suatu *tree*.
5. *Sub Tree* adalah node yang memiliki percabangan, yang membuat tree baru.

Cara kerja dari pada *Decision Tree* sendiri adalah dengan menentukan root node dan *internal node* dari fitur yang diberikan terlebih dahulu. Untuk pencarian atribut yang akan digunakan untuk *decision node* dapat dilakukan dengan cara *information gain* atau dengan cara *gini index*. *Information gain* adalah cara dimana nilai dari suatu informasi dapat ditentukan atau dihitung, untuk setiap atribut. Tahap awal dari information gain adalah dengan mencari nilai *entropy* dari setiap atribut fitur dan atribut target. jika suatu fitur memiliki nilai *entropy* 0, maka fitur tersebut akan menjadi *leaf node*. Kemudian untuk mendapatkan *information gain*, maka dilakukan pengurangan dari *entropy* fitur dan *entropy* atribut yang telah didapat tadi. Nantinya Information Gain yang tertinggi akan dijadikan *root node*. Adapun formula yang digunakan untuk mencari nilai *entropy*, *information gain*, serta *gini index* dapat dilihat pada Persamaan 2.1, 2.2, dan Persamaan 2.3 [19].

$$Entropy(S) = -\sum_{i=1}^{k} p_i \, log_2 \, p_i \tag{2.1}$$

Keterangan:

- S = Dataset S.
- p_i = Nilai probabilitas dari *event* atau kelas i.

- k = Banyak partisi S.

$$InformationGain(A) = Entropy(S) - \sum_{i=1}^{k} \frac{|S_i|}{S} \times Entropy(S_i) \tag{2.2}$$

Keterangan:

- $InforamtionGain$ = Nilai dari *Information Gain*.
- $Entropy(S)$ = Nilai *entropy* dari S.
- S_i = Subkelompok yang dibentuk berdasarkan kumpulan data dari S.
- $Entropy(S_i$ = Entropi subkelompok S_i.

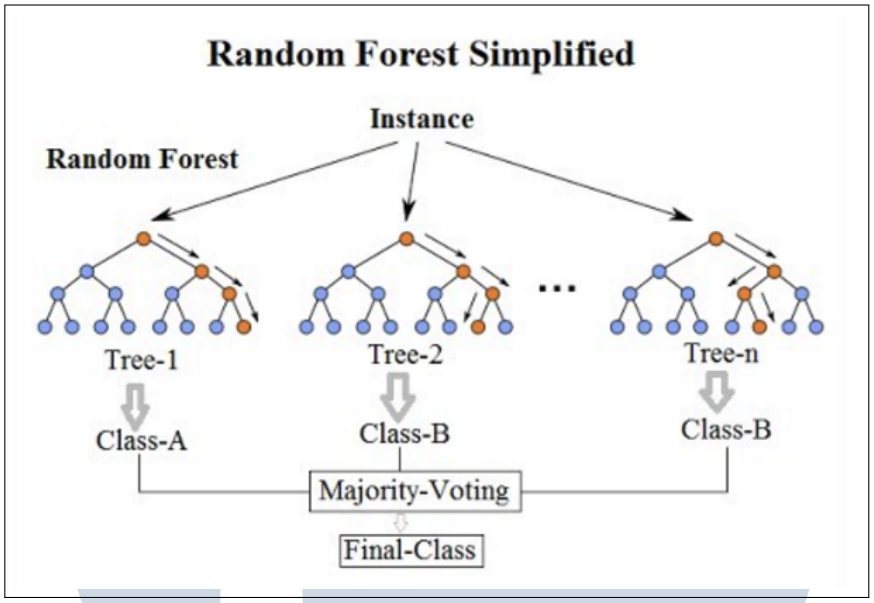
$$GiniIndex = 1 - \sum_{i=1}^{k} (p_i)^2 \tag{2.3}$$

Keterangan:

- $GiniIndex$ = Nilai dari *Gini Index*
- p_i = probabilitas dari *event i*

2.4 Random Forest

Random Forest (RF) merupakan salah satu algoritma yang termasuk kedalam *supervised learning* yang dikembangkan oleh Breiman pada tahun 2001 [20]. *Random Forest* merupakan algoritma pembelajaran mesin yang digunakan untuk menyelesaikan masalah regresi dan klasifikasi [7]. Algoritma *Random Forest* juga merupakan pengembangan dari algoritma *Decision Tree*, sebab pada algoritma *Decision Tree* terdapat kelemahan yaitu apabila data set yang dimiliki berjumlah sedikit akan tetapi memiliki banyak atribut, serta tingginya potensi terjadinya *overfitting* [8]. Seperti pada Gambar 2.1, algoritma *Random Forest* merupakan kombinasi dari beberapa *Decision Trees* atau *Tree Predictors*, dimana, semakin banyak *Decision Tree* yang terbentuk maka semakin baik pula hasil yang akan didapat.



Gambar 2.1. Random Forest
[21]

Semakin banyak *Decision Tree* yang terbentuk tidak mempengaruhi *overfitting* ataupun rentan terhadap *error*. *Output* dari setiap *Decision Tree* dalam melakukan klasifikasi dikumpulkan, kemudian, untuk membuat klasifikasi akhir sebagai *Final Class*, diterapkan persamaan *majority vote* seperti pada persamaan 2.4 [22]:

$$\hat{C}_{rf}^{B}(x) = majority\ vote(\hat{C}_b(x))_1^B \tag{2.4}$$

Adapun keuntungan menggunakan algoritma Random Forest dari pada algoritma *machine learning* yang lain [23], yaitu:

1. *Parallelizable*:

   Algoritma ini dapat diparalelkan, sehingga setiap proses dapat dibagi ke mesin-mesin yang berbeda pada saat dijalankan.

2. *Quick Prediction/Training Speed:*

   Pelatihan algoritma *Random Forest* Lebih cepat daripada algoritma *Decision Tree*, sebab yang dikerjakan hanyalah subset fitur dalam model. Untuk kecepatan prediksi juga secara signifikan lebih cepat daripada kecepatan pelatihan.

3. *Handles Unbalanced Data:*

Dalam algoritma *Random Forest*, terdapat metode untuk *balancing error* terhadap dataset yang tidak seimbang. Algoritma ini mencoba meminimalkan kesalahan dari keseluruhan model, sehingga saat melakukan pelatihan menggunakan dataset yang tidak seimbang, maka kelas yang lebih besar akan mendapatkan tingkat kesalahan yang rendah.

4. *Low Bias, Moderate Variance:*

   Setiap *Decision Tree* memiliki *high variance*, tetapi *low bias*. Tetapi, karena dalam algoritma *Random Forest*, sejumlah *tree* yang dihasilkan akan di rata-ratakan kembali, maka akan membuat tingkat variance datanya naik menjadi *moderate*.

2.5 Evaluasi Performa

Untuk mengukur performa dari model *machine learning* yang telah dibuat dapat menggunakan *Confusion Matrix* seperti pada Tabel 2.1, sebab *Confusion Matrix* berisi data prediksi yang positif dan negatif yang dihasilkan oleh sistem dan yang terdapat di dunia nyata [24].

Tabel 2.1. Table Confusion Matrix

	Positif (Aktual)	**Negatif Aktual**
Positif (Sistem)	TP	FP
Negatif (Sistem)	FN	TN

sumber: [25]

Adapun penjelasan Tabel 2.1 *Confusion Matrix* adalah sebagai berikut [26]:

1. TP (True Positive), yaitu perhitungan dari kelas positif sistem dan aktual yang positif.
2. TN (True Negative), yaitu perhitungan dari kelas negatif sistem dan aktual yang negatif.
3. FP (False Positive), yaitu perhitungan dari kelas positif sistem dan aktual yang negatif.
4. FN (False Negative), yaitu perhitungan dari kelas negatif sistem dan aktual yang positif.

Dalam *Confusion Matrix* terdapat beberapa metrik yang dapat diukur seperti *precision*, *recall*, *F1-score*, dan *accuracy*, adapun penjelasan mengenai metrik tersebut adalah sebagai berikut:

- *Precision*

  Precision merupakan tingkat ketepatan jawaban yang diberikan oleh sistem berdasarkan informasi yang diminta oleh pengguna [26]. Formula 2.5 merupakan formula untuk perhitungan metrik *precision*.

$$Precision = \frac{TP}{TP + FP} \times 100\% \tag{2.5}$$

- *Recall*

  Recall merupakan tingkat keberhasilan dalam mendapatkan informasi oleh sistem [26]. Formula 2.6 merupakan formula untuk perhitungan metrik *recall*.

$$Recall = \frac{TP}{TP + FN} \times 100\% \tag{2.6}$$

- *F1-Score*

  F1-Score merupakan pengukuran performa sistem yang telah dibuat dengan cara menggabungkan perhitungan *precision* dan perhitungan *recall*. Formula 2.7 merupakan formula untuk perhitungan metrik *F1-Score*

$$F1\text{-}Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \times 100\% \tag{2.7}$$

- *Accuracy*

  Accuracy merupakan representase ketepatan klasifikasi yang dilakukan berdasarkan dataset dan model.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \tag{2.8}$$