BAB 2
LANDASAN TEORI

Pada bagian ini akan dijelaskan mengenai teori-teori yang mendasari penelitian terkait *forecasting*, *machine learning*, *ensemble learning*, *random forest*, *confusion matrix*.

2.1 Tinjauan Teori

2.1.1 Forecasting

Forecasting, atau prediksi, merupakan sebuah metode ilmiah yang digunakan untuk memperkirakan nilai-nilai di masa depan dengan mendasarkan pada analisis data historis. Peramalan ini juga dapat diartikan sebagai suatu disiplin ilmu yang bertujuan untuk memproyeksikan kejadian-kejadian di masa depan berdasarkan pola-pola masa lalu [16]. Prediksi dapat dibedakan menjadi tiga kategori utama, yaitu prediksi jangka pendek, prediksi jangka menengah, dan prediksi jangka panjang. Prediksi jangka pendek adalah estimasi yang dilakukan dengan memperhatikan pola data yang ada, yang umumnya memerlukan periode waktu yang singkat. Fokusnya adalah untuk mengidentifikasi perubahan yang mungkin terjadi dalam jangka waktu dekat berdasarkan faktor-faktor yang membentuk pola data tersebut.

Di sisi lain, prediksi jangka menengah dan jangka panjang digunakan dalam konteks perencanaan strategis. Prediksi jangka menengah bertujuan untuk mempersiapkan ekspansi dan mengantisipasi kebutuhan di masa depan, sedangkan prediksi jangka panjang berfungsi untuk menjamin ketersediaan sumber daya atau kebutuhan di masa yang akan datang [17].

2.1.2 Machine Leaning

Machine learning, atau pembelajaran mesin, merupakan sebuah metode komputasi yang termasuk dalam ranah AI (*Artificial Intelligence*). Metode ini memungkinkan komputer untuk meniru, mempelajari, dan mengambil keputusan seperti manusia. Dalam prosesnya, komputer dilatih secara intensif sehingga mampu menyelesaikan masalah secara otomatis dengan menggunakan kecerdasan yang telah dilatih sebelumnya [18].

Ada dua fase utama dalam *machine learning*, yaitu fase klasifikasi dan fase prediksi. Ciri khas dari *machine learning* adalah adanya proses pelatihan dan pembelajaran. Proses pelatihan ini membutuhkan data yang disebut *data training*. Pada fase klasifikasi, objek akan dikelompokkan berdasarkan ciri-ciri dan atributnya [18]. Sedangkan pada fase prediksi, sistem akan melakukan perkiraan pada data baru yang dimasukkan, berdasarkan pengetahuan yang diperoleh dari data *training*. Model yang dihasilkan dari proses pelatihan ini akan digunakan untuk membuat keputusan baru dengan menguji data *testing* [19].

2.1.3 Ensemble Leaning

Ensemble learning adalah sebuah metode yang melibatkan pelatihan sejumlah model untuk menyelesaikan suatu masalah yang sama, dan kemudian menggabungkan hasil dari model-model tersebut untuk mendapatkan hasil yang lebih optimal [20]. Metode ini mencakup tiga jenis utama, yaitu *bagging*, *stacking*, dan *boosting* [21].

1. *Bagging*
   Dikenal sebagai *Bootstrap Aggregating*, *bagging* adalah *metode ensemble* yang mencari variasi dalam tipe model dengan menggunakan data pelatihan untuk menggabungkan prediksi yang beragam.

2. *Stacking*
   Dikenal juga sebagai *Stacked Generalization*, *stacking* adalah metode ensemble yang menggabungkan berbagai jenis model dengan cara memvariasikan data pelatihan untuk menghasilkan prediksi yang lebih baik.

3. *Boosting*
   Boosting adalah metode *ensemble* yang memfokuskan pelatihan pada contoh data yang sulit diprediksi oleh model, dengan tujuan memperbaiki akurasi keseluruhan dari model tersebut.

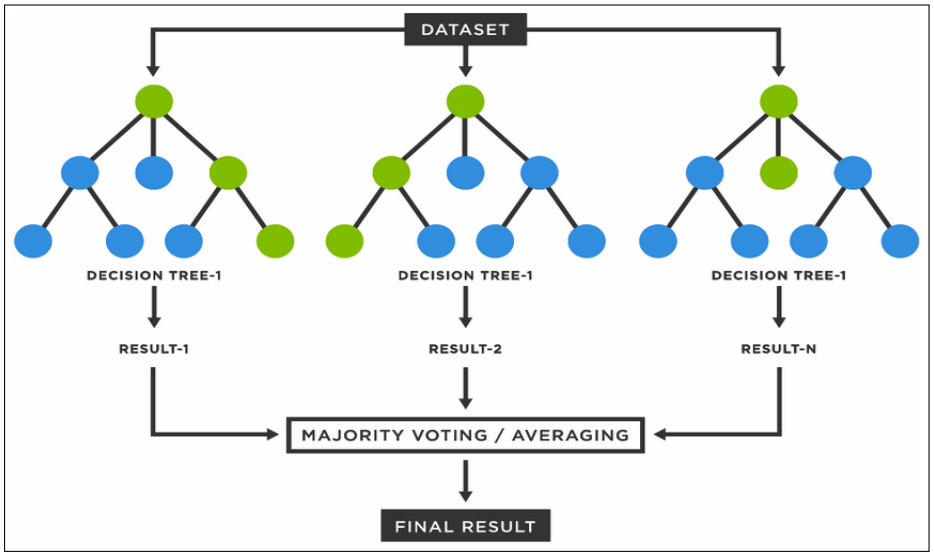
2.1.4 Random Forest

Random forest merupakan salah satu metode algoritma dalam *machine learning* yang sangat efektif digunakan untuk menangani masalah klasifikasi dan regresi [22]. Algoritma *random forest* dikenal memiliki akurasi tinggi dalam proses

klasifikasi, mampu mengatasi data yang tidak lengkap, serta dapat menangani data *training* dalam jumlah besar [23].

Random forest adalah algoritma yang dikembangkan oleh Leo Breiman dan Adele Cutler, yang menggabungkan hasil dari beberapa *decision tree* untuk mencapai satu hasil prediksi yang akurat [24].

Perbedaan utama antara algoritma *random forest* dan *decision tree* terletak pada hasil *output* yang dihasilkan [23]. Pada *decision tree*, setiap pohon menghasilkan *output* yang berbeda-beda. Sementara pada *random forest*, hasil akhir ditentukan melalui proses *voting* mayoritas dari seluruh pohon keputusan. Selain itu, perbedaan lain terletak pada kedalamannya, dimana algoritma *decision tree* lebih rentan terhadap *overfitting* dibandingkan *random forest* [25].



Gambar 2.1. Gambaran cara kerja *Random Forest*

Sumber: [26]

Konsep dasar dari metode *Random Forest* adalah membangun sejumlah pohon keputusan untuk mengklasifikasi suatu objek berdasarkan atribut-atributnya. Setiap pohon menghasilkan klasifikasi masing-masing, atau yang disebut dengan ”*votes*”. Hasil akhir kemudian dipilih berdasarkan mayoritas klasifikasi dari seluruh pohon tersebut. Namun, dalam kasus regresi, rata-rata *output* dari setiap pohon digunakan sebagai hasil akhir [27].

Random forest memiliki beberapa keunggulan, termasuk akurasi yang setara atau bahkan lebih baik dibandingkan metode *AdaBoost*, ketahanannya terhadap *outlier* dan *noise*, kecepatan yang lebih tinggi dibandingkan metode *bagging* atau *boosting*, kemampuan memberikan estimasi *error*, kekuatan, korelasi, serta variabel

yang penting. Algoritma ini juga mudah untuk diparalelkan [28].

Algoritma *random forest* juga sering digunakan dalam klasifikasi dan regresi. Salah satu penelitian yang menggunakan *random forest* adalah penelitian untuk memprediksi kemungkinan terjadinya banjir. Penelitian tersebut berhasil mencapai akurasi sebesar 99.05%.[13].

Berikut merupakan rumus untuk menghitung Random Forest:

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

Indeks: $\hat{C}_b(x) =$ Kelas prediksi dari pohon Random Forest ke-b.

Pada Python,*random forest* menggunakan *library sci-kit-learn* atau *sklearn*. Berikut merupakan parameter dari Random Forest pada *library sklearn*:

Tabel 2.1. Parameter model *Random Forest*

Parameter	**Deskripsi**
n_estimators	Jumlah *tree* dalam *forest*.
criterion	Kriteria yang digunakan untuk mengukur kualitas *split*. Kriteria yang didukung adalah ”gini” untuk ketidakmurnian dan ”entropy” untuk perolehan informasi.
max_depth	Menunjukkan kedalaman maksimum pada sebuah *tree*. Jika tidak ada batasan, node akan diperluas sampai semua *leaves* murni atau sampai daun mengandung sampel kurang dari $\min_samples_split$.
min_samples_split	Jumlah *minimum sampel* yang diperlukan untuk melakukan *splitting* pada node internal.
min_samples_leaf	Jumlah *minimum sampel* yang diperlukan untuk setiap *leaf* node. Titik pemisahan pada kedalaman apa pun hanya akan dipertimbangkan jika setiap cabang kiri dan kanan mengandung minimal jumlah sampel ini.
max_features	Jumlah fitur yang perlu dipertimbangkan saat mencari pemisahan terbaik.
	Lanjut pada halaman berikutnya

Tabel 2.2. Parameter model *Random Forest* (lanjutan)

Parameter	**Deskripsi**
max_features	Jumlah fitur yang perlu dipertimbangkan saat mencari pemisahan terbaik.
max_leaf_nodes	Tumbuhnya *tree* max_leaf_nodes. Node terbaik didefinisikan sebagai pengurangan relatif dalam pengotor. Jika tidak ada maka jumlah *leaf node* tidak terbatas.
min_impurity_decrease	Node akan terbagi jika pembagian tersebut mengurangi impuritas dengan nilai yang lebih besar atau sama dengan nilai ini.
bootstrap	Menentukan apakah sampel bootstrap digunakan saat *tree* dibangun.
oob_score	Menentukan apakah *Out-of-bags samples* digunakan untuk estimasi *generalization score*.
n_jobs	Jumlah pekerjaan yang harus dijalankan secara paralel untuk kecocokan dan prediksi.
random_state	Mengontrol keacakan pada pemilihan sampel bootstrap yang digunakan dalam membangun *tree*, serta pengambilan sampel fitur yang dipertimbangkan untuk pemisahan terbaik di setiap node.
verbose	Mengontrol verbositas ketika melakukan *fitting* dan prediksi.
warm_start	Jika diset ke *True*, maka model akan menggunakan kembali panggilan sebelumnya untuk menyesuaikan dan menambahkan lebih banyak estimator ke dalam *ensemble*. Jika tidak, seluruh *forest* baru akan dibangun.
ccp_alpha	Jumlah parameter kompleksitas yang digunakan untuk *Minimal Cost-Complexity Pruning*.
max_samples	Jika *bootstrap True*, jumlah sampel yang diambil dari X untuk melatih setiap *base estimator*.

Sumber: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

2.1.5 Confusion Matrix

Setelah model *machine learning* selesai dibangun, langkah selanjutnya adalah melakukan evaluasi untuk mengukur kinerjanya. Salah satu metode evaluasi yang umum digunakan untuk mengukur performa model adalah *confusion matrix*. *Confusion matrix* adalah metode evaluasi yang sederhana namun efektif untuk menunjukkan performa model klasifikasi. Metode ini memiliki keunggulan dalam memudahkan interpretasi hasil evaluasi. Indikasi dari *confusion matrix* bisa dilihat pada Tabel 2.3.

Tabel 2.3. *Confusion matrix*

Confusion Matrix		True Class (Actual)	
		Positive	**Negative**
Hypothesized Class (Predicted)	**Yes**	True Positives	False Positives
	No	False Negatives	True Negatives

Sumber: [29]

Terdapat empat metrik pada *confusion matrix* besertakan penjelasannya sebagai berikut:

1. Accuracy
   Accuracy didapatkan dengan cara membagi rata total angka yang benar (*true* $(TP+TN)$) dengan total data yang ada.

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

2. Precision
   Precision merupakan metrik yang digunakan untuk mengukur proporsi prediksi positif *true positive* (TP). Metrik ini digunakan sebagai indikator untuk mengurangi jumlah *false positive* (FP). *Precision* dihitung dengan membagi jumlah *true positive* dengan jumlah total prediksi positif, yang mencakup *true positive* dan *false positive*. Dengan kata lain, *precision* menunjukkan seberapa akurat model dalam mengidentifikasi contoh positif di antara semua prediksi positif yang dibuat.

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

3. Recall

   Recall merupakan angka keberhasilan dalam mendapatkan informasi oleh sistem.

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

4. F1 Score

   F1 Score merupakan nilai performa sistem berdasarkan perhitungan *precision* dan *recall*.

$$F1\,Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.5}$$