# BAB 2
# LANDASAN TEORI

## 2.1 Tinjauan Teori

Pada bagian ini dijabarkan teori-teori yang digunakan untuk mendukung penelitian adalah sebagai berikut.

### 2.1.1 Feature Selection

*Feature selection* mempunyai langkah penting dalam banyak tugas *pattern recognition* dan tugas *machine learning*. Salah satu tantangan utama dari *feature selection* adalah menghadapi dataset yang memiliki dimensi tinggi dan banyak sekali dataset yang mengandung fitur yang redundan dan tidak relevan [12]. *Feature selection* dapat digunakan untuk mengidentifikasi dan memilih fitur mana yang sangat relevan dan informatif dalam sebuah dataset sehingga fitur yang tidak relevan dihapus [13]. Fitur yang tidak relevan dan redundan dalam dataset memiliki dampak negatif pada proses *machine learning* yang dapat mengakibatkan terjadinya *overfitting*, melambatkan proses kecepatan, dan mempersulit dalam interpretasi model [14]. Terdapat beberapa kategori *feature selection* diantaranya yaitu[15]:

1. Metode *Filter*. Metode *filter* memilih fitur berdasarkan peringkat dan akan menghapus fitur yang memiliki nilai dibawah *threshold*. Keuntungan dari menggunakan metode *filter* adalah cepat, dapat diukur ke dataset besar, dan independen. Terdapat juga kekurangan dari menggunakan metode *filter*, yaitu mengabaikan ketergantungan fitur dan kurangnya interaksi dengan *classifier*. Salah satu contoh metode *filter* adalah *Information Gain*.

2. Metode *Wrapper*. Metode *wrapper* menggunakan teknik pencarian untuk menghasilkan sebuah subset fitur, dan akan dievaluasi dengan *supervised learning algorithm* berdasarkan kesalahan klasifikasi dan akurasi. Metode *wrapper* memiliki keunggulan dari metode *filter* dikarenakan dapat berinteraksi dengan *classifier* dan mempertimbangkan dependensi antar fitur. Tetapi terdapat kekurangan yaitu metode *wrapper* cenderung lebih kompleks dan memakan waktu serta akan berpotensi untuk menimpulkan *over-fitting* pada *small training datasets*. Salah satu contoh metode *wrapper* adalah *Genetic Algorithm*.

3. Metode *Embedded*. Metode *embedded* akan mencari subset fitur optimal untuk algoritma klasifikasi tertentu. Metode ini dapat berinteraksi dengan pengklasifikasi, memperhatikan dependensi antar fitur, dan memiliki kompleksitas komputasi yang lebih baik daripada metode *wrapper*. Metode *embedded* lebih mahal dari segi komputasi jika dibandingkan dengan metode *filter*. Salah satu contoh metode *embedded* adalah *Artificial Neural Network*.

4. Metode *Hybrid*. Metode *hybrid* menggabungkan beberapa metode *feature selection* dan akan memanfaatkan kelebihan dari masing-masing untuk meningkatkan akurasi dan efisiensi. Salah satu contoh metode *hybrid* adalah *Particle Swarm Optimization-Support Vector Maching* (PSO-SVM).

### 2.1.2 Entropy

*Entropy* digunakan untuk mengukur sejauh mana ketidakmurnian atau *randomness* dalam sebuah dataset yang digunakan. *Entropy* berguna dalam *information gain*, karena *information gain* didasarkan pada *Entropy* [16]. Nilai *Entropy* selalu berada di antara 0 dan 1. Nilai *Entropy* lebih baik ketika mendekati atau sama dengan 0, lebih buruk ketika mendekati atau sama dengan 1 [17].

$$Entropy(S) = \sum_{i=1}^{n} pi.log2pi \tag{2.1}$$

Pada Persamaan 2.1, nilai S adalah himpunan kasus. Selanjutnya nilai n adalah jumlah partisi, dan nilai pi adalah proporsi Si terhadap S [18].

### 2.1.3 Logistic Regression

Algoritma *logistic regression* merupakan metode *supervised learning* yang digunakan untuk memecahkan masalah *binary classification* [19]. Variabel hasil atau target hanya terdapat dua kemungkinan, ya dan tidak, dengan hanya dua kemungkinan, *logistic regression* dapat digunakan untuk mendeteksi *spam email*, atau penyakit [20]. Meskipun *logistic regression* adalah teknik klasifikasi yang sederhana namun kuat dan bahkan digunakan sebagai fungsi aktivasi dalam *neural networks* [19].

### 2.1.4 Information Gain

*Information gain* adalah perbedaan antata *entropy class* dan *entropy conditional class* dan *feature* yang dipilih [16]. *Information gain* umum digunakan sebagai kriteria pemilihan fitur, dan *information gain* memberikan peringkat pada setiap fitur yang ada dan menghapus fitur yang tidak memenuhi kriteria. Fitur yang memiliki nilai *information gain* yang lebih tinggi maka fitur tersebut lebih baik daripada fitur lainnya, dan menunjukkan bahwa fitur tersebut memiliki lebih banyak informasi terkait dengan kelas [21].

$$Gain(S,A) = Entropy(S) - \sum_{i=1}^{n} \frac{|si|}{|s|} + entropy(Si) \tag{2.2}$$

Pada Persamaan 2.2, $S$ melambangkan himpunan kasus dan A melambangkan atribut. Nilai *Entropy* (S) merupakan nilai *entropy* yang sudah dihitung sebelumnya setiap atribut. Lambang n merupakan jumlah partisi atribut A. Lalu $|Si|$ adalah jumlah kasus pada partisi ke -i, dan $|S|$ adalah jumlah kasus dalam s [18].

### 2.1.5 Threshold

*Threshold* merupakan sebuah nilai yang digunakan sebagai referensi untuk pemilihan fitur berdasarkan hasil *information gain* [6]. Sejak tahun 1940-an sudah terdapat ratusan artikel yang sudah diterbitkan dan menggunakan nilai threshold 0,05. Nilai *threshold* tidak seharusnya bernilai 0,05, tetapi nilai *threshold* tidak disarankan untuk mengabaikan nilai 0,05. Alasan utamanya adalah agar tidak menambah kebingungan dalam penelitian [4]. Penelitian yang dilakukan oleh Tsai dan Sung menggunakan perhitungan nilai rata-rata setiap fitur atau bisa disebut nilai *mean* untuk dijadikan nilai *threshold* akhir [10]. Ide dari penelitian yang dilakukan oleh Tsai memperbolehkan bahwa nilai *threshold* bisa menggunakan perhitungan statistika seperti *mean* atau perhitungan statistika lainnya. Sehingga penelitian ini menggunakan nilai median sebagai nilai akhir *threshold*

Terdapat dua rumus perhitungan untuk mencari nilai median, yaitu

1. Rumus perhitungan median apabila data tersebut memiliki jumlah ganjil
2. Rumus perhitungan median apabila data tersebut memiliki jumlah genap

$$Me = X\frac{1}{2}(n+1) \tag{2.3}$$

Pada Persamaan 2.3 merupakan rumus apabila data nya berjumlah ganjil. Me adalah median, selanjutnya nilai n adalah banyak data.

$$Me = \frac{\mathrm{X}\left(\frac{n}{2}\right) + \mathrm{X}\left(\frac{n}{2}n + 1\right)}{2} \tag{2.4}$$

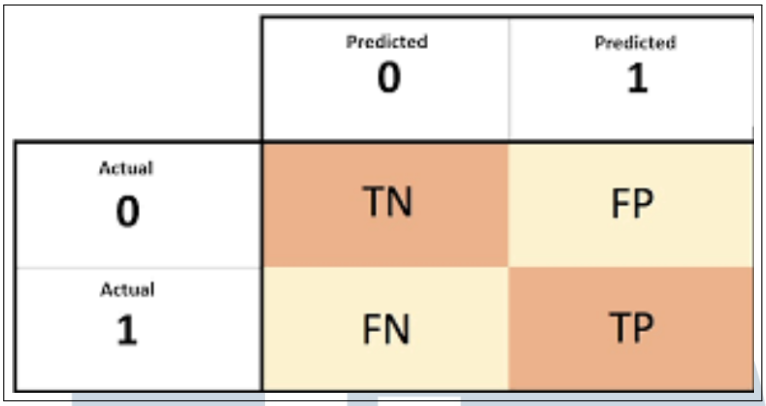
Pada Persamaan 2.4 merupakan rumus median apabila data nya berjumlah genap. Nilai n adalah banyak data [22].

### 2.1.6 K-Fold Cross Validation

Teknik *cross validation* utamanya digunakan untuk melakukan prediksi model dan melakukan perkiraan seberapa akurat sebuah model algoritma prediktif ketika dijalankan [23]. Proses pelatihan dan pengujian sebuah model algoritma sering kali dilakukan menggunakan satu set data tunggal, yang dapat mengakibatkan bias dalam kinerja algoritma. Melalui teknik *cross validation*, memungkinkan untuk melatih, menguji, dan memvalidasi dengan menggunakan beberapa set data atau lipatan. Ada beberapa contoh teknik *cross validation*, tetapi teknik *K-fold cross validation* dianggap paling menguntungkan karena menggunakan seluruh data untuk melatih dan memvalidasi, menghasilkan hasil yang lebih representatif tanpa bias atau risiko kesalahan [24]. *K-fold cross validation* merupakan sebuah metode statistik yang digunakan untuk melakukan evaluasi performa dari sebuah model algoritma [25]. Dalam *K-fold cross validation*, *sample* asli dibagi menjadi K *sub-sample*, dan hasil dari setiap iterasi dihitung rata-rata [26].

### 2.1.7 Confusion Matrix

*Confusion matrix* adalah sebuah metode untuk menilai performa model klasifikasi. *Confusion matrix* akan merangkum prediksi yang benar dan salah yang dibuat oleh model algoritma, disusun berdasarkan *class* [19].




Gambar 2.1. Confusion Matrix

Sumber: [20]

Pada Gambar 2.1 terdapat 4 kondisi dalam *confusion matrix*:

- *True Positive* (TP) prediksi nilai *positive, real positive*
- *True Negative* (TN) prediksi nilai *negative, real negative*
- *False Positive* (FP) prediksi nilai *positive, real negative*
- *False Negative* (FN) prediksi nilai *negative, real positive*

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.5}$$

Pada Persamaan 2.5, *accuracy* mengukur tingkat ketepatan prediksi.

$$Precision = \frac{TP}{TP + FP} \tag{2.6}$$

Pada Persamaan 2.6, *precision* memberi tahu berapa banyak dari kasus yang diprediksi dengan benar ternyata *positive*. Hasil dari *precision* menentukan apakah model algoritma tersebut dapat diandalkan atau tidak.

$$Recall = \frac{TP}{TP + FN} \tag{2.7}$$

Pada Persamaan 2.7 menunjukkan jumlah kasus *positive* yang sebenarnya dapat diprediksi dengan benar menggunakan model algoritma.

$$F1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.8}$$

Pada Persamaan 2.8 memberikan sebuah gambaran gabungan tentang *matrix precision* dan *recall*.