BAB 2
LANDASAN TEORI

Ada beberapa teori yang digunakan untuk menunjang keberhasilan penelitian ini yaitu Myers-Briggs Type Indicator(MBTI) dan algoritma Support Vector Machines(SVM).

2.1 Myers-Briggs Type Indicator (MBTI)

Myers-Briggs Type Indicator (MBTI) adalah klasifikasi kepribadian yang dirancang untuk mengidentifikasi tipe kepribadian, kekuatan, dan preferensi seseorang. Pada awalnya, survei dilakukan oleh Isabel Myers dan ibunya, Katherine Briggs berdasarkan teori tipe kepribadian Carl Jung. Berdasarkan hasil survei, manusia dikelompokkan menjadi 16 tipe kepribadian [13]. Dengan adanya MBTI ini, diharapkan setiap individu dapat lebih memahami dirinya sendiri dan orang lain terhadap preferensi, kelebihan, kekurangan, pemilihan karir, serta kecocokan antar individu. Berikut merupakan empat kombinasi dimensi kepribadian yang membentuk 16 tipe kepribadian manusia menurut teori MBTI.

1. Individu dan benda (*Extraversion* atau ”E”) atau ide dan informasi (*Introversion* atau ”I”).
2. Fakta dan realitas (*Sensing* atau ”S”) atau kemungkinan dan potensi (*Intuition* atau ”N”).
3. Logika dan kebenaran (*Thinking* atau ”T”) atau nilai dan hubungan (*Feeling* atau ”F”).
4. Gaya hidup yang terstruktur (*Judgment* atau ”J”) atau gaya hidup yang mengikuti arus (*Perception* atau ”P”).

Berikut merupakan penjelasan dari setiap dimensi kepribadian tersebut [14].

1. *Extraversion* dan *Introversion*

   Kombinasi ini merupakan kombinasi yang berkaitan dengan arah energi yang dimiliki oleh setiap individu. *Extraversion* merupakan preferensi untuk individu yang mengarahkan energinya ke orang lain, benda, situasi, atau dunia luar. Sedangkan *Introversion* merupakan individu yang mengarahkan energinya ke dalam ide, informasi, keyakinan, atau dunia batin dirinya sendiri.

2. *Sensing* dan *Intuition*

   Kombinasi yang kedua berkaitan dengan jenis informasi yang diproses oleh setiap individu. *Sensing* merupakan preferensi untuk individu yang cenderung menyukai fakta terhadap apa yang diketahui dan dilihat. Sedangkan individu yang lebih menyukai ide yang belum diketahui serta dapat menghasilkan kemungkinan baru maka preferensinya yaitu *Intuition*(N).

3. *Thinking* dan *Feeling*

   Kombinasi yang ketiga merupakan cerminan dari cara pengambilan keputusan masing-masing individu. *Thinking* bila individu tersebut memutuskan sesuatu berdasarkan logika dan analisis. Sedangkan *Feeling* bila pengambilan keputusan berdasarkan suatu nilai (contoh, apa atau siapa yang dipercayai).

4. *Judgment* dan *Perception*

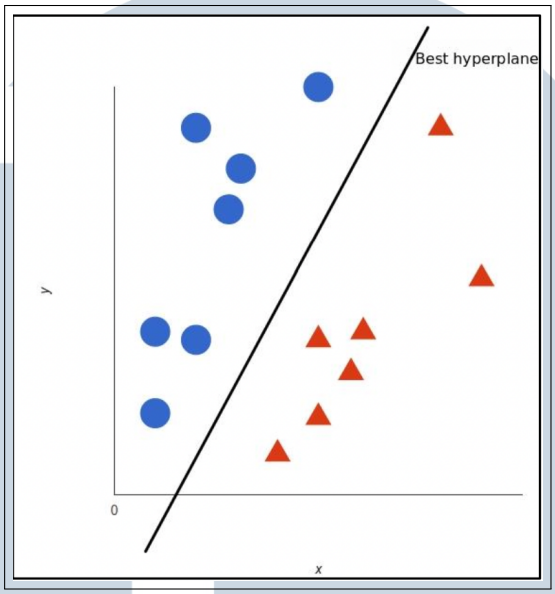
   Terakhir, kombinasi ini merupakan gambaran dari jenis gaya hidup yang diterapkan. Seorang *Judging* menyukai gaya hidup yang terencana dan terstruktur dengan baik. Sedangkan *Perception* menyukai gaya hidup yang mengikuti arus, serta fleksibel dalam menanggapi berbagai hal yang muncul.

Berdasarkan dimensi kepribadian di atas, kombinasi dari empat huruf tersebut kemudian membentuk kode dari tipe kepribadian seseorang. Dengan begitu, terdapat 16 kombinasi tipe kepribadian Myers-Briggs. Kombinasi tipe kepribadian tersebut yaitu ISTJ, ISFJ, INFJ, INTJ, ISTP, ISFP, INFP, INTP, ESTP, ESFP, ENFP, ENTP, ESTJ, ESFJ, ENFJ, ENTJ.

2.2 Support Vector Machines

Support Vector Machines merupakan algoritma *supervised machine learning* yang dapat digunakan untuk memecahkan masalah klasifikasi maupun regresi. Tetapi, SVM kebanyakan digunakan untuk kasus klasifikasi. Sedangkan untuk memecahkan kasus regresi, digunakan Support Vector Regression (SVR) yang merupakan bagian dari algoritma SVM dengan pengimplementasian yang berbeda. SVM dapat digunakan untuk memecahkan masalah linear maupun non linear dengan beberapa karakteristik yaitu bersifat *non-probabilistic*, membagi data menjadi kelompok data yang dibatasi oleh hyperplane [15]. SVM mengklasifikasikan data dengan menggunakan *hyperplane* yang memisahkan data menjadi kelas-kelas tertentu. *Hyperplane* ini dihasilkan dari transformasi data dengan bantuan fungsi

matematika yang disebut sebagai *kernels*. Terdapat beberapa tipe *kernels* yaitu linear, polynomial, sigmoid, RBF, Anova RB, Laplacian, dan lain-lain [16].



Gambar 2.1. *Hyperplane* 2D terbaik
Sumber: [17]

Berdasarkan titik data, SVM menghasilkan *hyperplane* (pada 2D merupakan sebuah garis) yang memisahkan antar label. Garis tersebut merupakan *decision boundary* yang menandakan titik data yang berada pada sisi kiri akan tergolong ke label biru dan titik data yang berada pada sisi kanan akan tergolong ke label merah. Pada SVM, *hyperplane* terbaik merupakan *hyperplane* yang memaksimalkan jarak antar label, di mana jarak ke elemen terdekat pada setiap label adalah yang terbesar [16]. Sedangkan data terdekat pada *hyperplane* disebut sebagai *support vector* [7].

Data *training* dinotasikan dengan $x \in R^n$ yang merupakan anggota dari kelas y dan $y \in [-1,1]$. Fungsi *linear decision* SVM ditentukan sebagai berikut.

$$f(x) = w^T x + b, w \in R^n, b \in R \tag{2.1}$$

Hyperplane dapat dinyatakan melalui persamaan berikut.

$$\vec{w} \cdot \vec{x} - b = 0 \tag{2.2}$$

Pencarian *hyperplane* optimum dengan mencari jarak *margin* maksimal di-

dapatkan dengan menghitung titik minimal pada Persamaan 2.3.

$$min\frac{1}{2}\|\vec{w}\|^2 \tag{2.3}$$

dengan batasan

$$y_i(x_i \cdot \vec{w} + b) - 1 \geq 0, \forall i \tag{2.4}$$

Kemudian, persamaan ini dapat diselesaikan dengan metode *Lagrange Multiplier* dengan memaksimalkan titik optimal gradien, maka persamaannya yaitu sebagai berikut.

$$Max : W(\alpha) = -\frac{1}{2}\sum_{i,i=1}^{N} \alpha_i \alpha_j y_i y_j x + \sum_{i=1}^{N} \alpha i \tag{2.5}$$

dengan batasan

$$\sum_{i=1}^{N} \alpha_i y_i = 0, \alpha_i \geq 0 (i = 1, 2, ...N) \tag{2.6}$$

Berdasarkan hubungan Persamaan 2.5 dan 2.6, maka nilai α dan b menggunakan fungsi *Quadratic Programming* (QP) telah didapatkan. Berikutnya, fase pengujian dilakukan berdasarkan hubungan berikut.

$$F(x, \alpha, b) = \omega x + b = \sum_{i=1}^{N} \alpha_i y_i x_i x + b = \sum_{i=2x} \alpha_i y_i x_i x + b \tag{2.7}$$

Penyelesaian dengan fungsi kernel dapat dilakukan sebagai berikut.

$$n\text{-}th\ Degree\ Polynominal : K(x, x') = (1 + (x, x')) \tag{2.8}$$

$$Neural\ Network : K(x, x') = tan(k_1 < x, x' > + k_2) \tag{2.9}$$

$$Radial\ basis : K(x, x') = exp(-\|x - x'\|^2 / c) \tag{2.10}$$

2.3 Preprocessing

Tahap *preprocessing* merupakan tahap yang memiliki dampak signifikan pada algoritma *supervised machine learning* [18]. Berikut merupakan beberapa tahap *preprocessing* :

1. Data Cleaning

   Data cleaning merupakan langkah *preprocessing* yang dilakukan dengan mengidentifikasi dan memperbaiki kesalahan pada *dataset* yang dapat memberikan pengaruh buruk pada model [19]. Langkah ini diperlukan karena *dataset* yang dimiliki masih belum siap untuk diproses dan memiliki banyak *noise*. *Dataset* masih mengandung tanda baca, karakter dan simbol yang tidak memiliki fungsi tertentu, serta emotikon. Oleh karena itu, diperlukan *preprocessing* untuk menghilangkan karakter selain huruf, menyelaraskan huruf dan kata sehingga dapat meningkatkan performa sistem dalam menjalankan proses klasifikasi.

2. Case Folding

   Langkah *preprocessing* berikutnya yang banyak diterapkan dalam klasifikasi teks yaitu *case folding*. *Case folding* merupakan langkah *preprocessing* dengan mengganti seluruh huruf dengan *lower case* [20]. Karena makna kata dengan huruf besar atau kecil dianggap tidak memiliki perbedaan, semua karakter huruf besar biasanya diubah ke bentuk huruf kecilnya sebelum memulai klasifikasi [21].

3. Stop Words Removal

   Stop words merupakan kumpulan kata yang biasa ditemukan pada teks tanpa bergantung pada suatu topik tertentu misalnya, konjungsi, preposisi, artikel, dan lain-lain. Oleh karena itu, *stop words* biasanya dianggap tidak relevan dalam klasifikasi teks dan dihilangkan sebelum proses klasifikasi [21]. Dengan adanya penghapusan *stop words*, *dataset* yang digunakan akan lebih bersih dan dapat menciptakan fitur yang lebih baik pada model [22].

4. Lemmatization

   Lemmatization merupakan salah satu teknik *Text Normalization* yang biasanya digunakan dalam praproses data teks [23]. Dalam penggunaan kata dalam kalimat, seringkali terdapat kata yang dalam penggunaannya berasal

dari kata lain. Dalam hal ini, biasanya terdapat awalan, akhiran, maupun sisipan. Sehingga, kata tersebut memiliki akar kata. Contohnya, *playing*, *plays*, dan *played* berakar dari kata 'play'. Beberapa kata yang berbeda memiliki arti yang sama, sehingga pada tahap *preprocessing* diperlukan *lemmatization* untuk mencari akar kata. Setelah membentuk akar kata, seluruh kata yang terduplikasi akan dihapus.

2.4 TF-IDF

Ekstraksi fitur teks merupakan salah satu tahap yang penting dalam pemrosesan data dan memperoleh informasi dari suatu teks. Tahap ini mengukur fitur kata yang diekstraksi dari teks untuk merepresentasikan informasi teks dan mengubahnya dari teks tidak terstruktur menjadi informasi yang dapat dikenali dan diproses oleh komputer [24]. Dalam proses ekstraksi fitur, data yang tidak relevan akan dihapus dan data yang penting akan digabungkan dengan bobot yang mencerminkan informasi yang terkandung dalam teks.

Term Frequency-Inverse Document Frequency adalah metode ekstraksi fitur yang biasanya digunakan dalam klasifikasi teks berbasis ruang vektor [25]. TF-IDF merupakan ukuran yang digunakan dalam bidang pencarian informasi dan *machine learning* yang dapat mengukur kepentingan atau relevansi dari suatu string (kata, frasa, dan lain-lain) dalam suatu dokumen [26]. *Term frequency* merupakan angka yang menunjukkan seberapa sering kata tersebut muncul dalam sebuah dokumen, sedangkan *inverse document frequency* merupakan nilai yang menunjukkan seberapa umum atau langka kata tersebut dalam sebuah dokumen [27].

TF-IDF dihasilkan dengan menimbang setiap kata yang muncul dan menghitung *inverse value* dalam kalimat. Kata tersebut mewakili setiap fitur dalam dokumen. TF-IDF direpresentasikan dalam sebuah *array*, di mana masing-masing baris *array* berisi data dan kolom *array* berisi kumpulan kata. Hasil yang diperoleh yaitu berupa berat yang digunakan sebagai *input* untuk pelatihan model [25]. Perhitungan nilai TF, IDF, dan TF-IDF dapat dilihat pada rumus Persamaan 2.11, Persamaan 2.12, dan Persamaan 2.13 di bawah ini.

$$TF(w) = \frac{count(w)}{\sum_{j=1}^{m} count(w^{t_j})} \tag{2.11}$$

$$IDF(w) = ln(\frac{n}{m+1}) \tag{2.12}$$

$$TFIDF(w) = \frac{n \cdot count(w)}{ln(\frac{n}{m+1}) \cdot \sum_{j=1}^{m} count(w^{t_j})} \tag{2.13}$$

dengan :

$TF(w)$: frekuensi kata w dalam teks

Count(w) : jumlah kata w pada teks target

$count(w^{t_j})$: jumlah kata w pada sampel t_j

$IDF(w)$: frekuensi *inverse file* dari kata w

m : jumlah sampel yang mengandung kata w dalam korpus

n : jumlah seluruh teks dalam korpus

2.5 Synthetic Minority Oversampling Technique

Pada *machine learning*, untuk kasus klasifikasi, *dataset* akan digunakan untuk menjadi data latih yang kemudian diprediksi oleh sistem melalui data uji. Berbagai algoritma *machine learning* menganggap bahwa seluruh target kelas yang digunakan memiliki persebaran data yang seimbang. Namun, di dunia nyata, data yang terbentuk tidak selalu seimbang, contohnya pada data diagonosis penyakit yang kebanyakan berisi satu label mayoritas. Pada kasus seperti ini, sistem akan condong bergantung pada data mayoritas dan cenderung mengabaikan data minoritas. Hal ini menyebabkan performa model akan menurun ketika *dataset* yang digunakan tidak seimbang [28]. Pada algoritma SVM, data yang tidak seimbang dapat menyebabkan *hyperplane* (garis pemisah pada algoritma SVM) akan lebih condong ke kelas minoritas yang berdampak pada penurunan kinerja model [29]. Kelas minoritas dapat menyebabkan *hyperplane* bergerak lebih jauh dari batas kelas ideal. Sehingga, agar mengurangi jumlah kesalahan klasifikasi, *hyperplane* akan bergeser condong ke kelas minoritas yang menyebabkan hasil prediksi *false negative*. Oleh karena itu, dibutuhkan teknik penyeimbangan *dataset* atau *resampling*.

Salah satu teknik menyeimbangkan *dataset* yaitu *oversampling*. Teknik *oversampling* merupakan teknik meningkatkan jumlah data kelas minoritas baik dengan menambah sampel baru maupun menduplikat data yang sudah ada [28].

Salah satu teknik *oversampling* yaitu Synthetic Minority Oversampling Technique (SMOTE). Teknik *synthetic minority oversampling* SMOTE diajukan oleh [30] sebagai perkembangan dari metode *oversampling* dengan replikasi data. Teknik ini melakukan *oversample* data kelas minoritas dengan mengambil setiap sampel data kelas minoritas, kemudian tergantung pada jumlah data sintetik yang diperlukan, sejumlah tetangga terdekat akan dipilih secara acak. Sampel sintetik dihasilkan dari perkalian antara jarak tetangga dan kelas minoritas yang dipilih, lalu dikalikan dengan angka acak antara 0 dan 1, dan ditambahkan ke vektor data minoritas tersebut. Teknik ini membuat data kelas minoritas menjadi lebih umum [30].

Untuk data numerik diukur jaraknya dengan jarak Euclidean sedangkan data kategorikal yaitu dengan nilai modus. Perhitungan jarak antar kelas minoritas yang berskala kategorikal dihasilkan dari rumus Value Difference Metric (VDM) berikut [31].

$$\Delta(X,Y) = w_x w_y \sum_{i=1}^{N} \delta(x_i, y_i)^r \tag{2.14}$$

dengan :

$\delta(X,Y)$: jarak antara X dan Y

W_x, W_y : bobot amatan

N : banyaknya peubah

R : bernilai 1 bila jarak Manhattan dan bernilai 2 bila jarak Euclidean

$\delta(x_i, y_i)^r$: jarak antar kategori, dengan rumus :

$$\delta(V_1, V_2) = \sum_{i=1}^{N} \left| \frac{C_{1i}}{C_1} - \frac{C_{2i}}{C_2} \right|^k \tag{2.15}$$

dengan :

$\delta(V_1, V_2)$: jarak nilai V_1 dan V_2

C_{1i} : banyaknya V_1 yang termasuk kelas i

C_{2i} : banyaknya V_2 yang termasuk kelas i

I : banyaknya kelas

C_1 : banyaknya nilai 1

C_2 : banyaknya nilai 2

N : banyaknya kategori

K : konstanta

2.6 Evaluation Metrics

Pada tahap uji, *evaluation metric* digunakan sebagai evaluator untuk mengukur efektivitas klasifikasi yang dihasilkan saat menguji data [32]. Metrik evaluasi yang digunakan untuk mengevaluasi baik kasus klasifikasi biner maupun klasifikasi *multi-class* yaitu *confusion matrix*. *Confusion matrix* merupakan tabel yang berisi jumlah kemunculan antara dua penilai yaitu klasifikasi yang sebenarnya dan klasifikasi hasil prediksi [33].

1. Accuracy

   Accuracy merupakan ukuran seberapa banyak model memprediksi dengan benar data yang diuji. *Accuracy* dapat dikalkukasikan melalui rumus berikut.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.16}$$

   Dengan kata lain, nilai *accuracy* merupakan probabilitas model akan memprediksi data secara benar.

2. Precision

   Precision merupakan proporsi kebenaran hasil prediksi model yang menyatakan *positive* dan hasilnya benar-benar *positive*. Dengan kata lain, seberapa model dapat dipercaya ketika hasil prediksinya yaitu *positive*. *Precision* dapat dihitung melalui rumus :

$$Precision = \frac{TP}{TP + FP} \tag{2.17}$$

3. Recall

   Recall mengukur seberapa akurat model memprediksi kelas positif. Untuk menghitung nilai *recall*, digunakan rumus Persamaan 2.18 berikut.

$$Recall = \frac{TP}{TP + FN} \tag{2.18}$$

4. F1 Score

   F1-Score dapat diartikan sebagai rata-rata antar *precision* dan *recall* di mana nilai tertinggi yang dapat diraih yaitu 1 dengan nilai terendah 0. Perhitungan *F1-Score* dapat dilihat melalui rumus Persamaan 2.19 berikut.

$$F1\text{-}Score = 2 \cdot \left(\frac{precision \cdot recall}{precision + recall} \right) \tag{2.19}$$