BAB II
LANDASAN TEORI

2.1 Tanaman Tomat

Tomat (Solanum lycopersicum L) merupakan salah satu tanaman sayuran yang dapat tumbuh di seluruh dunia. Tanaman tomat tergolong tanaman semusin (annual). Artinya, tanaman berusum pendek yang hanya satu kali berproduksi dan setelah itu mati. Secara taksonomi tanaman tomat digolongkan sebagai berikut :

Kingdom : *Plantae*

Divisio : *Magnoliophyta*

Kelas : *Magnoliopsida*

Subkelas : *Asteridae*

Ordo : *Solanales*

Famili : *Solanaceae*

Genus : *Solanum*

Species : *Silanum lycopersicum* L.

Tanaman merupakan tanaman herbal semusim dari keluarga *Solanaceae*. Batang tanaman tomat bervariasi ada yang tegak atau menjalar, padat dana merambat, berwarna hijau, berbentuk silinder dan ditumbuhi rambut-rambut halus terutama dibagian yang berwarna hijau. Daunnya berbentuk oval dan bergerigi dan termasuk daun majemuk. Daun tanaman tomat biasanya berukuran panjang sekitar 20 – 30 cm serta lebarnya 16 – 20 cm. Daun tanaman tomat memiliki jarak dekat dengan ujung dahan sementara tangkai daunnya berbentuk bulat berukuran 7 – 10 cm. Bungan tomat berwarna kuning cerah, termasuk hermaprodit dan dapat menyerbuk sendiri (Setiawan, 2015).

2.2 Penyakit pada Daun Tomat

Tanaman dikatakan terserang penyakit bila pertumbuhannya menyimpang dari keadaan normal. Penyebabnya terdiri dari beberapa macam, diantaranya jamur atau cendawan, bakteri, dan virus. Beberapa jenis penyakit yang terdapat pada tanaman tomat diantaranya penyakit *yellow leaf curl, potato virus y, tomato mosaic virus*.

a. Yellow Leaf Curl

   Ciri dari penyakit pada bagian ujung dari daun akan menggulung ke dalam daun disertakan warna kuning pada setiap tepian daun. Hal ini dapat menjadikan pertumbuhan menjadi terhambat.

b. Potato Virus Y

   Penyakit ini disebabkan oleh kutu daun yang memakan daun tanaman dan virus akan menyebar dengan cepat dalam 1 menit saja. Ciri dari virus ini terdapat bercak pada daun, tangkai daun melengkung dan daun menggulung ke bawah.

c. Tomato Mosaic virus

   Tomato Mosaic Virus (ToMV) dapat ditularkan melalui benih dan virus menyebar dengan sangat mudah. Ciri dari penyakit ini ialah adanya bintik berwarna terang dan hijau tua pada daun, daun melengkung dan memiliki bentuk daun yang cacat, pada suhu hangat bintik-bintik pada jaringan daun akan terlihat lebih jelas.

d. Early Blight

   Early blight merupakan masalah penyakit yang disebabkan oleh jamur

A. solani. Gejala pada penyakit ini yaitu pada daun terlihat adanya bercak berwarna coklat sampai kehitaman. Bercak membentuk lingkaran kosentris dengan jalus halo berwarna kuning. Pada tingkat serangan berat, bercak membesar berwarna kecoklatan dan kemudian mengering (Kalay et al.).

2.3 Pengolahan Citra

Pengolahan citra merupakan bidang yang bersifat multidisiplin, yang terdiri dari banyak aspek, antara lain: fisika (optik, nuklir, gelombang, dll), elektronika, matematika, seni, fotografi, dan teknologi komputer. Pengolahan citra (*image processing*) memiliki hubungan yang sangat erat dengan disiplin ilmu yang jika sebuah disiplin ilmu dinnyatakan dalam bentuk proses suatu *input* menjadikan *output*, maka pengolahan citra memiliki *input* berupa citra serta *output* berupa citra (Fitri Muwardi, 2017). Pada proses pengolahan citra salah satunya yaitu menkonversikan rgb ke grayscale, rumus yang digunakan dalam menkonversikan rgb menjadi grayscale, sebagai berikut:

$$\text{Grayscale} = 0.33R + 0.333G + 0.333B \quad (2.1)$$

2.4 K-Support Vector Nearest Neighbor (K-SVNN)

K-Support Vector Nearest Neighbor (K-SVNN) merupakan metode reduksi data latih yang didasarkan pada kedua metode yaitu *K-Nearest Neighbor* (KNN) dan *Support Vector Machine* (SVM), dengan prinsip K tetangga terdekat pada setiap data latih. Tidak ada proses clustering yang dilakukan pada sisa data latih yang dihasilkan dan juga belum ada pembobotan pada data latih yang didapatkan sebagai support vector sehingga komputasi pada saat pelatihan menjadi lebih cepat.

K-Support Vector Nearest Neighbor (K-SVNN) termasuk dalam kategori *semi-eiger learning*. Hal ini dikarenakan secara eksplisit ada proses pelatihan yang dilakukan sebelum proses prediksi untuk mendapat sejumlah data latih yang berpengaruh sebagai fungsi tujuan, tetapi harus menyimpan data latih tersebut dalam memori. Maka dari itu, K-SVNN berusaha mengurangi data latih berdasarkan properti skor dan properti relevansi / derajat signifikansi pada setiap data latih berdasarkan prinsip K tetangga terdekat. Setiap data latih mempunyai kedua properti tersebut. Properti skor untuk setiap data latih ada 2 yaitu, nilai kiri(*left value*/LV) dan nilai kanan(*right value*/RV). Nilai yang kiri untuk kelas yang sama, sedangkan nilai yang kanan untuk kelas yang berbeda. Jumlah LV dan RV dari semua data latih sama dengan N×K, seperti dinyatakan oleh persamaan berikut:

$$\sum_{i=1}^{N} LV_i + \sum_{i=1}^{N} RV_i = \text{N×K} \quad (2.2)$$

Significance degree adalah nilai yang menyatakan tingkat signifikansi (relevansi) dari data latih tersebut pada fungsi tujuan (daerah hyperplane). Nilainya dalam rentang 0 sampai 1 [0, 1], semakin tinggi nilainya maka relevansinya untuk menjadi *support vector* (data latih yang digunakan pada saat prediksi) juga semakin tinggi.

Dalam penelitian ini digunakan batas ambang T (threshold) > 0, yang artinya sekecil apapun nilai relevansi akan tetap digunakan sebagai *support vector*. Nilai 0 pada derajat signifikansi sebuah data latih berarti data latih tersebut harus dibuang (tidak digunakan sebagai *support vector).* Nilai *significant Degree* didapatkan dengan membagi LV terhadap RV atau RV terhadap LV sesuai syarat yang terpenuhi pada persamaan (2.3).

$$SD_i = \begin{cases} 0, SV_i = RV_i = 0 \\ \frac{SV_i}{RV_i}, SV_i < RV_i \\ \frac{RV_i}{SV_i}, SV_i > RV_i \\ 1, SV_i = RV_i \end{cases} \quad (2.3)$$

Proses untuk mendapatkan *support vector* dalam K-SVNN merupakan bagian utama dalam proses pelatihan yang dilakukan. Selanjutnya *support vector* tersebut disimpan dalam memori untuk digunakan pada saat prediksi. Algoritma pelatihan K-SVNN dapat dijelaskan sebagai berikut:

1. Inisialisasi: D adalah set data latih, K adalah jumlah tetangga terdekat, T adalah threshold SD yang dipilih, LV dan RV untuk semua data latih, K adalah jumlah tetangga terdekat, T adalah threshold SD yang dipilih, LV dan RV untuk semua data latih = 0.
2. Untuk setiap data latih $d_i \epsilon D$, lakukan langkah 3 sampai 5.
3. Hitung ketidakmiripan (jarak) dari d_i ke data latih yang lain.
4. Pilih d_t sebagai K data latih tetangga terdekat (tidak termasuk d_i).
5. Untuk setiap data latih dalam d_t, jika label kelas sama dengan d_i, maka tambahkan nilai 1 pada LV_i, jika tidak sama maka tambahkan nilai 1 pada *RVi*.
6. Untuk setiap data latih d_i, hitung SD_i menggunakan persamaan (2.3)
7. Pilih data latih dengan SD ≥ T, simpan dalam memori (variabel) sebagai template untuk prediksi.

Properti lain yang juga menentukan kinerja K-SVNN baik pada saat pelatihan maupun pada saat prediksi nilai K. Secara umum, nilai K yang

disarankan untuk digunakan adalah K > 1. Pengaruh besar kecilnya nilai K yang digunakan pada saat pelatihan adalah jika semakin kecil nilai K maka semakin sedikit jumlah *support vector* (semakin besar reduksinya), semakin cepat waktu kerjanya saat prediksi, dan semakin kecil pula akurasi yang didapatkan, dan demikian pula sebaliknya. Akan tetapi, yang perlu diperhatikan nilai K yang besar juga tidak selalu menjamin akurasi kinerja yang bagus (Eko Prasetyo,2014).

2.5 K-Nearest Neighbor (K-NN)

Metode K-Nearest Neighbor (K-NN) menjadi salah satu meode berbasis NN yang paling tua da popular. Nilai K yang digunakan disini menyatakan jumlah tetangga terdekat yang terpilih kemudian dilakukan *voting* kelas dari K tetangga terdekat tersebut. Kelas dengan jumlah suara tetangga terbanyaklah yang diberikan sebagai label hasil prediksi pada data uji tersebut (Tan *et al*, 2005). Algoritma prediksi dengan KNN disajikan, sebagai berikut:

1. Z = (x`, y`) adalah data uji dengan data x` dan label kelas y` yangbelum diketahui.
2. C adalah himpunan label kelas data
3. Hitung jarak d(x`, x), jarak di antara data uji z ke setiap vektor data latih, simpan dalam D.
4. Pilih $D_z \subseteq$ D, yaitu K tetangga terdekat z.
5. y` = $\underset{v \in C}{\arg max} \sum_{yi \in Dz} I(v = yi)$

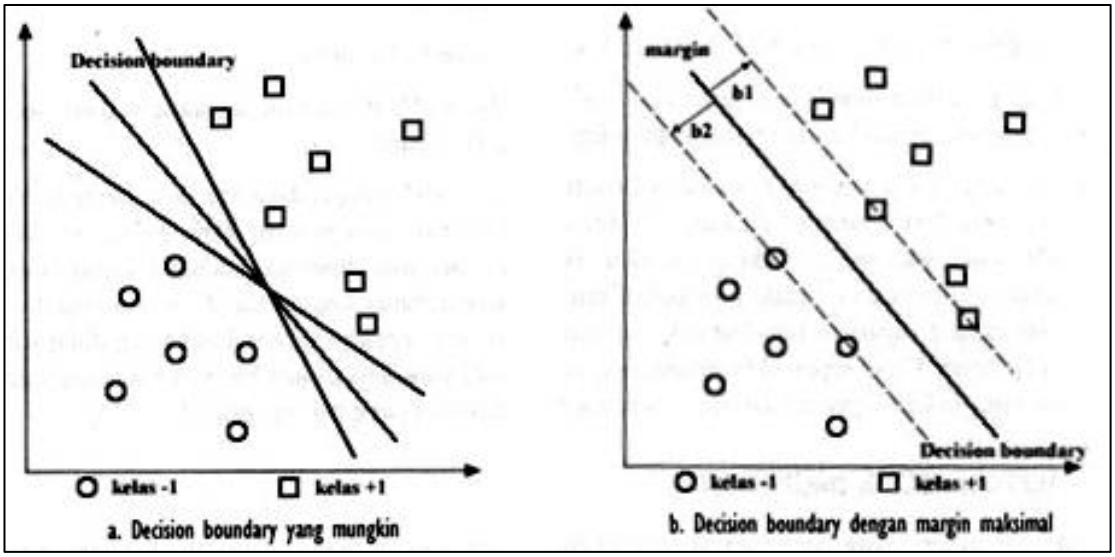
I(.) menyatakan fungsi indicator perbandingan kesamaan kelas, yang memberikan nilai 1 jika argument didalamnya benar, dan nilai 0 jika argument di dalamnya salah (Eko Prasetyo,2014). Dalam mencari jarak dari 2 buah obyek yang terdekat, secara default algoritma KNN menggunakan *Euclidean Distances*. Rumus

perhitungan jarak seperti ditunjukkan pada persamaan (2.4):

$$D_{p,q} = \sqrt{\sum_{i=1}^{n}(p_i \quad q_i)^2} \tag{2.4}$$

2.6 Support Vector Machine (SVM)

Ide dasar SVM adalah memaksimalkan batasan hyperlane, yang diilustrasikan seperti pada Gambar 2.1. Pada gambar (a) ada sejumlah pilihan hyperlane yang mungkin untuk set data, sedangkang pada gambar (b) merupakan hyperlane dengan margin yang paling maksimal. Meskipun sebenarnya pada gambar (a) bisa juga menggunakan hyperlane sembarang, tetapi hyperlane dengan margin yang maksimal akan memberikan generalisasi yang lebih baik pada metode klasifikasi.



Gambar 2.1 Batasan keputusan yang mungkin untuk set data

Konsep klasifikasi dengan menggunakan SVM dapat dijelaskan secara sederhana sebagai usaha dalam mencari hyperlane terbaik yang akan berfungsi sebagai pemisah dari dua buah kelas data pada input space (Nugroho,2007). Gambar 2.1 memperlihatkan beberapa data yang merupakan bagian anggota dari dua biah kelas data, yaitu +1 dan -1. Data yang tergabung pada kelas -1

disimbolkan dengan bentuk lingkaran, sedangkan data pada kelas +1, disimbolkan dengan bentuk bujur sangkar.

Hyperlane (batas keputusan) pemisah terbaik antara kedua kelas dapat ditemukan dengan mengukur margin hyperlane tersebut dan juga mencari titik maksimal pada kedua kelas tersebut. Margin adalah jarak anatara hyperlane tersebut dengan data terdekat dari masing-masing kelas yang ada. Data yang paling dekat ini disebut dengan support vector. Garis solid pada Gambar 6.1 (b) sebelah kanan menunjukkan hyperlane yang terbaik terletak tepat pada tengah-tengah kedua kelas, sedangkan data lingkaran dan bujur sangkar yang dilewati garis batas margin (garis putus-putus) adalah support vector. Usaha untuk mencari lokasi hyperlane ini merupakan inti dari proses pelatihan data pada SVM. (Eko Prasetyo, 2014).

2.7 Confusion Matrix

Confusion matrix adalah suatu alat yang memiliki fungsi untuk melakukan analisis apakah *classifier* tersebut baik dalam mengenali objek dengan kelas yang berbeda. *Confusion matrix* digambarkan dengan tabel yang menyatakan jumlah data uji yang bernilai benar dan jumlah data uji yang bernilai salah dalam proses klasifikasi (M. Fadly Rahman *et al,* 2017).

Tabel 2.2 Tabel *Confusion Matrix*

Correct Classification	*Classified as*	
	Predicted "+"	*Predicted* "-"
Actual "+"	*True Positive* (TP)	*False Negative* (FN)
Actual "-"	*False Positive* (FP)	*True Negative* (TN)

Berdasarkan tabel *Confusion Matrix* diatas

a. *True Positive* (TP) adalah jumlah data dengan nilai sebenarnya positif dan nilai prediksi positif.
b. *False Positive* (FP) adalah jumlah data dengan nilai sebenarnya negatif dan nilai prediksi positif.
c. *False Negative* (FN) adalah jumlah data dengan nilai sebenarnya negatif dan nilai prediksi negatif.
d. *True Negative* (TN) adalah jumlah data dengan nilai sebenarnya positif dan nilai prediksi negatif.

Nilai yang dihasilkan melalui metode *Confusion Matrix* adalah berupa evaluasi sebagai berikut :

a. *Accuracy* adalah presentase jumlah *record* data yang diklasifikasikan (prediksi) secara benar oleh algoritma.

   Accuracy = (TP + TN) / Total data (2.5)

b. *Misclassification (Error) Rate* adalah presentase jumlah *record* data yang diklasifikasikan (prediksi) salah oleh algoritma.

   Misclassification (Error) Rate = (FP + FN) / Total data (2.6)

c. *Precission* adalah presentase jumlah *record* data yang benar dengan cara membandingkan dengan keseluruhan hasil data positif.

   Precission = (TP) / (TP + FP) (2.7)

d. Recall adalah presentase jumlah *record* data yang diklasifikasikan (prediksi) benar oleh algoritma dengan keseluruhan data positif.

   Recall = (TP) / (TP + FN) (2.8)

e. F1 *score* adalah presentase jumlah dari perbandingan rata-rata *precission* dan *recal.*

$$\text{F1 } score = 2 \times (\text{Recall} \times \text{Precission}) / (\text{Recall} + \text{Precission}) \quad (2.9)$$