BAB II

TINJAUAN PUSTAKA

2.1. Universitas Muhammadiyah Semarang

Universitas Muhammadiyah Semarang (UNIMUS) merupakan salah satu perguruan tinggi swasta yang berada di Semarang, Jawa Tengah, Indonesia. UNIMUS didirikan pada tanggal 4 agustus 1999 dengan program studi yang memperoleh ijin operasional pada awal pembukaan tahun 1999 sebanyak 14 program studi. Namun seiring berjalannya waktu UNIMUS telah tumbuh berkembang menjadi tempat pembelajaran yang terpilih. Hingga pada akhirnya UNIMUS membuka beberapa program studi baru, pada tahun 2020 ini Universitas Muhammadiyah Semarang memiliki total 8 fakultas, dengan 4 (empat) program Diploma tiga, 1 (satu) program Diploma empat, 2 (dua) program pascasarjana, 1 (satu) program studi profesi ners, 1 (satu) program studi profesi doctor gigi, 1 (satu) program studi profesi dokter dan 15 program sarjana.

2.2. Mahasiswa

Mahasiswa merupakan seseorang yang sedang dalam proses mencari ilmu ataupun belajar dan terdaftar sedang menjalani pendidikan pada salah satu perguruan tinggi yang terdiri dari akademik, politeknik, sekolah tinggi, institut dan universitas (Hartaji, 2012). Menurut Undang – Undang Republik Indonesia Nomor 12 Tahun 2012 tentang Pendidikan pasal 1 ayat 15 , mahasiswa adalah Peserta didik

pada jenjang Pendidikan Tinggi. Menurut Siswoyo (2007) mahasiswa dapat didefinisikan sebagai individu yang sedang menuntut ilmu ditingkat perguruan tinggi, baik negeri maupun swasta atau lembaga lain yang setingkat dengan perguruan tinggi. Mahasiswa dinilai memiliki tingkat intelektualitas yang tinggi, kecerdasan dalam berpikir dan kerencanaan dalam bertindak. Berpikir kritis dan bertindak dengan cepat dan tepat merupakan sifat yang cenderung melekat pada diri setiap mahasiswa, yang merupakan prinsip yang saling melengkapi.

2.3. Kelulusan Mahasiswa

Berdasarkan ketetapan Kementerian Pendidikan dan Kebudayaan Direktorat Jenderal Pendidikan Tinggi tentang Sistem Pendidikan Tinggi disebutkan bahwa untuk memenuhi standar kompetensi lulusan bagi mahasiswa program sarjana (S1) beban wajib yang harus ditempuh adalah paling sedikit 144-160 satuan kredit semester (sks) dengan masa studi selama 8-12 semester atau 4-6 tahun. Sedangkan Waktu lama studi untuk jenjang D3 lama studinya yaitu 6 semester (36 bulan), untuk jenjang S1 lama studinya yaitu 8 semester (48 bulan), dan untuk jenjang S2 lama studinya yaitu 4 semester (24 bulan). Seorang mahasiswa dinyatakan lulus program apabila telah menyelesaikan minimal SKS sesuai dengan kurikulum masing-masing Program Studi dengan Indeks Prestasi Kumulatif (IPK) minimal 2.00 dan menyelesaikan Tugas Akhir dan/atau skripsi dan telah mempublikasi-kan karya ilmiah untuk program S1 atau telah menyelesaikan Tugas Akhir atau Karya Tulis. Setelah melakukan Evaluasi Akhir Studi, Ketua Program Studi melaporkan kepada Dekan mahasiswa yang telah memenuhi persyaratan untuk lulus untuk

diajukan pengesahan kelulusannya kepada Rektor. Rektor menerbitkan ijazah bagi mahasiswa yang telah dinyatakan lulus. Seorang mahasiswa dinyatakan lulus berhak memakai gelar sarjana untuk Program Strata Satu (Erene Fajrila, 2018). Indeks Prestasi Kelulusan sebagai dasar penentuan predikat kelulusan ditentukan sebagai berikut :

1. IPK 2.00-2.75 : Lulus dengan predikat cukup.
2. IPK 2.76-3.00 : Lulus dengan predikat memuaskan.
3. IPK 3.01-3.05 : Lulus dengan predikat sangat memuaskan.
4. IPK 3.51-4.00 : Lulus dengan predikat cumlaude.

2.4. Klasifikasi

Menurut Bramer (2007) klasifikasi merupakan tugas yang sering terjadi dalam kehidupan sehari-hari. Pada dasarnya klasifikasi membagi objek kesalah satu dari sejumlah kategori yang disebut kelas, sedangkan menurut Han dkk (2012) klasifkasi adalah proses menemukan sebuah model atau fungsi yang menggambarkan dan membedakan kelas data atau konsep. Model didasarkan pada 10 analisis set data training yaitu objek data untuk label kelas yang sudah diketahui, model ini digunakan untuk memprediksi kelas label dari objek label kelas yang belum diketahui. Masalah prediksi merujuk pada masalah dimana variabel yang akan diprediksi memiliki jumlah nilai yang terbatas yaitu variabel kategorik.

Dalam pengklasifikasian data terdapat dua proses yang akan dilakukan yaitu:

1. Proses *Training*

   Pada proses ini digunakan training set yang telah diketahui labelnya untuk membangun model atau fungsi.

2. Proses *Testing*

   Pada proses ini untuk mengetahui akurasi model atau fungsi yang akan dibangun pada proses training, maka digunakan data yang disebut dengan testing set untuk memprediksi labelnya.

Metode dalam klasifikasi terbagi atas dua yaitu metode klasifikasi secara statistical learning dan metode klasifikasi secara *machine learning*.

2.5. Klasifikasi Machine Learning

Machine learning merupakan kecerdasan buatan dengan tujuan memungkinkan mesin untuk melakukan pekerjaan dengan terampil menggunakan perangkat lunak cerdas. *Machine learning* adalah kombinasi dari pembelajaran statistik dan komputer, pembelajaran statistik merupakan tulang punggung dari perangkat lunak cerdas yang digunakan untuk mengembangkan kecerdasan mesin. Karena algoritma pembelajaran mesin memerlukan data untuk dipelajari, sehingga harus memiliki koneksi dengan disiplin *database* (Mohammed dkk, 2017). Ada dua jenis klasifikasi *machine learning* yaitu:

1. *Supervised Learning* adalah jenis klasifikasi *machine learning* yang paling mudah diantara jenis klasifikasi yang lain yaitu komputer diberikan contoh input dan output yang diinginkan, dan komputer harus mencari tahu

hubungan atau aturan umum di antara keduanya. Label dari data telah disediakan.

2. *Unsupervised Learning* adalah pembelajaran yang tidak terstruktur, tidak ada pedoman yang diberikan untuk algoritma, dan komputer harus menguraikan input untuk setiap struktur atau pola. Tidak ada label yang disediakan, seperti data mana yang "benar" atau "salah".

Terdapat beberapa tipe algoritma dalam *machine learning* yaitu regresi linier, regresi logistik, *decision tree*, *naive bayes*, *k-nearest neighbor* (KNN), *random forest* dan *support vetor machine* (SVM) (Attaran & Deb, 2018).

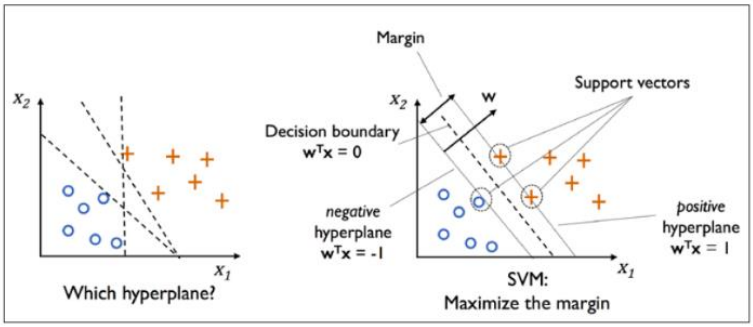
2.6. Support Vector Machine (SVM)

Pada tahun 1992 untuk pertama kali SVM diperkenalkan oleh Vapnik sebagai rangkaian konsep unggulan pada bidang *pattern recognition*. Usia SVM sebagai salah satu metode *pattern recognition* masih terbilang relatif muda. Dewasa ini SVM meruapakn salah satu metode yang berkembang pesat. SVM merupakan salah satu metode *machine learning* yang bekerja atas prinsip *Structural Risk Minimization* (SRM) yang bertujuan untuk menemukan *hyperplane* terbaik yang dapat memisahkan kelas-kelas pada *input space*.

SVM merupakan salah satu metode klasifikasi dalam data *mining*. SVM juga dapat melakukan prediksi baik pada klasifikasi maupun regresi (Santosa, 2007). Pada dasarnya SVM memiliki prinsip linear, akan tetapi kini SVM telah berkembang sehingga dapat bekerja pada masalah *non*-linear. Cara kerja SVM pada masalah *non*-linear adalah dengan memasukkan konsep kernel pada ruang

berdimensi tinggi. Pada ruang yang berdimensi ini, nantinya akan dicari pemisah atau yang sering disebut *hyperplane*. *Hyperplane* dapat memaksimalkan jarak atau *margin* antara kelas data. *Hyperplane* terbaik antara kedua kelas dapat ditemukan dengan mengukur *margin* dan kemudian mencari titik maksimalnya. Usaha dalam mencari *hyperplane* yang terbaik sebagai pemisah kelas-kelas adalah inti dari proses pada metode SVM (Cristianai, 2000).

Support Vector Machine (SVM) dikembangkan oleh Boser, Guyon, dan Vapnik pertama kali dipresentasikan pada tahun 1992 di *Annual Workshop on Computational Learning Theory*. Konsep dasar SVM merupakan kombinasi dari teori-teori komputasi yang telah ada pada tahun-tahun sebelumnya seperti margin hyperplane (Vapnik & N, 1999). Cara kerja dari SVM adalah dengan menemukan suatu bidang pemisah yang disebut dengan *hyperplane* terbaik yang membagi data menjadi 2 kelas yang berbeda. Pada perkembangannya SVM dapat diperluas untuk klasifikasi lebih dari dua kelas atau mutlti kelas. Berbeda dengan ANN, SVM akan menentukan *hyperplane* paling optimum dimana hyperplane dikatakan optimum apabila berada tepat ditengah-tengah kedua kelas sehingga memiliki jarak paling jauh ke data-data terluar di kedua kelas.



Gambar 2. 1 Contoh Hyperplane

Hyperplane yang ditemukan SVM diilustrasikan seperti Gambar 2.2 posisinya berada ditengah-tengah antara dua kelas, artinya jarak antara *hyperplane* dengan objek-objek data berbeda dengan kelas yang berdekatan (terluar) yang diberi tanda bulat kosong dan positif. Dalam SVM objek data terluar yang paling dekat dengan *hyperplane* disebut *support vector*. Objek yang disebut *support vecto*r paling sulit diklasifikasikan dikarenakan posisi yang hampir tumpang tindih (*overlap*) dengan kelas lain. Mengingat sifatnya yang kritis, hanya *support vector* inilah yang diperhitungkan untuk menemukan *hyperplane* yang paling optimal oleh SVM (Suyanto, 2017).

Suatu *hyperplane* pada SVM dapat dinotasikan dengan:

$$w.x + b = 0 \quad (2.1)$$

Dimana w adalah sebuah bobot suatu vector dan b adalah nilai bias. Konsep dari kerja SVM yaitu dengan mencari *hyperplane* terbaik untuk memisahkan dua kelas, yaitu kelas +1 (positif) dan kelas -1 (negatif). Fungsi klasifikasi pada SVM didefinisikan dengan persamaan:

$$f(x) = sign\ (w.x + b) \quad (2.2)$$

Apabila nilai dari w.x+b > 0 maka data akan diklasifikasikan kedalam kelas +1 (positif) dan apabila nilai dari w.x+b < 0 maka data akan diklasifikasikan kedalam kelas -1 (negatif). Seperti yang telah dijelaskan sebelumnya, *hyperplane* terbaik adalah *hyperplane* yang mempunyai margin paling besar atau maksimum. Margin maksimum dapat didapatkan dengan memaksimalkan jarak antara hyperplane dengan titik terdekatnya yaitu $\frac{1}{\|w\|}$.

Permasalahan seperti ini biasa disebut dengan *Quadratic Programming* (QP), yaitu dengan menemukan titik minimal dari:

$$\min \tau(w) = \frac{1}{2}||\mathrm{w}||^2 \quad (2.3)$$

Dengan batasan *constraint*:

$$y_i(x_i.w + b) - 1 \geq 0, i = 1,2,3, \ldots, l \quad (2.4)$$

Permasalahan seperti ini dapat diselesaikan dengan *lagrange multiplier* dengan persamaan:

$$\mathrm{L(w, b, a)} = \frac{1}{2}||\mathrm{w}||^2 - \sum_{i=1}^{l} a_i\, y_i(x_i.w + b) - 1, i = 1,2,3, \ldots, l \quad (2.5)$$

Dimana $a_i \geq 0$ adalah nilai koefisien *lagrange multipliers*. Selanjutnya meminimumkan nilai L terhadap w dan b, sehingga diperoleh:

$$\sum_{i=1}^{l} a_i\, y_i = 0 \quad (2.6)$$

$$w = \sum_{i=1}^{l} a_i\, y_i x_i \quad (2.7)$$

Selanjutnya persamaan 2.7 dimodifikasi sebagai maksimalisasi problem yang mengandung nilai a_i sehingga diperoleh persamaan:

$$\mathrm{L(a)} = \sum_{i=1}^{l} a_i - \frac{1}{2}\sum_{i=i,j=i} a_i a_j\, y_i y_j x_i x_j$$

$$\sum_{i=1}^{n} a_i y_i = 0\,, a_i \geq 0 \quad (2.8)$$

Dari proses yang telah dilakukan, maka akan diperoleh $a_i > 0$ yang disebut dengan *support vector* dan sisanya memiliki nilai $a_i = 0$. Fungsi keputusan yang dihasilkan hanya dipengaruhi oleh nilai *support vector*.

Pada umumnya permasalahan-permasalahan di dunia nyata sangat jarang ditemukan data terpisah secara *linear*, sehingga dalam menyelesaikan permasalahan *nonlinear* SVM menggunakan fungsi kernel. Kernel memetakan

sampel data kedalam ruang dimensi yang lebih tinggi sehingga dapat menyelesaikan kasus dengan hubungan antara kelas dan atributnya tidak linear. SVM mempunyai 4 kernel yang dapat digunakan untuk menyelesaikan permasalahan linear dan nolinear diantaranya adalah:

1. Kernel Linear

$$K(x, x_k) = x_k^T x \tag{2.9}$$

2. Kernel Polynomial

$$K(x, x_k) = (x_k^T x + 1)^{\mathrm{d}} \tag{2.10}$$

3. Kernel Gaussian Radial Basis Function (RBF)

$$K(x, x_k) = \exp\{-||\mathrm{x} - x_k||_2^2 / \sigma^2\} \tag{2.11}$$

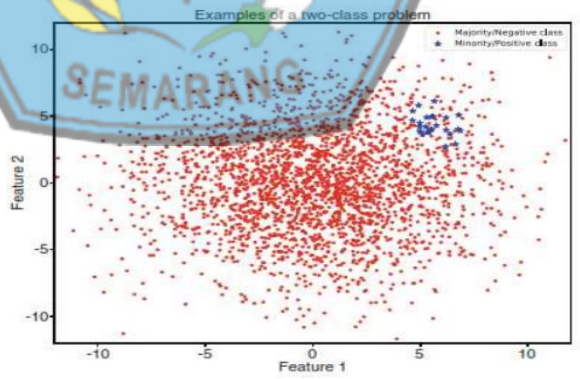
4. Sigmoid

$$K(x, x_k) = \tanh[KX_k^T x + \theta] \tag{2.12}$$

Nilai akurasi dari model yang dihasilkan dengan menggunakan SVM sangat tergantung pada fungsi kernel dan nilai parameter yang digunakan. Parameter yang digunakan pada algoritma SVM adalah parameter *cost (C)* dan *gamma (γ).* Semakin besar nilai C akan menghasilkan penalty yang besar pula terhadap klasifiikasinya. Pada fungsi kernel RBF parameter *gamma (γ)* digunakan untuk metransformasikan data train ke ruang fitur yang kemudian dioptimasi menggunakan metode *Lagrange Multipliers* sehingga menghasilkan nilai α yang digunakan untuk menentukan *support vector* dan memperkirakan koefisien w (bobot) ataupun b (bias) pada model klasifikasi (Handayani dkk, 2017).

2.7. Imbalanced Data

Permasalahan *imbalanced class* banyak terjadi pada klasifikasi data dalam dunia nyata dan masalahnya terdiri dari jumlah disproporsi kelas yang berbeda. Imbalanced merupakan suatu kondisi dimana jumlah dari data yang mempresentasikan satu kelas (minoritas) sangatlah kecil dibandingkan kelas lain (mayorits). Sebagai ilustrasi misalkan sebuah data imbalanced dengan rasio 1:100 untuk masing-masing objek kelas minoritas, terdapat 100 objek kelas mayoritas (Pangastuti, 2018). *Classifier* standar yang digunakan untuk mengklasifikaskan permasalahan tersebut untuk melihat akurasinya, dimana akurasinya akan mencapai 99% dengan mengabaikan kelas minoritas sehingga semua kelas akan dianggap sebagai mayoritas. Masalah ini menghambat performance klasifikasi karena desainnya yang berorientasi pada akurasi, yang biasanya membuat kelas minoritas diabaikan.



Gambar 2. 2 Imbalanced Data

(Sumber: Fernandez dkk, 2018)

Pada Gambar 2.3 terlihat bahwa jumlah kelas minoritas yang berwarna biru lebih sedikit dibandingkan dengan jumlah kelas mayoritas yang berwarna merah.

Beberapa teknik yang telah dikembangkan untuk mengatasi masalah *imbalanced data*, teknik ini dikategorikan kedalam empat kelompok yaitu sebagai berikut (Fernandez dkk, 2018):

1. *Algorithm Level* bekerja dengan mencoba untuk menyesuaikan algoritma dengan memperhitungkan kelas minoritas (positif).
2. Data Level dengan menyeimbangkan distribusi kelas dengan resampling ruang data.
3. *Cost-Sensitive* adalah menggabungkan algorithm level dan data level dengan mengasumsikan kesalahan biaya klasifikasi pada kelas minoritas dan berusaha meminimalkan total kesalahan biaya dari dua kelas.
4. *Ensemble Method* adalah kombinasi antara algoritma *ensemble* dan salah satu teknik sebelumnya, khusunya data level dan cost-sensitive.

Beberapa metode yang disarankan berdasarkan teknik algoritma dalam mengatasi masalah imbalanced yaitu kombinasi data level dan *ensemble method* merupakan kombinasi dari *Synthetic Minority Oversampling Technique* (SMOTE) dan algoritma *Bagging*, *Boosting* yang dapat meningkatkan akurasi pada proses klasifikasi.

2.8. Synthetic Minority Oversampling Technique (SMOTE)

Teknik SMOTE merupakan salah satu metode oversampling yang bekerja dengan menerapkan metode sampling untuk meningkatkan jumlah kelas minoritas (positif) melalui replikasi data secara acak, sehingga sejumlah data minoritas sama

dengan data mayoritas. Algoritma SMOTE pertama kali diperkenalkan oleh Nithes V. Chawla (2002), ide utama pendekatan ini bekerja dengan membangun data 23 *synthetic* replikasi data minor. Algoritma SMOTE dilakukan untuk mendefinisikan *k-nearest neighbor* (KNN) untuk setiap kelas minoritas, kemudian menyusun duplikasi data synthetic sebanyak persentase yang diinginkan antara kelas minoritas dan KNN yang dipilih secara acak (Sain & Purnami, 2015).

Gambar 2. 3 Contoh Pembangkit SMOTE

(Sumber: Fernandez dkk, 2018)

Pada Gambar 2.4 terlihat bahwa *xi* adalah data ke-i dari kelas minoritas yang dipilih sebagai dasar untuk membangkitkan data *synthetic* baru, sehingga didapatkan *synthetic* baru yaitu $r1$ sampai $r4$. Chawla dkk (2002) menunjukkan bahwa pendekatan SMOTE dapat meningkatkan akurasi klasifikasi untuk kelas minoritas. Kombinasi SMOTE dan metode *ensemble* memiliki *performance* yang lebih baik daripada metode SMOTE standar.

2.9. Boostrap Aggregating (Bagging)

Breiman memperkenalkan konsep *bootstrap aggregating* dengan tujuan untuk memanipulasi data training dengan mengganti data training asli T secara acak menjadi N item (Yongqing dkk, 2014). Algoritma ini juga digunakan untuk

membangun model *ensembles*. Model ensemble ini terdiri dari beberapa pelatihan pengklasifikasian dengan *bootstrap* replika yang berbeda dari training dataset asli untuk melatih setiap classifier. Dataset baru dibentuk secara acak (dengan penggantian) objek dari dataset asli. Ketika sebuah objek yang tidak diketahui akan diklasifikasikan maka voting mayoritas atau bobot yang akan digunakan untuk mendapatkan kelasnya. Voting mayoritas dilakukan dengan menggunakan confidence yang diberikan oleh masing-masing pengklasifikasian dalam prediksi. Salah satu kelebihan bagging adalah kesederhanaannya. Selain itu, *bagging* dapat mengurangi varians. Karena efek voting mirip dengan rata-rata dalam regresi di mana pengurangan *overfitting* menjadi lebih mudah untuk diatasi. Didalam kasus klasifikasi, algoritma klasifikasi akan membentuk *classifier* $H{:}D \rightarrow \{-1, +1\}$ sebagai dasar dari data training. Metode ini membuat urutan classifier Ht , dimana $t = 1, \cdots, T$ sebagai modifikasi dari data training. Classifier tersebut kemudian digabungkan sehingga menjadi satu classifier. Hasil prediksi classifier gabungan kemudian diberikan sebagai bobot gabungan classifier individu atau disebut prosedur voting (Fernandez dkk, 2018):

$$H(x) = sign\left(\sum_{t=1}^{T} h_t(x)\right) \qquad (2.13)$$

Dengan :

$H(\mathbf{x})$ = Hasil pengklasifikasian akhir atau kuat

T = Jumlah ukuran resample model bagging

$h_t(x)$ = *Base classifier* atau pengklasifikasian lemah

Peneliti yang mengkaji tentang bagging yaitu Patankar dan Chavda (2015) menunjukkan efek *bagging* pada akurasi klasifikasi dengan menggunakan

pengklasifikasian yang berbeda. Percobaan dilakukan dengan menggunakan klasifikasi CART sebagai weak learner dan menunjukkan efek bagging pada berbagai pengklasifikasian dasar. Untuk tambahan diamati untuk tiga dataset, akurasi klasifikasi meningkat ketika menggunakan ensemble dan bukan classifier biasa. Singkatnya, teknik *ensemble bagging* membantu meningkatkan akurasi klasifikasi. Tetapi untuk hasil yang lebih baik dapat dilakukan pada pengkombinasian dengan metode lain untuk lebih meningkatkan akurasi. Machova dkk (2006) menunjukkan bahwa metode bagging adalah cara yang cocok untuk meningkatkan efisiensi *algoritma machine learning*. Jumlah minimum pengklasifikasian yang diperlukan untuk mencapai efisiensi ini dapat ditemukan.

2.10. Boosting

*Boostin*g adalah salah satu metode ensemble untuk meningkatkan performance pada suatu algoritma dengan mengkombinasikan classifier yang lemah menjadi classifier yang kuat. Ide utama didalam proses boosting adalah memilih sekumpulan data training dengan beberapa cara untuk kemudian dipelajari oleh suatu base learner, dimana *base learner* dipaksa menarik sesuatu yang baru tentang sampel tersebut setiap kali base learner itu dipanggil. Prinsip kerja boosting yaitu mempekerjakan sekumpulan *classifier* yang dilatih secara iteratif. Salah satu algoritma boosting yang paling populer adalah *Adaptive Boosting* (AdaBoost) yang diperkenalkan oleh Freund & Schapire (1995). AdaBoost mempertahankan sekumpulan bobot pengamatan pada saat training dan secara adaptif dapat menyesuaikan bobot-bobot diakhir ukuran resample *boosting*. Bobot dari

pengamatan yang salah diklasifikasikan akan ditingkatkan sedangkan bobot pengamatan yang benar diklasifikasikan akan dikurangi nilainya sehingga memaksa base learner untuk memberikan perhatian lebih terhadap sampel-sampel tersebut. Tugas dari base learner yaitu menemukan sebuah *weak hypothesis* ht : **X** → {−1, +1} sesuai distribusi Dt. Kebaikan dari suatu weak hypothesis diukur berdasarkan error (Pratama,2018):

$$\varepsilon_t = Pr_{i\sim D_t}[h_t(x_i) \neq y_i]$$
$$= \sum_{i,h_t(x_i)\neq y_i} D_t(i) \qquad (2.14)$$

Dengan:

εt = error dari kebaikan weak hypothesis

Dt = Distribusi untuk ukuran resample ke-t

ht (**x**i) = Base classifier atau pengklasifikasian lemah ke-i

yi = Data training dimana setiap $yi \in$ y = {−1, +1}

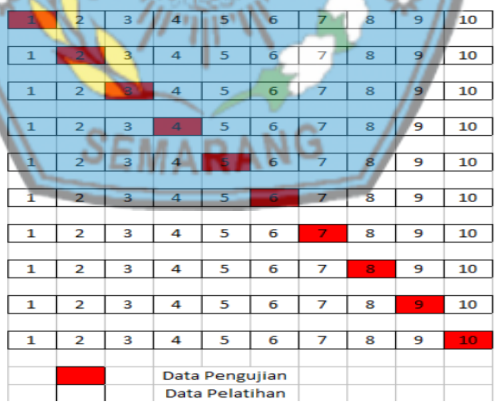
Dt (i) = Bobot sampel untuk training ke-i

Peneliti yang mengkaji tentang Adaboost yaitu Kewen dkk (2019) menunjukkan bahwa algoritma AdaBoost standar berfokus pada sampel yang salah diklasifikasi dan bukannya pada sampel dari kelas minoritas. Penelitian ini mengusulkan untuk meningkatkan algoritma AdaBoost. Nilai AUC dapat secara efektif mencerminkan *performance classifier*. AUC merupakan perhitungan kesalahan, menjadikan AdaBoost lebih berfokus pada akurasi klasifikasi minoritas. Chengseng dkk (2017) menunjukkan bahwa AdaBoost memiliki keunggulan dengan kecepatan, sederhana dalam pengoperasian dan mudah diprogram. Tidak perlu menyesuaikan parameter kecuali untuk jumlah ukuran resample. Dapat

dikombinasikan secara fleksibel dengan metode apa pun untuk mencari hipotesis yang lemah tanpa perlu pengetahuan dari weak learner.

2.11. K-Fold Validation

Teknik *K-Fold Cross Validation* adalah salah satu metode yang digunakan untuk mempartisipasi data menjadi data *training* dan data *testing*. Pada penelitian ini digunakan metode tersebut karena dapat mengurangi bias yang terjadi dalam pengambilan sampel. Metode ini digunakan secara berulang-ulang membagi data menjadi dua yaitu data training dan testing, setiap data memperoleh kesempatan menjadi data *testing* (Ibrahim, N. dan Wibowo, A. 2014). K disini merupakan besar angka partisi data yang akan digunakan untuk pembagian antara *training* dan *testing*. Berikut ilustrasi pembagian data menggunakan *k-fold cross validation.*



Gambar 2.5 Ilustrasi Pembagian Data

2.12. Evaluasi Performance Metode

Kriteria evaluasi merupakan hal yang paling penting dalam penilaian performance klasifikasi. Cara yang paling mudah untuk mengetahui performance klasifikasi dengan melakukan tabulasi silang antara kelas aktual dan prediksi. Hasil dari tabulasi silang, disebut *confusion matrix*. Pada masalah dua kelas, *confusion matrix* digunakan sebagai informasi bahwa pada kolom menjelaskan jumlah dari kelas prediksi dan pada baris menjelaskan jumlah dari kelas aktual (Han dkk, 2012).

Tabel 2. 1 Confussion Matrix

Confusion Matrix		**Predicted Class**	
		Positive	***Negative***
Actual Class	***Positive***	*TP (True Positive)*	*FN (False Negative)*
	Negative	*FP (False Positive)*	*TN (True Negative)*

Sumber: Chawla dkk, 2003

1. *True Positive* (TP) menunjukkan bahwa kelas yang dihasilkan dari prediksi klasifikasi adalah positif dan kelas aktualnya adalah positif.
2. *True Negative* (TN) menunjukkan bahwa kelas yang dihasilkan dari prediksi klasifikasi adalah negatif dan kelas aktualnya adalah negatif.
3. *False Positive* (FP) menunjukkan bahwa kelas yang dihasilkan dari prediksi klasifikasi adalah negatif dan kelas aktualnya adalah positif.
4. *False Negative* (FN) menunjukkan bahwa kelas yang dihasilkan dari prediksi klasifikasi adalah positif dan kelas aktualnya adalah negatif.

Dalam imbalanced, hasil klasifikasi akan mencapai akurasi tinggi karena hanya berfokus pada kelas mayoritas. Jelas bahwa akurasi tidak cukup kuat jika dijadikan sebagai tolak ukur untuk ukuran kriteria performance. Sehingga untuk

mengukur performa algoritma pada kelas minoritas digunakan kriteria sensitivity

$$Recall\ /sensitivity\ = \frac{TP}{TP+FN}\, x\ 100\% \tag{2.15}$$

$$Specificity = \frac{TN}{TN+FP}\ x\ 100\% \tag{2.16}$$

Untuk melakukan evaluasi secara keseluruhan, dapat digunakan kriteria seperti *Geometric Mean (G-mean)* dan analisis AUC.

$$G - mean = \sqrt{Sensitivity * Specificity} \tag{2.17}$$

$$FPR = 1 - Specificity \tag{2.18}$$

$$AUC = \frac{1+sensitivity-FPR}{2} \tag{2.19}$$

Geometric Mean (G-mean) adalah rata-rata *geometrik sensitivity* dan *specificity*. G-mean akan bernilai tinggi apabila TP dan TN tinggi dan perbandingan antara TP dan TN kecil. Apabila semua kelas positif tidak dapat diprediksi maka G-mean akan bernilai nol sehingga diharapkan suatu algoritma klasifikasi mencapai nilai G-mean yang tinggi.