Seminar II

KOMBINASI TEKNIK RESAMPLING DAN ALGORITMA MACHINE LEARNING PADA KELAS TAK SEIMBANG

Oleh :

ZINEDINE KAHLIL GIBRAN ZIDANE

H13116304

Pembimbing Utama : **Dr. Amran, S.Si., M.Si.**

Pembimbing Pertama : **Supri Bin Hj Amir, S.Si., M.Eng.**

Penguji : **1. Dr. Anna Islamiyati, S.Si, M.Si.**

2. Nur Hilal A Syahrir, S.Si, M.Si.

PROGRAM STUDI ILMU KOMPUTER

DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

2019

Daftar Isi

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam beberapa tahun terakhir, perkembangan yang pesat dalam sains dan teknologi telah berdampak pada pertumbuhan data mentah secara eksponensial (He & Garcia, Learning from Imbalanced Data, 2008). Berdasarkan World Economic Forum (Desjardins, 2019), data digital di dunia akan mencapai 44 zettabyte, atau 44 triliun gigabyte pada tahun 2020. Jumlah ini akan terus berkembang hingga lebih dari dua kali lipat setiap dua tahunnya (Chen, Mao, & & Liu, 2014; Lohr, 2012).

Dari pertumbuhan data tersebut, kebutuhan untuk menganalisis data terus meningkat (Elgendy & Elragal, 2014). Menurut laporan dari LinkedIn pada tahun 2018, permintaan pekerjaan yang membutuhkan analisis data berkembang hingga 12 kali lipat lebih banyak dari tahun 2014 (LinkedIn Economic Graph Team, 2018). Ini disebabkan karena data merupakan kunci dari setiap organisasi, institusi atau perusahaan untuk mengambil keputusan yang lebih cermat dan efektif (McAfee, Brynjolfsson, Davenport, Patil, & Barton, 2012).

Namun data mentah atau *raw data* tidak dapat memberikan informasi yang berguna. Data tersebut perlu diolah dan dianalisis menjadi informasi yang berguna. *Raw data* yang dimaksud adalah data yang belum diproses, yang terstruktur dan yang tidak terstruktur. Data yang terstruktur adalah data yang disimpan dengan format yang telah ditentukan seperti database (Beal, 2019), di mana atribut data dapat dibedakan dengan jelas sehingga dapat langsung diproses oleh peralatan komputasi (Baars & Kemper, 2008). Sedangkan data tidak terstruktur adalah data yang disimpan dalam format yang tidak terstruktur, sehingga membutuhkan campur tangan manusia agar dapat diinterpretasi oleh mesin; seperti dokumen, gambar, video dan audio (Weglarz, 2004). Menurut laporan Beal (Beal, 2019), 80% hingga 90% data di dunia tidak terstruktur.

Dengan jumlah data yang sangat banyak, tidak mungkin oleh manusia untuk menganalisis dan membuat perhitungan mengenai data secara manual. Maka dari itu diperlukan bidang khusus untuk mengolah dan menganalisis data. *Data Science* adalah bidang yang mempelajari bagaimana meng-ekstraksi *raw data* menjadi *meaningful information* atau informasi yang berguna (Berman, et al., 2018; Dhar, 2012). Data Science melibatkan prinsip, proses, dan teknik untuk memahami fenomena melalui data (Provost & Fawcett, 2013). Data Science merupakan bidang yang sangat luas dan sedang dikembangkan (Provost & Fawcett, 2013), namun salah satu bidang khusus dari data science adalah *machine learning* yang merupakan paduan antar *computer science* (ilmu komputer) dan *statistics* (statistika) (Jordan & Mitchell, 2015). *Machine learning* membahas mengenai bagaimana membangun sistem komputer yang dapat belajar melalui pengalaman tanpa harus diprogram secara spesifik dan manual (Jordan & Mitchell, 2015) (Domingos, 2011).

Untuk menyelesaikan suatu masalah pada komputer, dibutuhkan algoritma. Namun untuk beberapa masalah, tidak ada algoritma yang memadai. Contohnya adalah bagaimana komputer mengenali angka dalam bentuk tulisan tangan (Bishop, Neural Networks for Pattern Recognition, 1995) dan mengklasifikasi suatu email menjadi spam atau bukan spam (Ethem, 2009). Dalam masalah seperti ini, seluruh objek dengan label atau kelas berbeda dikumpulkan dan diidentifikasi ciri-ciri yang membedakan mereka. Di dalam hal ini lah *machine learning* bekerja.

Data yang dikumpulkan biasanya dalam bentuk dataset atau tabel, di mana setiap kolomnya adalah atribut atau ciri-ciri dan setiap barisnya adalah instansi atau observasi. Dataset tersebut ada yang memiliki kolom label, kelas, atau kolom yang berisi informasi mengenai kategori dari setiap observasi (contohnya, spam atau bukan spam), dan ada juga dataset yang tidak memiliki kolom label, di mana isinya hanyalah atribut atau ciri-ciri dari setiap observasi, tanpa mengindikasikan kategori dari tiap observasi (Bishop, Neural Networks for Pattern Recognition, 1995). Pembelejaran pada dataset berlabel disebut *supervised learning*. Kasus di mana tujuannya adalah

mengklasifkasikan input data ke suatu kategori diskrit tertentu disebut *klasifikasi*, dan kasus di mana outputnya adalah suatu variabel kontinu disebut *regresi*. Selain itu, pembelajaran pada dataset tanpa label atau acuan kategori yang benar disebut *unsupervised learning*. Kasus *unsupervised learning* di mana tujuannya adalah mengelompokkan observasi-observasi yang mirip disebut *clustering*, jika menentukan distribusi data pada input disebut *estimasi kepadatan*. Dan yang terakhir, pembelajaran di mana mesin dilatih untuk membuat keputusan tertentu dengan cara *trial and error* disebut *reinforcement learning* (Bishop, Pattern Recognition and Machine Learning, 2006). Masing-masing jenis pembelajaran memiliki banyak algoritma yang telah dikembangkan dengan berbagai pendekatan yang berbeda-beda (Jordan & Mitchell, 2015). Berdasarkan dokumentasi dari *scikit-learn* (Pedregosa, et al., 2011), terdapat lebih dari 100 algoritma *machine learning* yang ada.

Untuk distribusi data pada suatu dataset, terdapat istilah kelas yang terdistribusi secara seimbang (*balanced*) dan secara tak seimbang (*imbalanced*). Dataset dengan kelas yang seimbang berarti jumlah observasi untuk setiap kelas tidak jauh dari kelas-kelas yang lain (Galar, Fernandez, Barrenechea, Bustince, & Herrera, 2011). Sedangkan untuk dataset dengan distribusi kelas yang tak seimbang, jumlah suatu observasi pada kelas tertentu sangat jauh berbeda dengan kelas yang lain. Hal ini berlaku pada dataset dengan kelas biner (dua kelas saja) dan juga *multiclass* (lebih dari dua kelas) (He & Garcia, Learning from Imbalanced Data, 2008). Kelas dengan jumlah observasi sedikit disebut kelas minoritas (*minority class*) dan kelas dengan jumlah observasi yang sangat banyak disebut kelas mayoritas (*majority class*). Tidak jarang suatu dataset terdistribusi secara tak seimbang dengan proporsi antara kelas minoritas dan mayoritasnya adalah 1:100, 1:1000, atau 1:1000 (He & Garcia, Learning from Imbalanced Data, 2008). Sebagian besar data asli di dunia terdistribusi secara tak seimbang (He & Garcia, Learning from Imbalanced Data, 2008; Kotsiantis, Kanellopoulos, & Pintelas, 2006; Kumar & Sheshadri, 2012; Visa & Ralescu, 2005).

Pada umumnya, algoritma-algoritma *machine learning*, dalam hal ini pada masalah klasifikasi, bekerja dengan tujuan utama memaksimalkan akurasi (Provost F. , 2000). Hal ini sangat masuk akal, karena akurasi yang tinggi menjelaskan bahwa model algoritma tersebut melaksanakan tugasnya dengan baik, mengklasifikasikan kelas data dengan benar dengan sedikit kesalahan. Namun, akurasi hanya memberikan informasi secara umum, bagaimana jika model algoritma tersebut bekerja pada dataset tak seimbang, dan hanya mampu mengklasifikasikan kelas mayoritas dengan benar tetapi tak mampu mengklasifikasikan kelas minoritas? Jika perbandingan antara kelas minoritas dan mayoritas saja satu berbanding seratus, maka akurasi yang akan diperoleh lebih besar dari 99%, dengan kesalahan lebih kecil dari 1% yang hampir seluruhnya adalah kelas minoritas. Masalah ini memberi bias terhadap performa algoritma-algoritma klasifikasi, terutama jika kelas yang lebih utama untuk diklasifikasikan dengan benar adalah kelas minoritas, seperti email spam, diagnosis penyakit di bidang kedokteran, deteksi kartu kredit palsu dan lain-lain (Visa & Ralescu, 2005; Rahman & Davis, 2013). Hal ini menunjukkan bahwa pada kasus dataset tak seimbang, dibutuhkan perhatian lebih terhadap *preprocessing* data sebelum dimasukkan ke model.

Banyak cara yang telah ditemukan untuk mengatasi dataset tak seimbang ini, seperti melakukan *resampling* terhadap data yang ada. Resampling adalah teknik mengambil sampel secara berulang dari sampel data asli (Statistics Solution, 2016). Teknik resampling terdiri dari *oversampling*, yaitu mengambil sampel berulang kali dari kelas minoritas; dan undersampling, yaitu mengambil sampel secara acak dari kelas mayoritas (Burnaev, Erofeev, & Papanov, 2015). Kedua teknik ini dapat digunakan secara terpisah ataupun digabung (Burnaev, Erofeev, & Papanov, 2015; Anand, Pugalenthi, Fogel, & Suganthan, 2010; More, 2016; Yen & Lee, 2006). SMOTE adalah teknik oversampling yang terpopuler, dengan Borderline-SMOTE merupakan ekstensi dari SMOTE. Salah satu Teknik resampling yang cukup popular adalah ADASYN yang mampu menyesuaikan jumlah data sintetiknya.

Dalam beberapa penelitian terkait (More, 2016; Batista, Prati, & Monard, 2004; Amin, et al., 2016; Burnaev, Erofeev, & Papanov, 2015), telah dilakukan berbagai percobaan untuk mengatasi masalah dataset tak seimbang, namun metode-metode resampling maupun algoritma *machine learning* yang digunakan tidak beragam untuk mengetahui metode terbaik untuk mengatasi masalah ini. Seperti penelitian yang dilakukan Amin (Amin, et al., 2016) hanya meneliti teknik oversampling, Burnaev, More, dan Batista dkk (Burnaev, Erofeev, & Papanov, 2015; More, 2016; Batista, Prati, & Monard, 2004) meneliti teknik oversampling dan undersampling namun hanya menggunakan satu algoritma *machine learning*, sedangkan Diri (Diri & Albayrak, 2008) hanya meneliti beberapa algoritma *machine learning* tanpa pertimbangan dataset tak seimbang.

Sedangkan untuk mengetahui metode resampling dan algoritma *machine learning* terbaikuntuk masalah ini, dibutuhkan kombinasi-kombinasi antar teknik resampling, dan juga antar algoritma *machine learning*. Setiap kombinasi (pasangan) ini, seperti SMOTE dengan Support Vector Machine, atau Tomek Links dengan Regresi Logistik akan diuji performanya terhadap dataset yang diberikan, kemudian dari kombinasi-kombinasi tersebut dapat ditarik kesimpulan mengenai kombinasi algoritma dan teknik resampling yang terbaik, dan algoritma *machine learning* dengan performa terbaik, dan teknik resampling dengan performa terbaik. Setiap kombinasi atau pasangan dievaluasi hasilnya dengan tidak hanya pada satu dataset tak seimbang saja, melainkan dengan beberapa dataset tambahan untuk mendapatkan hasil yang lebih umum.

Berdasarkan uraian di atas, penulis ingin melakukan penelitian mengenai dataset tak seimbang dengan menggunakan "Kombinasi Teknik Resampling dan Algoritma *Machine Learning* pada Kelas tak Seimbang".

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang masalah di atas, dapat dirumuskan masalah: Kombinasi algoritma *Machine Learning* dan teknik Resampling yang mana saja yang berperforma dengan baik untuk mengatasi dataset tak seimbang?

1.3 Batasan Masalah

Batasan masalah pada penelitian ini adalah:

1. Merupakan masalah klasifikasi (*Supervised Learning*).
2. Dataset yang digunakan adalah dataset tak seimbang (*imbalanced ratio* < 3:4*)*.
3. Dataset hanya memiliki dua kelas (biner).
4. Atribut-atribut dataset merupakan tipe numerik.
5. Teknik resampling yang digunakan adalah Random Undersampling, Tomek Links, Random Oversampling, SMOTE, Borderline-SMOTE, dan ADASYN.
6. Algoritma *machine learning* yang digunakan adalah Regresi Logistik, Decision Tree, Neural Network, Nearest Neighbor, dan Support Vector Machine.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah, maka tujuan dari penelitian ini adalah:

1. Mengetahui algoritma *machine learning* yang berperforma dengan baik dalam mengklasifikasi data.
2. Mengetahui teknik resampling yang memiliki performa baik pada dataset tak seimbang.
3. Mengetahui kombinasi algoritma *machine learning* dan teknik resampling yang berperforma dengan baik dalam mengatasi dataset tak seimbang.

1.5 Manfaat Penelitian

Hasil penelitian ini diharapkan dapat bermanfaat:

1. Sebagai rujukan untuk mengatasi dataset tak seimbang yang sering dijumpai.
2. Menjadi sumber informasi mengenai performa beberapa teknik resampling.
3. Menjadi sumber informasi mengenai performa dari beberapa algoritma *machine learning*.

BAB II
TINJAUAN PUSTAKA

2.1 *Machine Learning*

Machine learning adalah memprogram komputer untuk mengoptimalkan suatu ukuran kinerja menggunakan sampel data atau berdasarkan pengalaman (Ethem, 2009). *Machine learning* menggunakan suatu algoritma untuk menganalisis data.

Dalam pembelajaran yang terawasi atau *supervised learning*, pengklasifikasi akan diberikan suatu input tertentu dan menghubungkannya dengan suatu output. Kasus di mana tujuannya adalah mengklasifkasikan input data ke suatu kategori diskrit tertentu disebut *klasifikasi*, dan kasus di mana outputnya adalah suatu variabel kontinu disebut *regresi*. Dalam pembelajaran tanpa pengawasan atau *unsupervised learning*, pengklasifikasi diberi input dan dibiarkan sendiri untuk menemukan pola pada data tersebut. Kasus *unsupervised learning* di mana tujuannya adalah mengelompokkan observasi-observasi yang mirip disebut *clustering*, jika menentukan distribusi data pada input disebut *estimasi kepadatan*. Dalam *reinforcement learning*, sistem komputer menerima input secara terus menerus dan mencoba memilih keputusan-keputusan yang paling optimal berdasarkan kondisi lingkungannya.

Masing-masing jenis pembelajaran memiliki banyak algoritma yang telah dikembangkan dengan pendekatan yang berbeda-beda (Jordan & Mitchell, 2015).

2.2 *Imbalanced Class*

Dataset adalah kumpulan data yang berbentuk tabel, di mana setiap kolomnya merepresentasikan suatu ciri-ciri, atribut atau fitur. Setiap barisnya menyatakan observasi suatu individu, record atau sampel (Snijders, Matzat, & Reips, 2012). Suatu dataset biasanya memiliki satu kolom tambahan yang merepresentasikan kelas dari observasi tersebut, kolom ini disebut kolom kelas. Kolom kelas ini juga disebut sebagai variabel dependen terhadap variabel-variabel independen yang merupakan ciri-ciri (atribut) dari suatu observasi tertentu.

Dalam *machine learning* dikenal istilah dataset dengan class yang tak seimbang. Istilah ini berlaku ketika kelas dari dataset tersebut bersifat kategorik diskrit. Dataset dengan class yang tak seimbang (*imbalanced class*) adalah dataset yang frekuensi kejadian dari kelas tertentu sangat jauh berbeda dengan kelas yang lain. Contohnya seperti suatu dataset dengan jumlah pasien yang berkelas "diabetes" jumlahnya jauh lebih sedikit dibanding pasien yang "tidak diabetes".

Masalah ketidakseimbangan ini akan memberi bias terhadap performa pengklasifikasi sebab jumlah sampel pada kelas tertentu tidak dapat memberi informasi yang cukup kepada pengklasifikasi berdasarkan ciri-ciri yang diberikan (Bishop, Pattern Recognition and Machine Learning, 2006; Ethem, 2009; Domingos, 2011).

2.3 *Confusion Matrix*

Di dalam *machine learning*, mengukur kinerja atau performa dari suatu model adalah hal yang esensial. Model yang diperoleh dari pelatihan melalui data training perlu diuji melalui data testing. Kinerja diukur berdasarkan seberapa baik model tersebut memprediksi dengan benar data yang ada.

Pada klasifikasi biner, kelas positif yang berhasil diprediksi dengan benar disebut *true positive*, jika kelas positif tersebut diprediksi negatif (salah) disebut *false negative*. Kelas negatif yang berhasil diprediksi negatif (benar) disebut *true negative*, dan kelas negatif yang diprediksi positif disebut *false positive*. Jumlah dari kasus-kasus tersebut direpresentasikan dalam suatu tabel kontingensi yang disebut *confusion matrix* (Swets, 1988).

Tabel 1: Confusion Matrix

	Kelas asli		
Hasil prediksi		Positif	Negatif
	Positif	TP	FP
	Negatif	FN	TN

Akurasi adalah ukuran kinerja yang menunjukkan seberapa baik suatu pengklasifikasi dalam mengklasifikasikan seluruh data. Akurasi adalah rasio antara observasi yang diklasifikasikan secara benar dengan total observasi:

$$Accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)} \quad \text{(II.1)}$$

Presisi adalah ukuran kinerja yang menunjukkan seberapa besar kebenaran suatu pengklasfikasi dari seluruh kelas positif yang diprediksi. Presisi adalah rasio antara jumlah kelas positif yang diklasifikasikan secara benar dengan jumlah observasi yang diklasifikasikan positif:

$$Precision = \frac{TP}{(TP + FP)} \quad \text{(II.2)}$$

Recall atau sensitivitas adalah ukuran kinerja yang menunjukkan seberapa baik suatu pengklasifikasi dalam mengklasifikasikan kelas positif. *Recall* adalah rasio antara jumlah observasi positif yang diklasifikasikan secara benar dengan jumlah observasi positif asli:

$$Recall = \frac{TP}{(TP + FN)} \quad \text{(II.3)}$$

F1-Score adalah *harmonic mean* antara *precision* dan *recall*:

$$F1 = 2 * \frac{Precision * Recall}{(Precision + Recall)} \quad \text{(II.4)}$$

2.4 Jarak Euklid

Jarak Euklid adalah jarak antara suatu vektor $X = (x_1, \dots, x_n)$ ke suatu vektor $Y = (y_1, \dots, y_n)$ pada ruang euklid berdimensi n:

$$d(X, Y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2} \quad \text{(II.5)}$$

Pada *machine learning*, jarak Euklid digunakan untuk menghitung jarak antar dua observasi berdasarkan vektor fitur yang bersifat kontinu (Howard, 2013).

2.5 *Principal Common Analysis* (PCA)

Principal Common Analysis (PCA) adalah metode *feature extraction* dari suatu dataset dengan p atribut numerik, untuk setiap atribut memiliki n observasi. Dengan kata lain, didefinisikan sejumlah p vektor X berdimensi n $(X_1, \dots, X_p)$ atau $n * p$ data matrix $\boldsymbol{X}$, di mana kolom ke-j adalah vektor X_j. Cari kombinasi linear dari kolom dari matrix $\boldsymbol{X}$ dengan variansi maksimum. Kombinasi linear tersebut diekspresikan dengan:

$$\sum_{j=1}^{p} A_j X_j = \boldsymbol{X}A$$

di mana A adalah vektor konstan $a_1, a_2, \dots, a_p$.

Variansi dari kombinasi linear tersebut didefinisikan sebagai:

$$var(\boldsymbol{X}A) = A^T \boldsymbol{S} A \tag{II.6}$$

di mana $\boldsymbol{S}$ adalah matrix kovarians dari $\boldsymbol{X}$ dan A^T adalah vektor transpose dari A. Maka mengidentifikasi kombinasi linear dengan variansi maksimum ekuivalen dengan memperoleh vektor A berdimensi p yang memaksimalkan ekspresi $A^T \boldsymbol{S} A$. Untuk memperoleh solusi yang terdefinisi, vektor A harus merupakan vektor satuan, atau $A^T A = 1$:

$$A^T S A - \lambda(A^T A - 1) \tag{II.7}$$

di mana λ adalah pengganda Lagrange. Menurunkan ekspresi (II.7) terhadap vektor A, dan menyamakan dengan vektor nol, menghasilkan persamaan:

$$\boldsymbol{S}A - \lambda A = \boldsymbol{0} \Leftrightarrow \boldsymbol{S}A = \lambda A \tag{II.8}$$

Dapat disimpulkan bahwa A adalah vektor eigen, dan λ adalah nilai eigen dari matrik kovarians $\boldsymbol{S}$. Karena nilai eigen adalah variansi dari transformasi linear pada persamaan (II.6):

$$var(\boldsymbol{X}A) = A^T \boldsymbol{S} A$$

$$= \lambda A^T A$$
$$= \lambda$$

maka persamaan (II.8) bernilai benar jika setiap vektor eigen dikali -1.

Seluruh matriks real simetrik berukuran $p * p$, seperti matriks kovarians $\boldsymbol{S}$, memiliki tepat p nilai eigen yang real, $\lambda_k (k = 1, \dots, p)$, dan vektor eigennya dapat didefinisikan untuk membentuk satu set vektor ortonormal. Pendekatan pengganda Lagrange, dengan batasan tambahan ortogonal dari vektor koefisien yang berbeda, juga dapat digunakan untuk menunjukkan bahwa seluruh vektor eigen dari **S** adalah solusi dari masalah dengan memperoleh p kombinasi linear baru:

$$\boldsymbol{X}A_k = \sum_{j=1}^{p} A_{jk} X_j \qquad \text{(II.9)}$$

yang memaksimalkan varians, tergantung pada korelasi dengan kombinasi linear sebelumnya. Pada kombinasi linear ini, $\boldsymbol{X}A_k$ adalah principal component atau komponen utama dari dataset.

(Jolliffe & Cadima, 2016)

2.6 LASSO dan Ridge *Regression*

Regularisasi adalah metode untuk menghindari *overfitting* dengan memberi nilai penalti terhadap koefisien regresi yang bernilai tinggi. Regularisasi mengurangi parameter dan menyederhanakan model yang berbentuk kompleks. Regularisasi menambah nilai penalti pada model yang lebih kompleks dan mengoptimisasi parameter dengan meminimalkan nilai suatu cost function.

LASSO Regression atau Least Absolute Shrinkage and Selection Operator Regression adalah metode regularisasi yang menambahkan penalti sebesar nilai absolut dari besarnya koefisien dikalikan dengan suatu nilai λ. Regularisasi ini dapat menghasilkan model dengan koefisien yang lebih sedikit dikarenakan beberapa koefisien dapat menjadi nol dan dihilangkan dari model. Nilai penalti λ yang lebih

besar menghasilkan nilai-nilai koefisien yang lebih mendekati nol, yang ideal untuk menghasilkan model yang lebih sederhana.

$$\sum_{i=1}^{n}\left(y_i - \sum_{j} x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \qquad \text{(II.10)}$$

Ridge Regression adalah metode regularisasi yang menambahkan penalti sebesar nilai kuadrat dari besarnya koefisien dikalikan dengan suatu nilai λ. Regularisasi ini menambahkan bias terhadap regresi dengan hasil nilai variansi yang lebih kecil.

$$\sum_{i=1}^{n}\left(y_i - \sum_{j} x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^{p} {\beta_j}^2 \qquad \text{(II.11)}$$

Dilihat dari persamaan (II.10) dan persamaan (II.11), jika λ bernilai nol, maka cost function tersebut kembali ke Ordinary Least Squares, jika nilai λ terlalu besar, maka bobot yang diberikan akan terlalu besar dan model dapat mengalami *underfitting*.

Perbedaan dari LASSO dan Ridge Regression adalah pada LASSO Regression, dengan semakin besarnya nilai λ, koefisien-koefisien parameter dapat dirubah menjadi nol, mengeliminasi fitur yang tidak berperan terhadap model; sedangkan pada Ridge Regression, koefisien parameter tidak akan mencapai nol, namun secara asimtotik mendekati nol. (Tibshirani, 1996; Pereira, Basto, & Silva, 2016)

2.7 Teknik *Resampling*

2.7.1 *Random Oversampling*

Random oversampling, atau oversampling secara acak adalah teknik oversampling di mana anggota dari kelas minoritas dipilih secara acak dan diduplikasi ke dataset yang baru hingga tercapai keseimbangan (Liu, 2004). Data minoritas tersebut dapat diduplikasi beberapa kali. Teknik ini biasanya menyebabkan *overfitting* pada model (Amin, et al., 2016; Liu, 2004).

2.7.2 *Synthetic Minority Oversampling Technique* (SMOTE)

SMOTE atau *Synthetic Minority Oversampling Technique* adalah teknik oversampling terpopuler yang diproposalkan oleh Chawla (Chawla, Bowyer, Hall, & Kegelmeyer, 2002) pada tahun 2002. Teknik ini membuat data tiruan atau sintetik berdasarkan tetangga-tetangga terdekat dari sampel kelas minoritas.

Teknik ini dimulai dengan menentukan n data yang akan dibuat untuk setiap data pada kelas minoritas dalam dataset $\boldsymbol{D}$. Kemudian untuk setiap data kelas minoritas $M_{i,c}$, pilih n tetangga secara acak dari k tetangga terdekat data tersebut, di mana $i = \{1,2, \dots m\}$ dengan m adalah jumlah data pada kelas minoritas dan $c \in C$, di mana $C = \{c_1, c_2, \dots, c_z\}$ adalah fitur-fitur pada $\boldsymbol{D}$. Lalu untuk setiap fitur c pada $M_{i,c}$, hitung jarak euklid d antara $M_{i,c}$ dengan salah satu tetangga $T_{s,c}$, di mana s adalah bilangan acak $s = \{1,2,..,k\}$ dari k tetangga terdekat $M_{i,c}$. Kemudian suatu bilangan acak $g = [0,1]$ ditentukan. Data tiruan dibuat berdasarkan:

$$S_{j,c} = M_{i,c} + (g * d) \tag{II.12}$$

di mana $j = \{1,2,..,n * m\}$ bersifat inkremental dan S adalah data kelas minoritas tiruan.

Teknik ini membuat $n * m$ data sintetik pada suatu titik dari jarak antara setiap fitur dari M dengan T (Chawla, Bowyer, Hall, & Kegelmeyer, 2002).

2.7.3 *Borderline – Synthetic Minority Oversampling Technique* (*Borderline*-SMOTE)

Terinspirasi oleh SMOTE, Han memproposalkan teknik oversampling baru yang menyerupai SMOTE. Teknik Han tersebut membuat data tiruan di sekitar data kelas minoritas yang berada di *borderline* atau perbatasan antara kelas mayoritas dan kelas minoritas saja, di mana SMOTE membuat N data tiruan pada setiap sampel kelas minoritas (Han, Wang, & Mao, 2005).

Misalkan untuk data training ***D*** terdapat kelas minoritas P dan kelas mayoritas N, diekspresikan:

$$P = \{p_1, p_2, \dots, p_{pnum}\}, N = \{n_1, n_2, \dots, n_{nnum}\}$$

di mana $pnum$ dan $nnum$ adalah jumlah sampel kelas minoritas dan jumlah sampel kelas mayoritas. Borderline-SMOTE bekerja sebagai berikut.

Untuk setiap $p_i (i = 1,2, \dots, pnum)$ pada kelas minoritas P, hitung m tetangga terdekat dari seluruh data training D. Jumlah kelas mayoritas dari m tetangga terdekat p_i disimbolkan $m' (0 \leq m' \leq m)$.

Jika $m' = m$, atau seluruh tetangga terdekat dari p_i adalah sampel kelas mayoritas, maka p_i dinyatakan sebagai noise dan bukan merupakan sampel perbatasan. Jika $m/2 \leq m' < m$, atau tetangga terdekat p_i lebih banyak merupakan sampel kelas mayoritas dibanding minoritas, maka p_i dianggap sampel perbatasan dan dimasukkan ke dalam himpunan sampel perbatasan B. Jika $0 \leq m' < m/2$, maka p_i dianggap aman dan bukan merupakan sampel perbatasan.

Untuk himpunan sampel perbatasan B, di mana $B \subseteq P$. Maka:

$$B = \{p'_1, p'_2, \dots, p'_{dnum}\}, 0 \leq dnum \leq pnum$$

di mana $dnum$ adalah jumlah sampel perbatasan. Untuk setiap sampel perbatasan di B, tentukan k tetangga terdekat dari P.

Data tiruan berkelas minoritas dibuat sebanyak $s * dnum$ $(s = 1, 2, .., k)$. Untuk setiap p'_i, Pilih sebanyak s tetangga terdekat secara acak dari P, lalu hitung jarak euklid (Persamaan (II.5)) $d_j (j = 1,2, \dots, s)$ antara p'_i dengan p_j di mana d_j adalah jarak antara p'_i dengan salah satu tetangga terdekatnya. Kemudian suatu bilangan acak g $(g = [0,1])$ ditentukan dan data tiruan sebanyak s dari p'_i dibuat.

$$S_j = p'_i + (g * d_j) \qquad \text{(II.13)}$$

Proses di atas dilakukan untuk setiap p'_i, maka data tiruan akan dibuat sebanyak $s * dnum$ kali (Han, Wang, & Mao, 2005).

2.7.4 *Adaptive Synthetic* (ADASYN)

Adaptive Synthetic adalah teknik oversampling yang diproposalkan oleh He di mana data tiruan dibuat berdasarkan tingkat kesulitan suatu sampel kelas minoritas untuk dipelajari. Setiap sampel kelas minoritas memiliki alokasi data tiruan sesuai dengan banyaknya tetangga sampel kelas mayoritas dari k tetangga terdekat sampel kelas minoritas tersebut (He, Bai, Garcia, & Li, 2008).

Misalkan untuk data training D dengan m sampel $\{x_i, y_i\}$, $i = 1, \dots, m$, di mana x_i adalah instansi dari n dimensi dari ruang fitur X dan $y_i \in \{1, -1\}$ adalah label kelas dari x_i. Didefinisikan m_s dan m_l sebagai jumlah sampel kelas minoritas, dan jumlah sampel kelas mayoritas secara berurutan. Maka $m_s \leq m_l$ dan $m_s + m_l = m$.

Tentukan derajat ketidakseimbangan d:

$$d = \frac{m_s}{m_l} \tag{II.14}$$

di mana $d \in (0, 1]$. Jika $d < d_{th}$ di mana d_{th} adalah nilai maksimum toleransi derajat ketidakseimbangan, maka hitung jumlah data kelas minoritas tiruan:

$$G = (m_l - m_s) * \beta \tag{II.15}$$

di mana $\beta \in [0, 1]$ adalah parameter yang menunjukkan rasio ketidakseimbangan yang diinginkan setelah data tiruan dibuat. $\beta = 1$ menunjukkan dataset akan seimbang sepenuhnya.

Untuk setiap sampel $x_i \in P$ di mana P adalah kelas minoritas, tentukan K tetangga terdekat dengan jarak Euklid pada n dimensi ruang fitur. Kemudian hitung rasio r_i, yang didefinisikan sebagai:'

$$r_i = \frac{\Delta_i}{K}, i = 1, \dots, m_s \tag{II.16}$$

di mana Δ_i adalah jumlah sampel pada K tetangga terdekat x_i yang merupakan sampel kelas mayoritas, maka $r_i \in [0, 1]$. Kemudian normalisasikan r_i menjadi:

$$\hat{r}_i = \frac{r_i}{\sum_{i=1}^{m_s} r_i} \tag{II.17}$$

agar $\hat{r}_i$ menjadi distribusi kepadatan ($\sum_i \hat{r}_i = 1$).

Hitung data tiruan yang akan dibuat untuk setiap sampel x_i:

$$g_i = \hat{r}_i * G \tag{II.18}$$

Untuk setiap sampel $x_i \in P$, buat g_i data tiruan dengan persamaan:

$$s_i = x_i + (x_{zi} - x_i) * \lambda \tag{II.19}$$

di mana $(x_{zi} - x_i)$ adalah selisih vektor pada n dimensi dan λ adalah bilangan acak $\lambda \in [0, 1]$.

2.7.5 *Random Undersampling*

Random undersampling, atau *undersampling* secara acak adalah teknik undersampling di mana anggota dari kelas mayoritas dipilih secara acak dan dihapus dari dataset training hingga tercapai keseimbangan. Kekurangan dari teknik ini adalah tidak ada cara untuk mengatur informasi apa saja yang dihilangkan dari dataset tersebut, informasi yang berguna bisa saja hilang (Amin, et al., 2016; Liu, 2004; Yen & Lee, 2006; More, 2016).

2.7.6 Tomek *Links*

Tomek *Link* atau tautan Tomek adalah metode undersampling yang diproposalkan oleh Tomek (Tomek, 1976) untuk memodifikasi CNN (Condensed Nearest Neighbor). Teknik ini menghapus sampel kelas mayoritas jika dan hanya jika tetangga kelas mayoritas tersebut berasal dari kelas berbeda dan merupakan tetangga terdekat satu sama lain. Jika x dan y adalah sampel dengan kelas berbeda, maka suatu Tomek Link didefinisikan untuk setiap sampel z:

$$(d(x, y) < d(x, z)) \wedge (d(x, y) < d(y, z))$$

di mana fungsi $d(.)$ adalah jarak Euklid pada persamaan (II.5).

2.8 Algoritma Klasifikasi *Machine Learning*

2.8.1 *Logistic Regression*

Regresi logistik adalah algoritma klasifikasi yang menggunakan kelas untuk membangun dan menggunakan model regresi logistik multinomial tunggal dengan *estimator* tunggal. Regresi logistik menyatakan probabilitas kelas (antara 0 dan 1) tergantung pada jarak dari batas, dengan sutu pendekatan. Hasil regresi logistik adalah suatu bilangan antara 0 dan 1 yang menyatakan probabilitas kelas.

Nilai atribut pada data observasi dari regresi logistik dapat berupa nominal, ordinal, interval atau skala rasio, sedangkan untuk atribut kelas harus merupakan kelas biner. Hubungan antara atribut dengan kelas bersifat non-linear. Distribusi data tidak tersebar dalam bentuk distribusi Gauss, melainkan distribusi Bernoulli, yang dikarenakan kelasnya berbentuk biner.

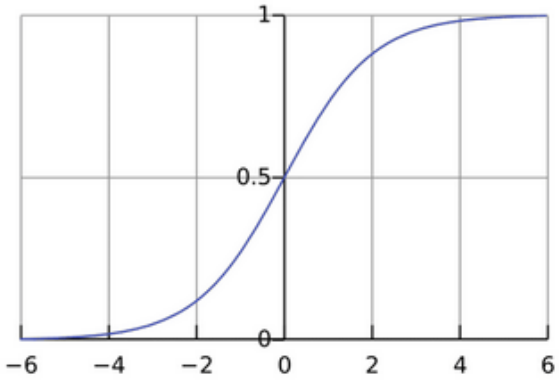
Hipotesis H_θ dari logistik regresi didefinisikan sebagai:

$$H_\theta(x) = g(\theta^T x) \tag{II.20}$$

di mana:

$$g(z) = \frac{1}{1 + e^{-z}}$$

adalah **fungsi logistik** atau **fungsi sigmoid**.



Gambar 1: Fungsi Sigmoid

Terlihat bahwa $g(z)$ cenderung ke 1 jika $z \to \infty$ dan $g(z)$ cenderung ke 0 jika $z \to -\infty$. Lebih jauh, $g(z)$ dan $h(z)$ selalu bernilai dalam interval 0 dan 1. Jika dimisalkan $x_0 = 1$, maka:

$$\theta^T x = \theta_0 + \sum_{j=1}^{n} \theta_j x_j$$

dengan θ_0 adalah nilai **intercept**.

Turunan dari fungsi sigmoid, atau g' adalah:

$$\begin{aligned} g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\ &= \frac{1}{(1+e^{-z})^2}(e^{-z}) \\ &= \frac{1}{(1+e^{-z})}\left(1 - \frac{1}{(1+e^{-z})}\right) \\ &= g(z)(1-g(z)) \end{aligned} \qquad \text{(II.21)}$$

Untuk melakukan fitting θ ke dalam regresi logistik, asumsikan beberapa persamaan probabilistik dan tentukan nilai θ melalui maximum likelihood.

Asumsikan:

$$P(y = 1|\, x; \theta) = \ h_\theta(x)$$
$$P(y = 0|\, x; \theta) = 1 - h_\theta(x)$$

dapat ditulis:

$$p(y \mid x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}$$

Dengan mengasumsikan m sampel data training yang saling independen, likelihood dari parameter-parameter tersebut dapat ditulis:

$$\begin{aligned} L(\theta) &= p(\vec{y} \mid X; \theta) \\ &= \prod_{i=1}^{m} p(y_i \mid x_i; \theta) \\ &= \prod_{i=1}^{m} (h_\theta(x))^y (1 - h_\theta(x))^{1-y} \end{aligned}$$

Log likelihood-nya adalah:

$$\begin{aligned} \ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{m} y_i \log h(x_i) + (1 - y_i) \log(1 - h(x_i)) \end{aligned}$$

Log likelihood di atas dapat dimaksimalkan menggunakan **gradient ascent**. Sebelum itu, **gradient descent**, adalah algoritma yang meminimalkan nilai suatu fungsi, berkebalikan dengan gradient ascent yang memaksimalkan nilai suatu fungsi. Pada gradient descent, θ dipilih sedemikian hingga nilai suatu fungsi $J(\theta)$ minimal. Algoritma ini memilih θ secara acak kemudian secara berulang (iterasi) mengubah θ untuk meminimalkan nilai $J(\theta)$ hingga mencapai nilai konvergen untuk θ yang meminimalkan $J(\theta)$. Setiap iterasi dari algoritma ini memilih nilai θ_j yang baru, disebut *update*, yang ditulis:

$$\theta_j \coloneqq \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \quad \text{(II.22)}$$

Update pada (II.22) dilakukan secara bersamaan untuk setiap nilai $j = 0, \dots, n$. Parameter α disebut **learning rate**, yang mengendalikan seberapa besar update dari θ_j.

Sebagai kontras, pada *gradient ascent*, *update* dari persamaan (II.22) ditulis sebagai:

$$\theta_j \coloneqq \theta_j + \alpha \frac{\partial}{\partial \theta_j} J(\theta) \quad \text{(II.23)}$$

Menggunakan persamaan (II.21) Turunan, atau **gradient** dari $\ell(\theta)$ adalah:

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} \ell(\theta) &= \left(y \frac{1}{g(\theta^T x)} - (1-y) \frac{1}{1 - g(\theta^T x)} \right) \frac{\partial}{\partial \theta_j} g(\theta^T x) \\
&= \left(y \frac{1}{g(\theta^T x)} - (1-y) \frac{1}{1 - g(\theta^T x)} \right) g(\theta^T x)\left(1 - g(\theta^T x)\right) \frac{\partial}{\partial \theta_j} \theta^T x \\
&= \left(y\left(1 - g(\theta^T x)\right) - (1-y) g(\theta^T x) \right) x_j \\
&= (y - h_\theta(x)) x_j
\end{aligned}$$

Substitusi ke persamaan (II.23):

$$\theta_j \coloneqq \theta_j + \alpha (y_i - h_\theta(x_i)) x_{j_i}$$

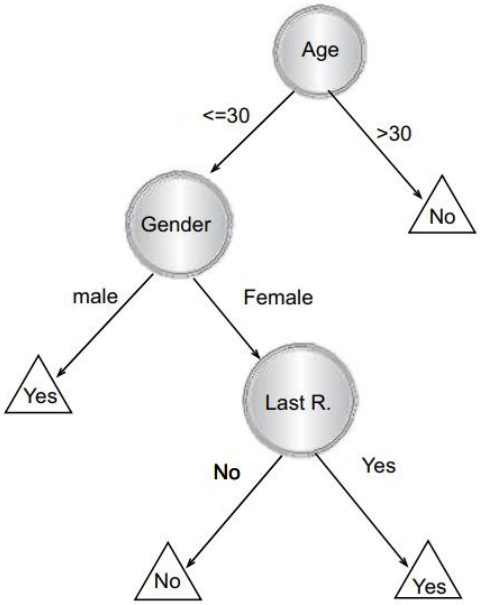
Diberikan input x testing, untuk membuat prediksi kelas, suatu nilai **threshold** r ditentukan antara 0 dan 1 sedemikian hingga $y = 1$ jika $H_\theta(x) \geq r$ dan $y = 0$ jika . $H_\theta(x) < r$.

(Tsangaratos & Ilia, 2016; Hosmer & Lemeshow, 2000; Osisanwo, et al., 2017).

2.8.2 *Decision Tree*

Decision Tree atau pohon keputusan adalah pengklasifikasi yang menentukan output kelas dari suatu sampel berdasarkan keputusan yang diambil dari setiap nilai atribut dari sampel tersebut. Pohon keputusan adalah suatu fungsi Boolean di mana inputnya adalah suatu sampel *E* dengan vektor fitur *X* dan outputnya adalah 0 atau 1. Pada pohon keputusan, setiap *node* pohon yang bukan *node* daun adalah suatu uji

atribut atau suatu ekspresi boolean, setiap *node* daun adalah nilai Boolean, dan setiap cabang mewakili salah satu nilai yang mungkin dari atribut yang diuji.



Gambar 2: Decision Tree

Gambar 2 menunjukkan apakah seorang customer merespon surat langsung dari perusahaan atau tidak. Pada gambar di atas, *node internal* atau atribut disimbolkan sebagai lingkaran dan *node* daun digambarkan sebagai segitiga.

Jika suatu atribut bersifat kontinu, decision tree mengurutkan nilai kontinu tersebut dan memilih splitting point di antara nilai tersebut berdesarkan kelasnya.

Untuk membuat decision tree, banyak algoritma yang dapat digunakan, seperti ID3, C4.5, dan CART.

Algoritma ID3 menggunakan teori entropi informasi yang menghitung entropi dari masing-masing atribut. Kemudian menghitung Information Gain dari suatu atribut berdasarkan entropinya terhadap target kelas. Di mana **entropi** adalah ukuran ketidapastian suatu atribut yang terkait dengan suatu kejadian. Semakin kecil entropi dari suatu atribut, semakin tinggi kemurnian informasi yang ada. Semakin besar entropi

dari suatu atribut, semakin besar ketidakpastian informasi tersebut. **Information Gain** adalah jumlah ketidakpastian informasi yang berkurang berdasarkan informasi yang diterima terkait dengan suatu atribut.

Misalkan S adalah data Training dengan jumlah sampel s, dengan m jumlah nilai atribut kelas yang berbeda dari $C_i(i = 1,2,\dots,m)$. Misalkan S_i adalah jumlah sampel pada C_i, Information Gain minimal dari suatu sampel klasifikasi adalah:

$$I(S_1, \dots, S_m) = -\sum_{i=1}^{m} p_i \, log_2(p_i) \qquad \text{(II.24)}$$

di mana p_i adalah peluang suatu sampel berasal dari kelas C_i yang diperoleh dari s_i/s. Suatu himpunan atribut A memiliki n nilai berbeda $\{a_1, \dots, a_n\}$. S dibagi menjadi n subset $\{S_1, \dots, S_n\}$, di mana S_j memiliki beberapa sampel di S yang memiliki nilai A_j di A.

Misalkan s_{ij} adalah jumlah sampel di kelas C_i dari subset S_j, entropi subset yang dibagi oleh A adalah:

$$E(A) = -\sum_{j=1}^{n} \frac{s_{1j} + s_{2j} + \cdots + s_{mj}}{s} * I(s_{1j}, s_{2j}, \dots, s_{mj}) \qquad \text{(II.25)}$$

Berdasarkan persamaan (II.25) di atas, informasi yang dibutuhkan dari suatu subset S_j dihitung berdasarkan:

$$I\left(s_{1j}, s_{2j}, \dots, s_{mj}\right) = -\sum_{i=1}^{m} p_{ij} \, log_2 p_{ij} \qquad \text{(II.26)}$$

Maka Information Gain dari A dapat dihitung dengan:

$$Gain(A) = I(s_1, s_2, \dots, s_m) - E(A) \qquad \text{(II.27)}$$

Gini index adalah criterion berbasis impurity yang mengukur divergensi antara distribusi peluang dari nilai atribut target, didefinisikan:

$$Gini(y,S) = \sum_{i=1}^{m} {p_i}^2$$

Algoritma ID3 membangun decision tree berdasarkan urutan atribut yang memiliki information gain tertinggi sebagai *root node*. *Node* tersebut kemudian bercabang sesuai dengan jumlah nilai berbeda pada atribut tersebut. Cabang dengan entropi nol adalah *node* daun, dan cabang dengan entropi yang lebih dari nol membutuhkan pecabangan lebih lanjut. Proses tersebut dilakukan secara rekursif pada semua *node* yang bukan merupakan *node* daun hingga seluruh data terklasifikasi

(Osisanwo, et al., 2017; Quinlan, 1986; Dai, Zhang, & Wu, 2016; Vafeiadis, Diamantaras, Sarigiannidis, & Chatzisavvas, 55).

2.8.3 *Support Vector Machine*

Support Vector Machine adalah algoritma pengklasifikasi yang membangun suatu *hyperplane* yang memisahkan data-data dengan kelas berbeda dengan margin sebesar mungkin.

Misalkan suatu data training pada sampel $(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$ dipisahkan menjadi dua kelas, di mana $x_i \in R^n$ adalah vektor fitur dan $y_i \in \{-1, 1\}$ adalah label kelasnya. Jika diasumsikan dua kelas tersebut dapat dipisahkan dengan *hyperplane* $w * x + b = 0$ pada suatu bidang H, dan informasi mengenai distribusi datanya tidak diketahui, maka *hyperplane* yang optimal adalah *hyperplane* yang memaksimalkan margin. Model SVM didefinisikan sebagai:

$$H_{w,b}(x) = g(w^T x + b)$$

di mana $g(z) = 1$ jika $z \geq 0$, dan $g(z) = -1$ jika $z < 0$.

Functional margin dari *hyperplane* (w, b) untuk x_i didefinisikan sebagai:

$$\hat{\gamma}_i = y_i(w^T x_i + b) \qquad \text{(II.28)}$$

dan *functional margin* dari seluruh dataset didefinisikan sebagai:

$$\hat{\gamma} = \min_{i=1,..,m} \hat{\gamma}_i$$

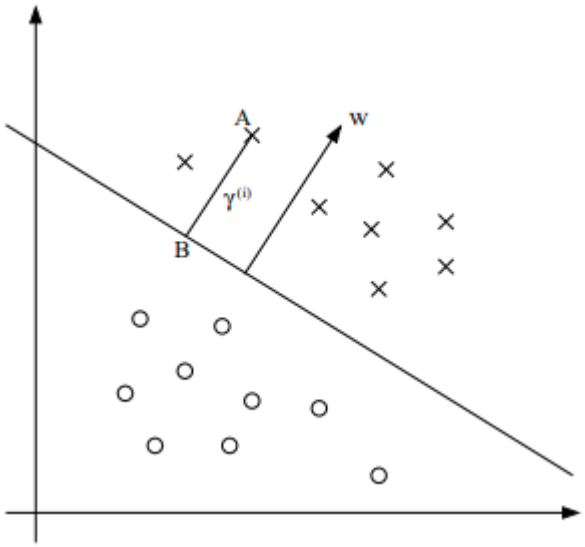
Geometric margin dari *hyperplane* (w, b) untuk x_i adalah *functional margin* yang dinormalisasikan dengan $\|w\|$:

$$\gamma_i = \frac{y_i(w^T x_i + b)}{\|w\|} \tag{II.29}$$

dan geometric margin dari seluruh dataset didefinisikan sebagai:

$$\gamma = \min_{i=1,..,m} \gamma_i$$

Dari persamaan (II.28), jika $y_i = 1$, maka untuk membuat *functional margin* yang besar, $w^T x_i + b$ harus menghasilkan nilai yang besar. Sebaliknya, jika $y_i = -1$, maka untuk membuat functional margin yang besar, $w^T x_i + b$ harus menghasilkan nilai negatif yang besar.



Gambar 3: Geometric margin pada SVM

Pada Gambar 3, *decision boundary* atau *hyperplane* digambarkan sebagai garis diagonal pemisah antara kelas X ($y_i = 1$) dan O ($y_i = -1$). Vektor w adalah vektor yang ortogonal terhadap *hyperplane*. Titik pada A, merupakan suatu vektor x_i dari data

training dengan label $y_i = 1$. Jarak dari titik A dari *hyperplane*, atau γ_i, digambarkan pada garis AB. Untuk mencari nilai γ_i, $\frac{w}{\|w\|}$ adalah vektor satuan yang mempunyai arah yang sama dengan w. Karena A merepresentasikan x_i, maka titik B adalah $x_i - \gamma_i * \frac{w}{\|w\|}$ yang terletak tepat pada *hyperplane*. Seluruh titik yang terletak pada *hyperplane* memenuhi persamaan $w^T x + b = 0$. Maka:

$$w^T \left(x_i - \gamma_i \frac{w}{\|w\|} \right) + b = 0$$

atau dapat ditulis:

$$\gamma_i = \frac{w^T x_i + b}{\|w\|}$$

$$\gamma_i = \left(\frac{w}{\|w\|} \right)^T x_i + \frac{b}{\|w\|} \qquad \text{(II.30)}$$

Namun persamaan (II.30) diperoleh dari titik A yang memiliki $y_i = 1$ kelas positif. Lebih umumnya, *geometric margin* dari (w, b) terhadap sampel training (x_i, y_i) didefinisikan:

$$\gamma_i = y_i \left(\left(\frac{w}{\|w\|} \right)^T x_i + \frac{b}{\|w\|} \right) \qquad \text{(II.31)}$$

yang ekuivalen dengan persamaan (II.29).

Jika diasumsikan training set yang diberikan dapat dipisah secara linear (*linearly separable*), *functional margin* tertinggi dapat direpresentasikan sebagai masalah optimisasi berikut:

$$\max_{\gamma, w, b} \gamma$$

dengan syarat:

$$y_i(w^T x_i + b) \geq \gamma, \qquad i = 1, \dots, m$$
$$\|w\| = 1$$

yang berarti γ dimaksimalkan dengan syarat functional margin bernilai lebih besar atau sama dengan γ dan $\|w\|$ bernilai 1. $\|w\| = 1$ memastikan bahwa functional margin akan bernilai sama dengan *geometric margin* yang menjamin seluruh *geometric margin* bernilai lebih besar atau sama dengan γ. Namun karena $\|w\| = 1$ juga masalah optimisasi di atas bersifat *non-convex*, maka masalah optimisasi tersebut dapat transformasikan menjadi:

$$\max_{\hat{\gamma},w,b} \frac{\hat{\gamma}}{\|w\|}$$

dengan syarat:

$$y_i(w^T x_i + b) \geq \hat{\gamma}, \qquad i = 1, \dots, m$$

yang berarti $\frac{\hat{\gamma}}{\|w\|}$ dimaksimalkan dengan syarat seluruh *functional margin* lebih besar atau sama dengan $\hat{\gamma}$. Bentuk ini menghapus syarat $\|w\| = 1$, namun masalah optimisasi ini masih bersifat *non-convex* yang disebabkan oleh fungsi objektif $\frac{\hat{\gamma}}{\|w\|}$. Namun dengan memberi syarat bahwa *functional margin* dari (w, b) harus bernilai 1:

$$\hat{\gamma} = 1$$

Maka fungsi objektif $\frac{\hat{\gamma}}{\|w\|}$ akan berubah menjadi $\frac{1}{\|w\|}$ yang ekuivalen dengan meminimalkan $\|w\|^2$, masalah optimisasi yang baru dapat ditulis:

$$\min_{w,b} \frac{1}{2} \|w\|^2 \qquad \text{(II.32)}$$

dengan syarat:

$$y_i(w^T x_i + b) \geq 1, \qquad i = 1, \dots, m$$

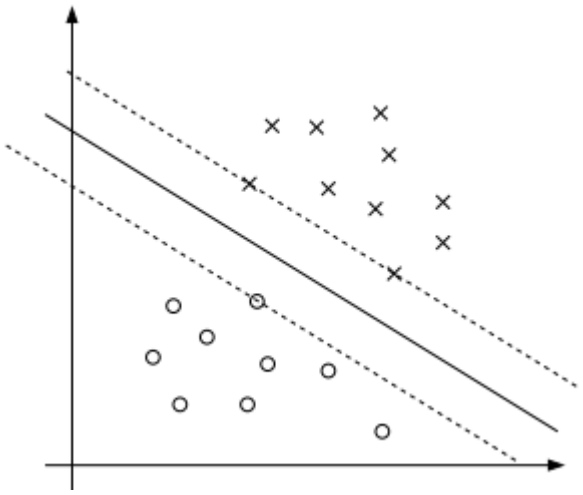
yang merupakan objective function kuadratik yang bersifat *convex* dengan syarat linear. Masalah optimisasi (II.32) dapat diselesaikan menggunakan *Quadratic Programming*. Dari masalah optimisasi (II.32), syarat optimisasinya dapat ditulis sebagai:

$$g_i(w) = -y_i(w^T x_i + b) + 1 \leq 0 \qquad \text{(II.33)}$$

Didefinisikan kondisi dual komplementer Karush-Kuhn-Tucker (KKT):

$$\alpha_i^* g_i(w^*) = 0, \qquad i = 1, \dots, k \qquad \text{(II.34)}$$

Dari persamaan (II.34), $\alpha_i > 0$ hanya untuk sampel training yang memiliki nilai functional margin 1, atau $g_i(w) = 0$



Gambar 4: Margin dari Support Vector Machines

Pada Gambar 4, titik-titik dengan *margin* terkecil adalah titik-titik yang terdekat dengan *decision boundary*. Terdapat tiga titik yang berada tepat pada garis putus-putus yang paralel terhadap *decision boundary*. Jadi, hanya terdepat tiga α_i yang tidak bernilai nol untuk solusi optimal pada masalah optimisasi di atas.

Masalah optimisasi di atas akan menggunakan inner product $\langle x_i, x_j \rangle$ atau ${x_i}^T x_j$ antara titik-titik pada ruang input. Inner product ini berguna pada saat menggunakan kernel trick. Mentransformasikan ke bentuk Lagrange, masalah optimisasi (II.32) akan berbentuk:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i [y_i(w^T x_i + b) - 1] \qquad \text{(II.35)}$$

Untuk menemukan dual dari masalah di atas, $\mathcal{L}(w, b, \alpha)$ perlu diminimalkan dengan w dan b (dengan α tetap), dan mengubah turunan dari $\mathcal{L}$ ke nol:

$$\nabla_w \mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^{m} \alpha_i y_i x_i = 0$$

Yang berarti:

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i \tag{II.36}$$

Dengan menurunkan terhadap b, diperoleh:

$$\frac{\partial}{\partial b} \mathcal{L}(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{II.37}$$

Kemudian w dari persamaan (II.36) disubstitusi ke persamaan (II.35), diperoleh:

$$\mathcal{L}(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j {x_i}^T x_j - b \sum_{i=1}^{m} \alpha_i y_i$$

Namun dari (II.37), suku terakhir bernilai nol:

$$\mathcal{L}(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j {x_i}^T x_j$$

Persamaan di atas diperoleh dari meminimalkan $\mathcal{L}(w, b, \alpha)$ dengan w dan b. Menggabungkan dengan syarat koefisien Lagrange $\alpha_i \geq 0$ dan (II.37), diperoleh masalah optimisasi dual:

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle$$

dengan syarat:

$$\alpha_i \geq 0, \qquad i = 1, \dots, m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

Masalah optimisasi di atas dapat diselesaikan dengan menggunakan persamaan (II.37), memperoleh w yang optimal sebagai fungsi dari α. Setelah memperoleh w^*, mencari nilai optimal untuk b diperoleh dari:

$$b^* = -\frac{\max_{i:y_i=-1} w^{*T} x_i + \min_{i:y_i=1} w^{*T} x_i}{2}$$

Pada persamaan (II.37), yang menghasilkan nilai w optimal dari α, misalkan model yang diperoleh digunakan untuk memprediksi titik x baru. $w^T x + b$ dihitung dan diprediksi $y = 1$ jika dan hanya jika $w^T x + b > 0$. Namun dengan menggunakan persamaan (II.37), $w^T x + b$ dapat ditulis sebagai:

$$w^T x + b = \left(\sum_{i=1}^{m} \alpha_i y_i x_i \right)^T x + b \qquad \text{(II.38)}$$

$$w^T x + b = \sum_{i=1}^{m} \alpha_i y_i \langle x_i, x \rangle + b \qquad \text{(II.39)}$$

Maka dari itu, ketika seluruh α_i telah diperoleh, untuk membuat prediksi hanya perlu menghitung inner product antara x dengan seluruh sampel training. Namun, dari apa yang dilihat sebelumnya, terdapat hanya beberapa α_i yang tidak bernilai nol. Jadi, banyak iterasi dari (II.39) yang bernilai nol. Dan pada akhirnya model hanya perlu menghitung inner product antara x dengan support vector saja (bukan dari seluruh sampel training).

Untuk membuat *hyperplane*, terkadang data training yang diberikan sulit untuk dipisahkan. Data training yang diberikan perlu untuk ditransformasikan ke suatu bentuk lain dalam ruang yang berbeda. Fungsi yang memetakan vektor input ke suatu vektor lain disebut **Kernel**. Sebagai contoh, diberikan fungsi

$$\phi(x) = \begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix}$$

Pada beberapa kasus, di mana input x tidak dapat dipisahkan secara linear, input tersebut perlu ditransformasikan menggunakan fungsi $\phi(x)$. Pada persamaan (II.39) seluruh x diganti dengan $\phi(x)$. Karena algoritma SVM ini hanya perlu menghitung *inner product* $\langle x, z \rangle$, maka inner product tersebut berubah menjadi $\langle \phi(x), \phi(z) \rangle$. Lebih spesifik, diberikan fungsi pemetaan ϕ, suatu **Kernel** didefinisikan sebagai:

$$K(x, z) = \phi(x)^T \phi(z) \tag{II.40}$$

Maka dari itu, seluruh *inner product* $\langle x, z \rangle$ pada persamaan (II.39) dapat disubstitusi menjadi persamaan (II.40), dan dengan mengganti $w^T x + b$ menjadi fungsi f yang menerima input x, persamaan (II.39) dapat ditulis menjadi:

$$f(x) = \sum_{i=1}^{m} \alpha_i y_i K(x_i, x) + b \tag{II.41}$$

di mana α_i dan b diperoleh dari hasil optimisasi di atas, dan α_i tidak nol adalah *support vector*.

Untuk $K(x_i, x)$ adalah *inner product* atau hasil kali dalam antara dua observasi $i \in \{1, \dots, n\}$ dan $i' \in \{1, \dots, n\}$ yang diekspresikan:

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j} \tag{II.42}$$

disebut kernel Linear.

Metode SVM yang menggunakan kernel linear dapat memisahkan kelas-kelas secara linear. Namun secara praktis dataset yang dijumpai sering tidak dapat dipisahkan secara linear. Kernel non-linear lebih fleksibel dikarenakan kernel ini memetakan variabel p dan x ke dimensi yang lebih tinggi. Salah satu kernel non-linear yang populer digunakan adalah kernel radial atau kernel Gauss:

$$K(x_i, x_{i'}) = exp\left\{-\gamma \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2\right\} \quad \text{(II.43)}$$

di mana $\gamma > 0$ adalah parameter tuning tambahan. Ketika suatu sampel test terletak jauh dari sampel training, eksponensial tersebut akan menjadi negatif, dan $K(x_i, x_{i'})$ mendekati nol. Pada kasus ini, sampel training yang terletak jauh dari sampel test hampir tidak memiliki efek atau kontribusi kepada keputusan yang digunakan untuk mengklasifikasikan sampel test. Parameter γ mempengaruhi seberapa jauh letak suatu observasi agar dapat berkontribusi terhadap keputusan klasifikasi. Kernel ini lebih fleksibel dari kernel linear (Guenther & Schonlau, 2016; Schuldt, Laptev, & Caputo, 2004; Vapnik & Cortes, 1995; Andrew, 2000).

2.8.4 *Multilayer Perceptron* (MLP)

Artificial Neural Network (ANN) atau Jaringan Saraf Tiruan adalah salah satu algoritma *machine learning* yang mencari suatu fungsi f yang tidak diketahui berdasarkan input vektor X ke suatu *output* y:

$$y = f(X)$$

Pada tahap pelatihan, fungsi f dioptimasikan sedemikian sehingga hasil yang didapat untuk setiap vektor X yang diberikan sedekat mungkin dengan nilai y.

Istilah neuron atau *node* digunakan pada ANN untuk menandakan suatu atribut (pada layer input), suatu jenis *output* (pada layer *output*) atau suatu ekspresi matematika (pada layer tersembunyi). Neuron atau *node* ini masing-masing memiliki *weight* (bobot) yang berbeda-beda yang mempengaruhi nilai *output* dari *node* tersebut berdasarkan suatu fungsi aktivasi. Untuk *node* pada layer *non-output*, *output node* tersebut akan dikirim ke seluruh *node* pada layer selanjutnya. Koneksi atau sambungan antara *node* memiliki nilai bias.

Misalkan diberikan suatu input fitur x_1, x_2, x_3 yang secara kolektif disebut *input layer*, empat unit tersembunyi yang secara kolektif disebut *hidden layer* dan satu *output*

neuron yang disebut *output layer*. Unit tersembunyi yang pertama membutuhkan input x_1, x_2, x_3 dan memberikan *output* a_1 (a untuk nilai "aktivasi"). Pada contoh ini, hanya terdapat satu hidden layer, namun hidden layer dapat berjumlah lebih dari satu. Misalkan ${a_1}^{[1]}$ menyatakan nilai *output* dari hidden unit pertama pada hidden layer pertama. Index berbasis 0 digunakan untuk penomoran index dari layer. Input layer adalah layer 0, hidden layer pertama adalah layer 1, dan *output* layer adalah layer 2. Dengan notasi matematika, *output* dari layer 2 adalah ${a_1}^{[2]}$. Notasi ini dapat disatukan:

$$x_1 = {a_1}^{[0]}$$
$$x_2 = {a_2}^{[0]}$$
$$x_3 = {a_3}^{[0]}$$

di mana ${a_j}^{[l]}$ menyatakan nilai *output* dari unit ke j pada layer ke l.

Unit pertama pada hidden layer melakukan perhitungan:

$${z_1}^{[1]} = {W_1}^{[1]^T} x + {b_1}^{[2]} \text{ dan } {a_1}^{[1]} = g({z_1}^{[1]})$$

di mana W adalah matrix parameter dan W_1 menyatakan baris pertama dari matrix W. Parameter yang terkait dengan hidden unit pertama adalah vektor ${W_1}^{[1]} \in \mathbb{R}^3$ dan skalar ${b_1}^{[1]} \in \mathbb{R}$.

Multilayer Perceptron (MLP) adalah salah satu jenis Artificial Neural Network yang memiliki layer tesembunyi atau hidden layer. MLP terdiri dari setidaknya 3 layer, yaitu input layer, hidden layer dan *output* layer. Secara praktis, MLP biasanya memiliki lebih dari satu hidden layer. Unit tersembunyi ke dua dan ketika dari hidden layer pertama, perhitungannya didefinisikan sebagai:

$${z_2}^{[1]} = {W_2}^{[1]^T} x + {b_2}^{[1]} \text{ dan } {a_2}^{[1]} = g\left({z_2}^{[1]}\right)$$
$${z_3}^{[1]} = {W_3}^{[1]^T} x + {b_3}^{[1]} \text{ dan } {a_3}^{[1]} = g({z_3}^{[1]})$$

di mana setiap unit tersembunyi masing-masing memiliki parameter W dan b tersendiri. *Output* layer melakukan perhitungan:

$$z_1{}^{[2]} = {W_1{}^{[2]}}^T a^{[1]} + b_1{}^{[2]} \text{ dan } a_1{}^{[2]} = g(z_1{}^{[2]})$$

di mana $a^{[1]}$ didefinisikan sebagai vektor dari seluruh nilai aktivasi dari layer 1:

$$a^{[1]} = \begin{bmatrix} a_1{}^{[1]} \\ a_2{}^{[1]} \\ a_3{}^{[1]} \\ a_4{}^{[1]} \end{bmatrix}$$

Nilai aktivasi $a_1{}^{[2]}$ dari layer 2, yang berupa nilai skalar yang didefinisikan oleh $a_1{}^{[2]} = g(z_1{}^{[2]})$, melambangkan prediksi nilai *output* terakhir dari neural network tersebut.

Menggunakan matrix aljabar, nilai aktivasi dapat dihitung:

$$\underbrace{\begin{bmatrix} z_1{}^{[1]} \\ \vdots \\ z_4{}^{[1]} \end{bmatrix}}_{z^{[1]} \in \mathbb{R}^{4x1}} = \underbrace{\begin{bmatrix} \cdots & {W_1{}^{[1]}}^T & \cdots \\ \cdots & {W_2{}^{[1]}}^T & \cdots \\ & \vdots & \\ \cdots & {W_4{}^{[1]}}^T & \cdots \end{bmatrix}}_{W^{[1]} \in \mathbb{R}^{4x3}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x \in \mathbb{R}^{3x1}} + \underbrace{\begin{bmatrix} b_1{}^{[1]} \\ b_2{}^{[1]} \\ \vdots \\ b_4{}^{[1]} \end{bmatrix}}_{b^{[1]} \in \mathbb{R}^{4x1}}$$

Di mana $\mathbb{R}^{dxn}$ mengindikasikan dimensi matrix.

Berdasarkan operasi matrix di atas, diberikan suatu input $x \in \mathbb{R}^3$, nilai aktivasi dari hidden layer dihitung dengan $z^{[1]} = W^{[1]}x + b^{[1]}$ dan $a^{[1]} = g(z^{[1]})$. Sedangkan untuk nilai aktivasi dari layer *output* dihitung:

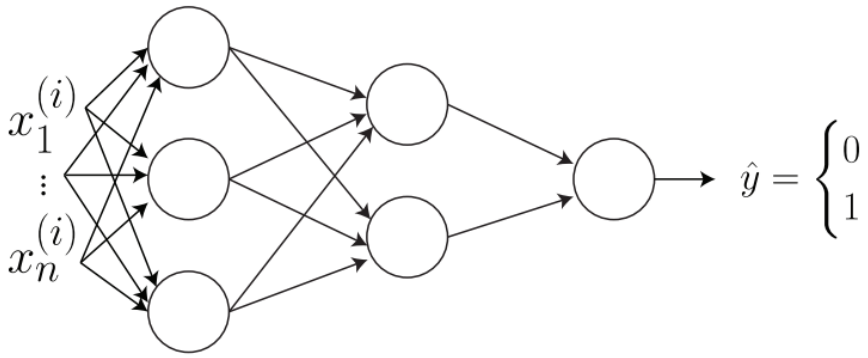
$$\underbrace{z^{[2]}}_{1x1} = \underbrace{W^{[2]}}_{1x4} \underbrace{a^{[1]}}_{4x1} + \underbrace{b^{[2]}}_{1x1} \text{ dan } \underbrace{a^{[2]}}_{1x1} = g(\underbrace{z^{[2]}}_{1x1})$$

Jika diberikan suatu training set dengan tiga sampel. Nilai aktivasi dari setiap sampel adalah:

$$z^{[1](1)} = W^{[1]}x^{(1)} + b^{[1]}$$
$$z^{[1](2)} = W^{[1]}x^{(2)} + b^{[1]}$$
$$z^{[1](3)} = W^{[1]}x^{(3)} + b^{[1]}$$

Dengan $[i]$ melambangkan layer ke-i dan (j) melambangkan sampel training ke-j.

Dilihat dari Gambar 5, seluruh input adalah vektor ${x_i}^1, \dots, {x_n}^{(i)}$. Pada hidden layer pertama, seluruh input tersambung ke seluruh neuron pada layer berikutnya. Layer ini disebut *fully connected* layer.



Gambar 5: Neural Network

Langkah berikutnya adalah menghitung berapa banyak parameter pada jaringan ini:

$$z^{[1]} = W^{[1]}x^{(i)} + b^{[1]}$$
$$a^{[1]} = g\left(z^{[1]}\right)$$
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$a^{[2]} = g\left(z^{[2]}\right)$$
$$z^{[3]} = W^{[3]}a^{[2]} + b^{[3]}$$
$$\hat{y}^{(i)} = a^{[3]} = g(z^{[3]})$$

Diketahui bahwa $z^{[1]}, a^{[1]} \in \mathbb{R}^{3x1}$ dan $z^{[2]}, a^{[2]} \in \mathbb{R}^{2x1}$ dan $z^{[3]}, a^{[3]} \in \mathbb{R}^{1x1}$. Untuk $W^{[1]}$ dapat dideduksi dari:

$$z^{[1]} = W^{[1]}x^{(i)}$$

yang size-nya dapat ditulis:

$$\mathbb{R}^{3x1} = \mathbb{R}^{?x?}x\ \mathbb{R}^{nx1}$$

Menggunakan perkalian matriks, dapat disimpulkan bahwa size $W^{[1]}$ adalah 3 x n. Dengan cara yang serupa untuk setiap hidden layer. Disimpulkan:

$$W^{[2]} \in \mathbb{R}^{2x3}, b^{[2]} \in \mathbb{R}^{3x1} \text{ dan } W^{[3]} \in \mathbb{R}^{3x1}, b^{[3]} \in \mathbb{R}^{3x1}$$

Secara keseluruhan, ada $3n + 3$ pada layer pertama, $2 * 3 + 2$ pada layer kedua dan $2 + 1$ pada layer terakhir, dengan total $3n + 14$ parameter. Parameter-parameter tersebut diinisialisasikan secara acak dengan distribusi normal sekitar nol; $N(0, 0.1)$. Kemudian parameter-parameter ini diestimasi menggunakan *gradient descent*.

Untuk memperbarui (*update)* parameter-parameter ini, input vektor harus melewati seluruh neuron pada neural network dan *output*-nya dinotasikan dengan suatu nilai prediksi $\hat{y}$. Error atau loss dari $\hat{y}$, dinotasikan $\mathcal{L}$, dapat dihitung:

$$\mathcal{L}(\hat{y}, y) = -[(1 - y)\log(1 - \hat{y}) + y\, log\hat{y}]$$

Fungsi *loss* $\mathcal{L}(\hat{y}, y)$ menghasilkan suatu nilai skalar. Seluruh parameter pada setiap layer di neural network diperbaharui berdasarkan nilai ini. Untuk setiap layer pada index ℓ, perbaharui parameter:

$$W^{[\ell]} = W^{[\ell]} - \alpha \frac{\partial \mathcal{L}}{\partial W^{[\ell]}}$$

$$b^{[\ell]} = b^{[\ell]} - \alpha \frac{\partial \mathcal{L}}{\partial b^{[\ell]}}$$

di mana α adalah learning rate.

Pada tahap optimisasi, turunan (*gradient*) untuk $W^{[3]}$ diutamakan untuk dihitung terlebih dahulu. Hal ini karena pengaruh $W^{[1]}$ terhadap *loss* lebih kompleks dari $W^{[3]}$ yang terletak lebih dekat ke *output* $\hat{y}$, dalam hal jumlah perhitungan. Dengan menggunakan sifat turunan sigmoid pada persamaan (II.22), turunan $W^{[3]}$ adalah:

$$\frac{\partial \mathcal{L}}{\partial W^{[3]}} = -\frac{\partial}{\partial W^{[3]}}\left((1-y)\log(1-\hat{y}) + y\log\hat{y}\right)$$

$$= -(1-y)\frac{\partial}{\partial W^{[3]}}\log\left(1 - g\left(W^{[3]}a^{[2]} + b^{[3]}\right)\right)$$

$$- y\frac{\partial}{\partial W^{[3]}}\log\left(g\left(W^{[3]}a^{[2]} + b^{[3]}\right)\right)$$

$$= -(1-y)\frac{1}{1 - g(W^{[3]}a^{[2]} + b^{[3]})}(-1)g'\left(W^{[3]}a^{[2]} + b^{[3]}\right){a^{[2]}}^T$$

$$- \quad y\frac{1}{g(W^{[3]}a^{[2]} + b^{[3]})}g'^{(W^{[3]}a^{[2]}+b^{[3]})}{a^{[2]}}^T \qquad \text{(II.44)}$$

$$= (1-y)\sigma\left(W^{[3]}a^{[2]} + b^{[3]}\right){a^{[2]}}^T - y(1$$

$$- \sigma\left(W^{[3]}a^{[2]} + b^{[3]}\right){a^{[2]}}^T$$

$$= (1-y)a^{[3]}{a^{[2]}}^T - y\left(1 - a^{[3]}\right){a^{[2]}}^T$$

$$= \left(a^{[3]} - y\right){a^{[2]}}^T$$

dengan $a^{[3]} = \sigma\left(W^{[3]}a^{[2]} + b^{[3]}\right)$.

Turunan $W^{[2]}$ dapat dihitung melalui aturan rantai (*chain rule*). Diketahui bahwa $\mathcal{L}$ bergantung pada $\hat{y} = a^{[3]}$:

$$\frac{\partial \mathcal{L}}{\partial W^{[2]}} = \frac{\partial \mathcal{L}}{?}\frac{?}{\partial W^{[2]}}$$

$\partial a^{[3]}/\partial a^{[3]}$ dapat disisipkan:

$$\frac{\partial \mathcal{L}}{\partial W^{[2]}} = \frac{\partial \mathcal{L}}{\partial a^{[3]}}\frac{\partial a^{[3]}}{?}\frac{?}{\partial W^{[2]}}$$

lebih lanjut:

$$\frac{\partial \mathcal{L}}{\partial W^{[2]}} = \frac{\partial \mathcal{L}}{\partial a^{[3]}}\frac{\partial a^{[3]}}{\partial z^{[3]}}\frac{\partial z^{[3]}}{\partial a^{[3]}}\frac{\partial a^{[2]}}{\partial z^{[2]}}\frac{\partial z^{2}}{\partial W^{[2]}} \qquad \text{(II.45)}$$

Dengan menggunakan persamaan (II.44), persamaan (II.45) menjadi:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial W^{[2]}} &= \frac{\partial \mathcal{L}}{\underbrace{\partial a^{[3]}}_{a^{[3]}-y}} \frac{\partial a^{[3]}}{\partial z^{[3]}} \frac{\partial z^{[3]}}{\underbrace{\partial a^{[3]}}_{W^{[3]}}} \frac{\partial a^{[2]}}{\underbrace{\partial z^{[2]}}_{g'(z^{[2]})}} \frac{\partial z^{2}}{\underbrace{\partial W^{[2]}}_{a^{[1]}}} \\ &= (a^{[3]} - y)\, W^{[3]} g'\left(z^{[2]}\right) a^{[1]} \\ &= {W^{[3]}}^{T} \cdot g'(z^{[2]})(a^{[3]} - y)\, {a^{[1]}}^{T}\end{aligned} \tag{II.46}$$

Parameter $W^{[1]}$ dapat dihitung melalui aturan rantai menggunakan persamaan (II.44) dan persamaan (II.46).

Dengan seluruh parameter telah diestimasi oleh gradient descent, model akhir MLP dapat diekspresikan sebagai berikut:

$$\hat{y} = v_0 + \sum_{j=1}^{NH} b_j g\left(W_j^T x'\right) \tag{II.47}$$

di mana x' adalah vektor input x dengan $x' = (1, x^T)^T$. Fungsi g adalah fungsi aktivasi pada *node* tersembunyi. Fungsi aktivasi adalah fungsi yang memberi *output* dari suatu *node* berdasarkan input yang diberikan untuk diteruskan ke *node* selanjutnya.

Fungsi aktivasi Sigmoid logistik adalah fungsi aktivasi yang memberi nilai antara 0 dan 1:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{II.48}$$

Fungsi aktivasi Sigmoid tangensial (tanh) adalah fungsi aktivasi yang memberi nilai antara -1 dan 1

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{II.49}$$

Fungsi aktivasi relu adalah fungsi aktivasi yang mengembalikan nilai nol untuk seluruh input negatif namun untuk seluruh input positif x, ReLU mengembalikan nilai x itu sendiri.

$$f(x) = f(x) = \max(0, x) \tag{II.50}$$

(Ahmed, Atiya, Gayar, & El-Shishiny, 2010; Zurada, 1992; Haykin, 1994; Zhou & Liu, 2006; Bishop, Pattern Recognition and Machine Learning, 2006; Bishop, Neural Networks for Pattern Recognition, 1995; MULTILAYER, 1998; Ng & Katanforoosh, 2018)

2.8.5 K-*Nearest Neighbor*

K-*Nearest Neighbor* adalah algoritma prediksi non-parametrik di mana hasil prediksi kelas dari suatu titik didasarkan oleh mayoritas kelas dari k tetangga terdekatnya. Diberikan suatu titik, hitung jarak Euklid sesuai dengan persamaan (II.5) antara titik tersebut dengan semua titik pada data training. Kemudian pilih k tetangga terdekat berdasarkan jarak Euklid tersebut, prediksi kelas dari titik tersebut adalah modus kelas dari k tetangga terdekatnya.

Untuk suatu sampel x dengan k tetangga terdekat dari B, prediksi kelas ditentukan oleh:

$$\hat{y} = argmax_r \left(\sum_{i=1}^{k} y_i (y_i = r) \right) \tag{II.51}$$

di mana y_i adalah label kelas dari x_i (Ahmed, Atiya, Gayar, & El-Shishiny, 2010; Osisanwo, et al., 2017).

Weighted-KNN, atau KNN berbobot adalah jenis KNN yang memberi bobot w kepada k tetangga terdekat berdasarkan invers jarak d. Prediksi kelas dipilih berdasarkan jumlah bobot tertinggi dari tiap kelas yang ada, diekspresikan:

$$\hat{y} = argmax_r \left(\sum_{i=1}^{k} w_i * I(y_i = r) \right) \tag{II.52}$$

(Hechenbichler, 2004)

BAB III
METODE PENELITIAN

3.1 Waktu dan Tempat

Penelitian ini dilaksanakan dari bulan Agustus 2019 sampai dengan bulan Oktober 2019. Lokasi penelitian dilakukan di Laboratorium Rekayasa Perangkat Lunak Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Hasanuddin.

3.2 Tahapan Penelitian

Untuk menyelesaikan penelitian ini, peneliti akan melewati beberapa tahap penelitian, yaitu: Pra-penelitian, eksplorasi dan preprocessing data, model *tuning and fitting*, dan analisis hasil.

Pada tahap pra-penelitian, peneliti menentukan tema penelitian, masalah yang akan diteliti, mengumpulkan sumber referensi atau literatur seperti jurnal dan buku yang mendukung dalam penelitian, dan menentukan metode yang digunakan beserta batasan masalahnya. Kemudian peneliti mencari data yang sesuai dengan tema penelitian sebagai objek penelitian.

Pada tahap eksplorasi dan preprocessing data, peneliti mencoba menguraikan karakteristik-karakteristik setiap dataset sebagai informasi untuk mengambil keputusan pada tahap preprocessing. Peneliti akan mengidentifikasi masalah-masalah yang terdapat pada dataset tersebut kemudian mengambil pendekatan untuk menyelesaikan masalah yang terkait. Normalisasi dan *attribute reduction* termasuk pada tahap ini.

Pada tahap model *tuning and fitting*, peneliti akan mencari parameter-parameter terbaik untuk model yang akan digunakan berdasarkan hasil eksplorasi data dan *trial and error* untuk mendapatkan hasil terbaik. Tuning juga dilakukan terhadap beberapa teknik resampling yang membutuhkan parameter. Kemudian model akan memberi hasil prediksi yang akan dianalisis pada tahap selanjutnya.

Pada tahap analisis hasil, peneliti akan merangkum hasil yang diperoleh dari metode-metode yang digunakan ke dalam bentuk tabel dan diagram, kemudian menyimpulkan hasilnya sebagai output dari penelitian ini.

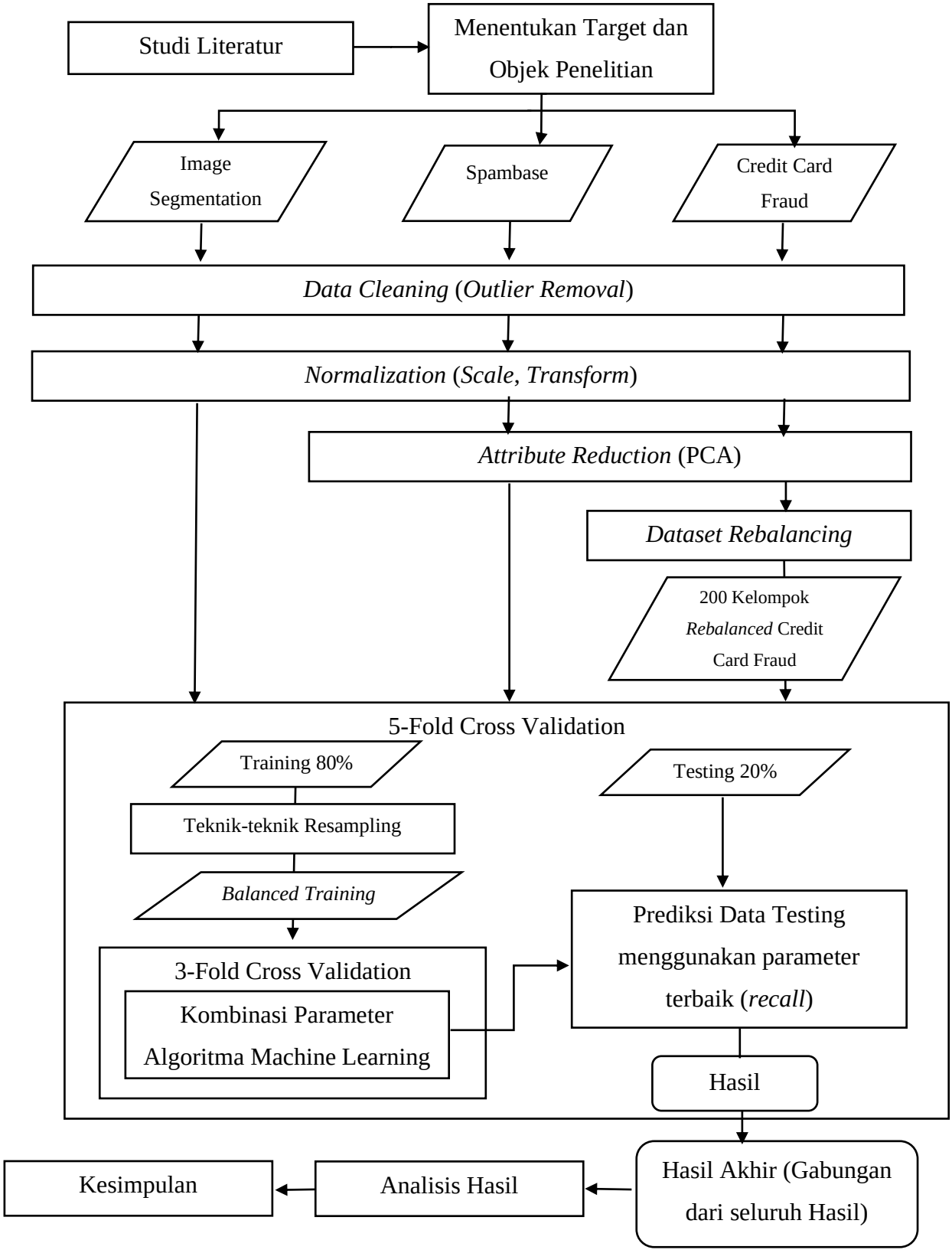
3.3 Deskripsi Data

Data diambil dari Website resmi Kaggle (kaggle.com), UCI Machine Learning Repository (archive.ics.uci.edu/ml/) dan KEEL (sci2s.ugr.es/keel/imbalanced.php). Data tersebut berupa tiga dataset, yaitu:

1. Credit Card Fraud Dataset (Kaggle), yang terdiri dari 30 kolom attribut dengan 1 kolom kelas, 284.807 baris. 284.315 jumlah sampel kelas mayoritas dan 492 jumlah sampel kelas minoritas dengan *imbalanced ratio* sebesar 577:1. Dataset ini merupakan dataset terpopuler di Kaggle sebab jumlah data yang besar dengan imbalanced ratio yang sangat tinggi.
2. Spambase Dataset (UCI), yang terdiri dari 57 kolom atribut dengan 1 kolom kelas, 4.601 baris. 2788 jumlah sampel kelas mayoritas dan 1813 jumlah sampel kelas minoritas dengan *imbalanced ratio* sebesar 1,5:1
3. Image Segmentation Dataset (KEEL), yang terdiri dari 19 kolom atribut dengan 1 kolom kelas, 2308 baris. 1962 jumlah sampel kelas mayoritas dan 346 jumlah sampel kelas minoritas dengan *imbalanced ratio* sebesar 6:1.

Seluruh dataset hanya memiliki atribut kontinu dengan label kelas biner yang sesuai dengan tema penelitian dan metode-metode yang digunakan.

3.4 Alur Penelitian



BAB IV
HASIL DAN PEMBAHASAN

4.1 Eksplorasi dan *Preprocessing* Data

4.1.1 Image Segmentation Dataset

4.1.1.1 *Data Cleaning* & *Normalization*

Image Segmentation dataset merupkan dataset mengenai citra di mana setiap pikselnya memiliki kelas berdasarkan hasil segmentasi manual dari citra outdoor. Setiap observasi dari dataset ini adalah gabungan piksel yang berukuran 3x3 (disebut *region*). Pada dataset asli terhadap enam kelas berbeda, namun dataset ini diubah oleh Keel di mana hanya terdapat 2 kelas, yaitu positif dan negatif. Kelas positif adalah kelas 0 pada dataset asli, dan kelas negatif adalah kelas 1, 2, 3, 4 dan 5 pada dataset asli. Perubahan ini dilakukan untuk menciptakan dataset yang tidak seimbang. Terdapat 20 atribut pada dataset ini:

1. 'region_centroid_row' dan 'region_centroid_col' menunjukkan lokasi *centroid* dari region observasi. region_centroid_row menunjukkan lokasi baris dari centroid, dan region_centroid_col menunjukkan lokasi kolom dari *centroid*. Atribut-atribut ini bersifat bilangan bulat [0, 255].
2. 'region_pixel_count' berisi informasi tentang jumlah piksel dari *region* observasi. Atribut ini bersifat bilangan bulat dan hanya terdapat satu jenis nilai pada atribut ini {9}.
3. 'short_line_density-5' dan 'short_line_density-2' berisi informasi mengenai hasil dari algoritma ekstraksi garis yang melewati *region* tersebut. short_line_density-5 menghitung garis dengan kontras rendah (lebih kecil dari atau sama dengan 5) dan short_line_density-2 menghitung garis dengan kontras tinggi (lebih besar dari 5). Atribut-atribut ini bersifat riil kontinu [0, 1]
4. 'vedge_mean', 'vedge_sd', 'hedge_mean', 'hedge_sd' berisi informasi mengenai piksel-piksel yang bertetangga dengan *region* tersebut. Terdapat 6

piksel pada masing-masing vedge dan hedge, yaitu 6 piksel tetangga secara horizontal dan 6 piksel tetangga secara vertikal. Rata-rata dari 6 piksel tersebut dimasukkan dalam vedge_mean dan hedge_mean, kemudian standar deviasi dari 6 piksel tersebut dimasukkan dalam vedge_sd dan hedge_sd. Atribut-atribut ini bersifat riil kontinu $[0, \infty]$

5. 'rawred_mean', 'rawgreen_mean', dan 'rawblue_mean', berisi informasi mengenai rata-rata dari masing-masing warna merah (R), hijau (G), dan biru (B) dari *region* tersebut. Atribut-atribut ini bersifat riil kontinu [0, 255]
6. 'intensity_mean' adalah rata-rata dari seluruh warna pada *region* tersebut, dihitung dari:

$$\frac{R + G + B}{3}$$

Atribut ini bersifat riil kontinu [0, 255]

7. 'exred_mean', 'exgreen_mean', dan 'exblue_mean', berisi informasi mengenai kelebihan (*excess*) dari masing-masing warna merah, hiau, dan biru. exred_mean dihitung dari:

$$2R - (G + B)$$

exgreen_mean dihitung dari:

$$2G - (R + B)$$

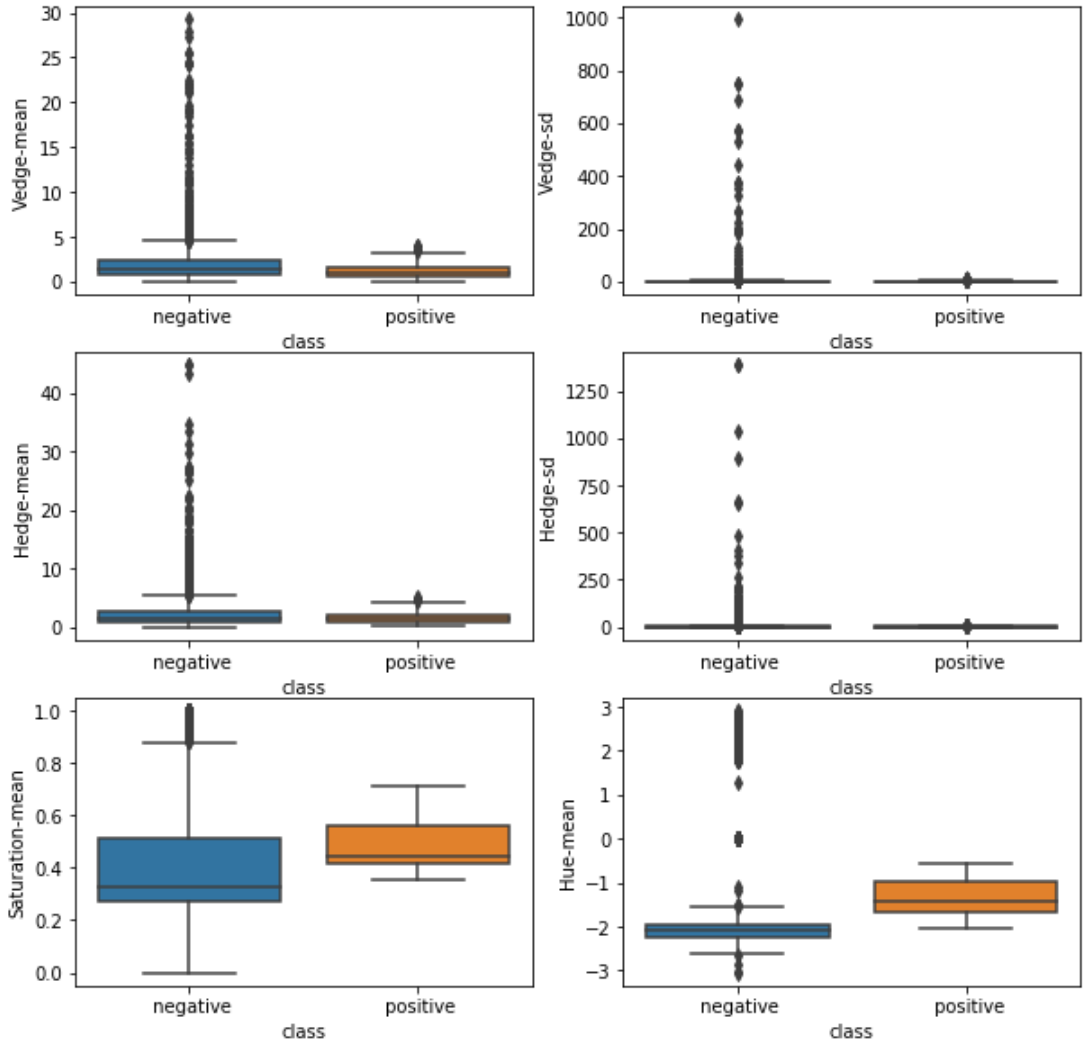
exblue_mean dihitung dari:

$$2B - (R + G).$$

Atribut-atribut ini bersifat riil kontinu [-256, 255]

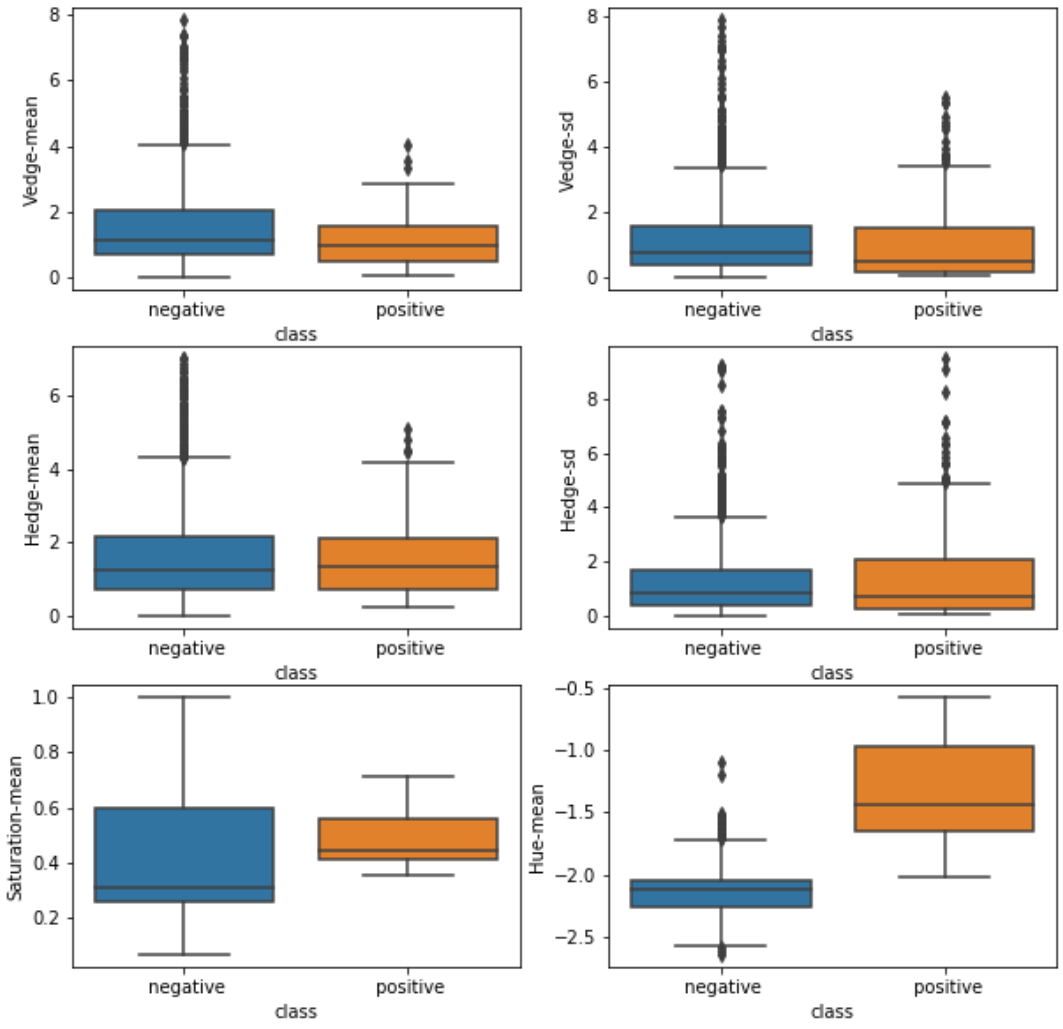
8. 'value_mean', 'hue_mean', dan 'saturation_mean', yaitu hasil transformasi linear 3D dari RGB (menggunakan algoritma dari buku Foley and VanDam, *Fundamentals of Interactive Computer Graphics*). Atribut-atribut ini bersifat riil kontinu, value_mean memiliki range [0, 255], hue_mean memiliki range [-100,100], dan saturation_mean memiliki range [0, 1].
9. Atribut class menunjukkan *region* adalah kelas positif atau negatif. Atribut ini bersifat kategorik {'positive', 'negative'}.

Image Segmentation dataset ini termasuk kategori dataset tak seimbang yang di mana kelas mayoritasnya adalah *region* berkategori negatif dengan jumlah 1962 dan kelas minoritas adalah *region* berkategori positif dengan jumlah 346.



Gambar 6: Boxplot sebelum Outlier Removal - Image Segmentation Dataset

Gambar 6 menunjukkan bahwa terdapat banyak outlier ekstrim yang tersebar jauh dari 90% sebaran data. Namun outlier ekstrim tesebut hanya tersebar pada kelas negatif (mayoritas) saja, yang berarti bahwa menghapus sebagian outlier ini tidak berdampak terhadap kelas positif (minoritas).



Gambar 7: Boxplot setelah Outlier Removal – Image Segmentation Dataset

Gambar 7 menunjukkan dengan jelas distribusi masing-masing kelas pada 6 atribut di atas setelah *outlier removal*.

Seluruh atribut telah diuji menggunakan Saphiro-Wilk test dengan *p-value* kurang dari dari 0.05 (Lampiran 2).

4.1.2 Spambase Dataset

4.1.2.1 *Data Cleaning* & *Normalization*

Spambase dataset merupkan dataset yang menunjukkan email berlabel spam atau ham. Email berlabel spam pada dataset ini diperoleh dari kantor pos dan orang-orang yang mengajukan spam pada emailnya. Email berlabel ham (non-spam) diperoleh dari email kerja dan email pribadi seseorang yang bernama George. Indikator non-spam adalah kata ‘George’ dan kode area ‘650’ yang hanya diketahui oleh orang-orang yang mengenal George. Dengan menggunakan *keyword* tertentu seperti ‘George’ dan ‘650’, *personalized spam filter*.Terdapat 58 atribut dalam dataset ini:

1. 48 atribut dalam format word_freq_WORD menunjukkan persentase dari frekuensi kemunculan kata WORD dari total kata pada email. Contoh: word_freq_george dan word_freq_order. Atribut-atribut ini bersifat riil kontinu [0, 100].
2. 6 atribut dalam format char_freq_CHAR menunjukkan persentase dari frekuensi kemunculan karakter CHAR dari total karakter pada email. Contoh: char_freq_(dan char_freq_!. Atribut-atribut ini bersifat riil kontinu [0, 100].
3. 1 atribut capital_run_length_average menunjukkan rata-rata panjang huruf kapital berurutan. Atribut ini bersifat riil kontinu [1, ...].
4. 1 atribut capital_run_length_longest menunjukkan jumlah terbanyak panjang huruf kapital berurutan. Atribut ini bersifat bilangan bulat [1, ...].
5. 1 atribut capital_run_length_total menunjukkan jumlah dari huruf kapital pada email. Atribut ini bersifat bilangan bulat [1, ...].
6. 1 atribut class menunjukkan email termasuk kategori spam (1) atau ham (0). Atribut ini bersifat nominal {0, 1}.

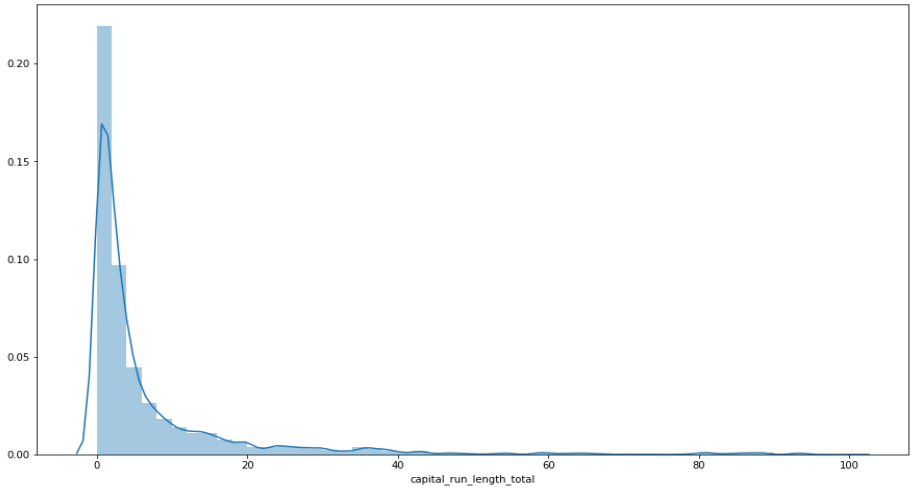
Spambase dataset ini termasuk kategori dataset tak seimbang yang di mana kelas mayoritasnya adalah email berkategori ham dengan jumlah 2788 dan kelas minoritas adalah email berkategori spam dengan jumlah 1813.

Sebagian besar atribut pada dataset ini memiliki ditribusi yang sama, yaitu 48 atribut word_freq_WORD dan 6 atribut char_freq_CHAR yang memiliki rentang nilai [1, 100]. Tiga atribut capital_run_length_average/longest/total memilki distribusi yang jauh berbeda dengan 54 atribut sebelumnya.

Tabel 2: Distribusi capital_run_length_average, capital_run_length_longest dan capital_run_length_total

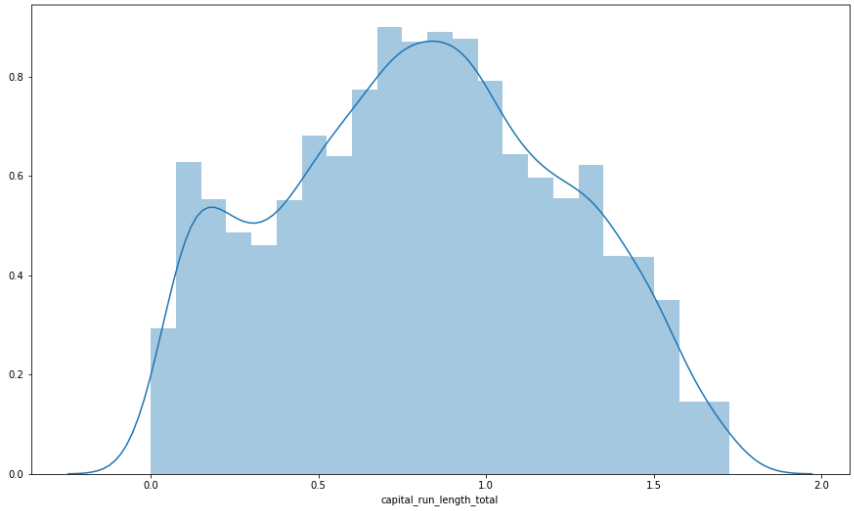
	capital_run_length_average	capital_run_length_longest	capital_run_length_total
count	4601.000000	4601.000000	4601.000000
mean	5.191515	52.172789	283.289285
std	31.729449	194.891310	606.347851
min	1.000000	1.000000	1.000000
25%	1.588000	6.000000	35.000000
50%	2.276000	15.000000	95.000000
75%	3.706000	43.000000	266.000000
max	1102.500000	9989.000000	15841.000000

Tabel 2 menunjukkan bahwa nilai maksimal dari masing-masing atribut berbeda jauh dari rentang nilai [1, 100]. Maka atribut-atribut ini perlu dinormalkan ke skala [1, 100] menggunakan MinMax. Namun terdapat lonjakan yang sangat besar dari kuartil ketiga (75%) ke nilai maksimal dari masing-masing atribut di atas. Hal ini mengindikasikan terdapat outlier ekstrim yang memberi bias terhadap standar deviasi atribut, dan memberi dampak negatif jika data diskalakan dengan outlier. Maka normalisasi akan dilakukan setelah outlier ekstrim telah dihapus dari dataset.



Gambar 8: Distribusi capital_run_length_total sebelum normalisasi

Sebagian besar atribut dari Spambase dataset condong ke kiri (*left-skewed*) seperti Gambar 8. Normalisasi dilakukan menggunakan transformasi log pada setiap atribut yang condong ke kiri.



Gambar 9: Distribusi capital_run_length_total setelah normalisasi

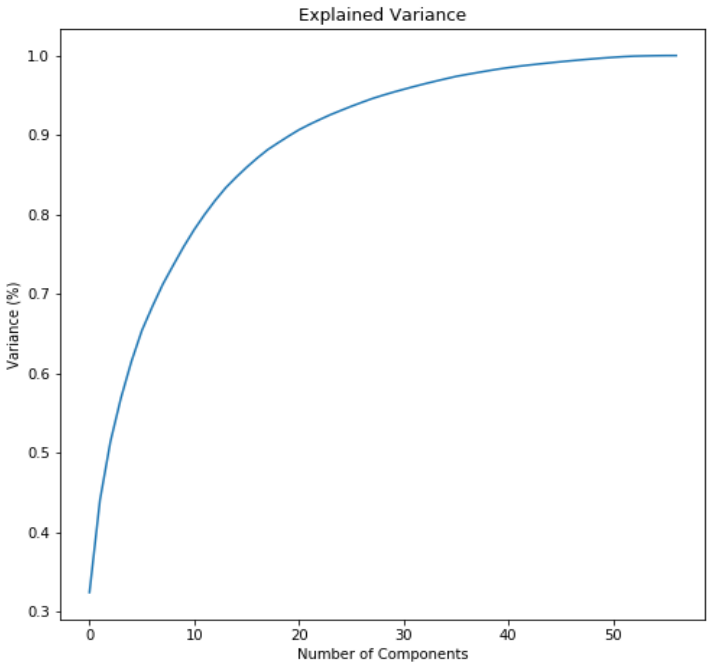
Gambar 9 menunjukkan distribusi dari capital_run_length_total setelah normalisasi. Seluruh atribut telah diuji menggunakan Saphiro-Wilk test dengan *p-value* kurang dari dari 0.05 (Lampiran 1).

4.1.2.2 *Attribute Reduction*

Atribut atau dimensi dataset perlu dikurangi untuk memberi hasil yang lebih umum, mengurangi *overfitting*, dan mengurangi *running time*. Terlebih jika dataset tersebut memiliki atribut yang sangat banyak atau data yang sangat besar.

Pada Spambase dataset terdapat 58 atribut independen yang ditransformasi dan direduksi menggunakan PCA. Atribut yang ditransformasi disebut komponen. Setiap jumlah komponen dilakukan uji sebanyak 5 kali *cross validation* yang dirata-ratakan.

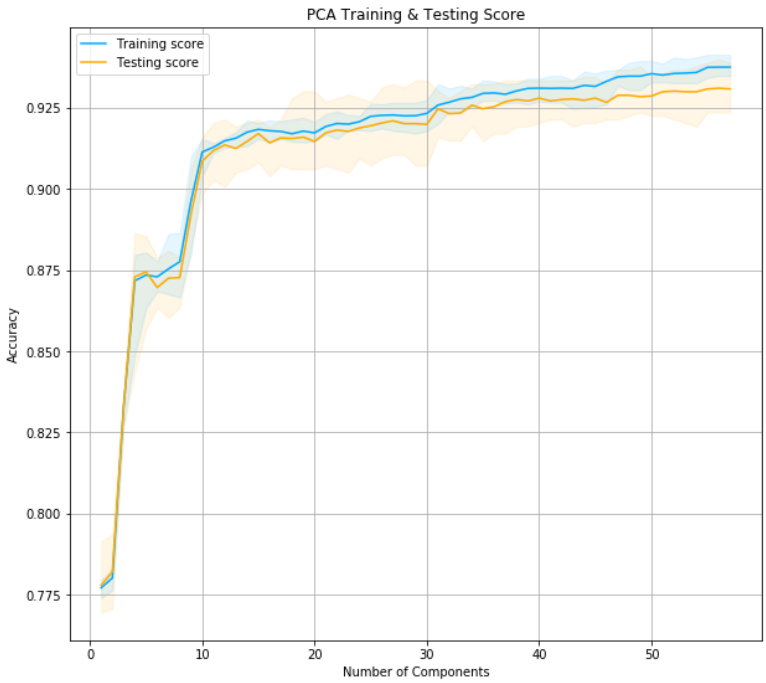
Gambar 10 menunjukkan variansi yang dijelaskan oleh komponen-komponen utama PCA. Sumbu X menyatakan jumlah komponen dan sumbu Y menyatakan total variansi yang dikandung untuk setiap jumlah komponen secara kumulatif. Pemilihan jumlah komponen dapat dilakukan dengan menggunakan *scree plot* seperti pada Gambar 10. Biasanya, dengan menentukan total jumlah variansi yang diinginkan, dapat menentukan jumlah komponen yang diperlukan untuk analisis selanjutnya.



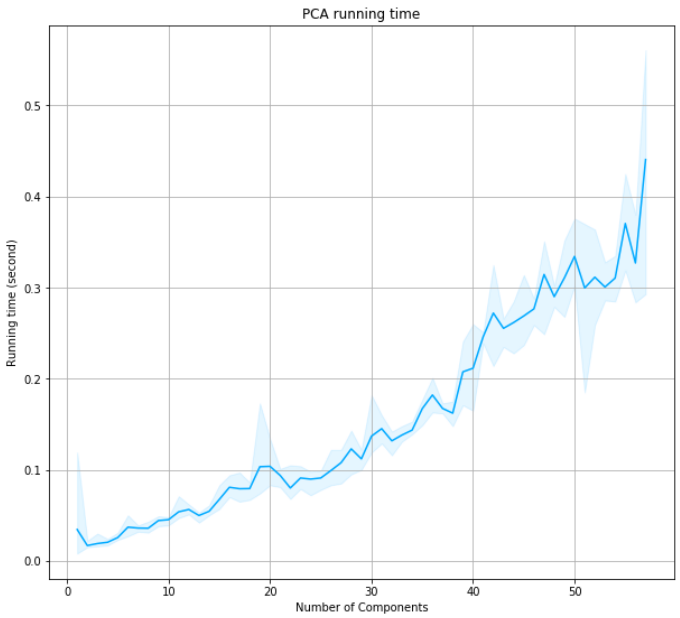
Gambar 10: *Scree plot* pada Spambase Dataset

Akurasi jumlah komponen yang diperlukan untuk analisis lanjutan, selain menggunakan *scree plot*, perlu juga mempertimbangkan performa pada tahap *training*, *testing* dan *running time*.

Gambar 11 menunjukkan bahwa dengan menggunakan 58 komponen menjamin akurasi training dan testing yang tertinggi. Gambar 11 juga menunjukkan bahwa 10 komponen pertama berkontribusi besar terhadap akurasi model, dengan rata-rata akurasi sebesar 91% dengan perbedaan kurang dari 2% dibanding dengan menggunakan seluruh komponen. Model yang menggunakan 58 komponen memperoleh testing score yang tinggi hanya dengan sedikit perbedaan dari training score-nya. namun perbedaan dari training score dan testing score terlihat jelas setelah 30 komponen pertama, dengan perbedaan atau bias terbesar terjadi ketika seluruh komponen digunakan.



Gambar 11: Training & Testing Score Spambase dataset berdasarkan jumlah komponen utama PCA



Gambar 12: Running time dari satu fit berdasarkan jumlah komponen utama PCA

Dari sisi *running time*, menggunakan 10 komponen pertama memiliki perbedaan *running time* yang sangat besar hingga 8 kali lebih cepat.

Berdasarkan hasil analisis dari Gambar 10, Gambar 11 dan Gambar 12, dapat disimpulkan bahwa jumlah komponen teroptimal untuk digunakan adalah 10 komponen. Data mengenai Gambar 10, Gambar 11 dan Gambar 12 dapat dilihat pada Lampiran 7.

4.1.3 Credit Card Fraud Dataset

4.1.3.1 *Data Cleaning & Normalization*

Credit Card Fraud dataset merupakan dataset berisi transaksi yang dilakukan dengan kartu kredit pada September 2013 di Eropa yang terjadi dalam dua hari. Dataset ini berisi informasi mengenai transaksi yang bersifat fraud dan non-fraud. Terdapat 31 atribut dalam dataset ini:`

1. ‘Time’, yang berisi informasi mengenai waktu yang telah berlalu (detik) mengenai masing-masing transaksi sejak transaksi pertama pada dataset ini.

2. 28 Fitur V1 hingga V28 merupakan fitur hasil transformasi PCA yang dilakukan oleh instansi yang mengembangkan dataset ini untuk melindungi informasi sensitif dari kartu kredit.
3. 'Amount' berisi informasi mengenai jumlah hasil transaksi dalam USD $[0, \infty]$.
4. Atribut class yang menunjukkan apakah suatu transaksi bersifat fraud atau non-fraud. Transaksi fraud berlabel 1 dan non-fraud berlabel 0. Atribut ini bersifat nominal {0, 1}.

Dataset ini berisi 284.807 total transaksi dengan 284.315 transaksi non-fraud dan 492 transaksi fraud. Dataset ini sangat tidak seimbang dengan kelas minoritas hanya berkisar sebesar 0,17% dari total transaksi.

Sebagian besar atribut pada dataset ini memiliki ditribusi yang sama, yaitu 28 atribut V1 hingga V28 yang memiliki mean 0 dan standar deviasi mendekati 1. Dua atribut Amount dan Time memilki distribusi yang jauh berbeda dengan 28 atribut tersebut.

Tabel 3: Karakteristik atribut dari Credit Card Fraud Dataset

	Time	*V1*	*V2*	...	*V27*	*V28*	*Amount*
mean	94813	0	0	...	0	0	88.349
min	0.000	-5.641e+01	-7.271e+01	...	-2.256e+01	-1.543e+01	0.000
25%	54201.500	-9.203e-01	-5.985e-01	...	-7.083e-02	-5.295e-02	5.600
50%	84692.000	1.810e-02	6.548e-02	...	1.342e-03	1.124e-02	22.000
75%	139320.500	1.315e+00	8.037e-01	...	9.104e-02	7.827e-02	77.165
max	172792.000	2.454e+00	2.205e+01	...	3.161e+01	3.384e+01	25691.160

Tabel 3 menunjukkan bahwa Time dan Amount memiliki mean dan nilai max yang sangat besar dibanding atribut V1 hingga V28. Atribut yang memiliki karakteristik berbeda ini dinormalisasi dengan cara normalisasi standar. Namun sebelum itu outlier ekstrim perlu dihilangkan terlebih dahulu.

Tabel 4: Karakteristik atribut dari Credit Card Fraud Dataset setelah normalisasi

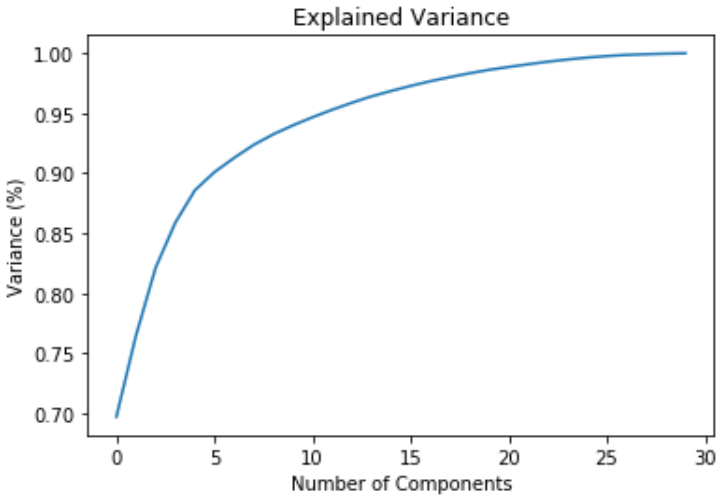
	Time	*V1*	*V2*	...	*V27*	*V28*	*Amount*
mean	0.118965	0	0	...	0	0	0.907330
min	-0.995030	-5.641e+01	-7.271e+01	...	-2.256e+01	-1.543e+01	-0.307865
25%	-0.358151	-9.203e-01	-5.985e-01	...	-7.083e-02	-5.295e-02	-0.229639
50%	0.000000	1.810e-02	6.548e-02	...	1.342e-03	1.124e-02	0.000000
75%	0.641849	1.315e+00	8.037e-01	...	9.104e-02	7.827e-02	0.770361
max	1.035070	2.454e+00	2.205e+01	...	3.161e+01	3.384e+01	69.099496

Tabel 4 menunjukkan bahwa atribut Time dan Amount telah berhasil dinormalisasi mengikuti distribusi sebaran atribut V1 hingga V28.

Seluruh atribut telah diuji menggunakan Saphiro-Wilk test dengan *p-value* kurang dari dari 0.05 (Lampiran 3).

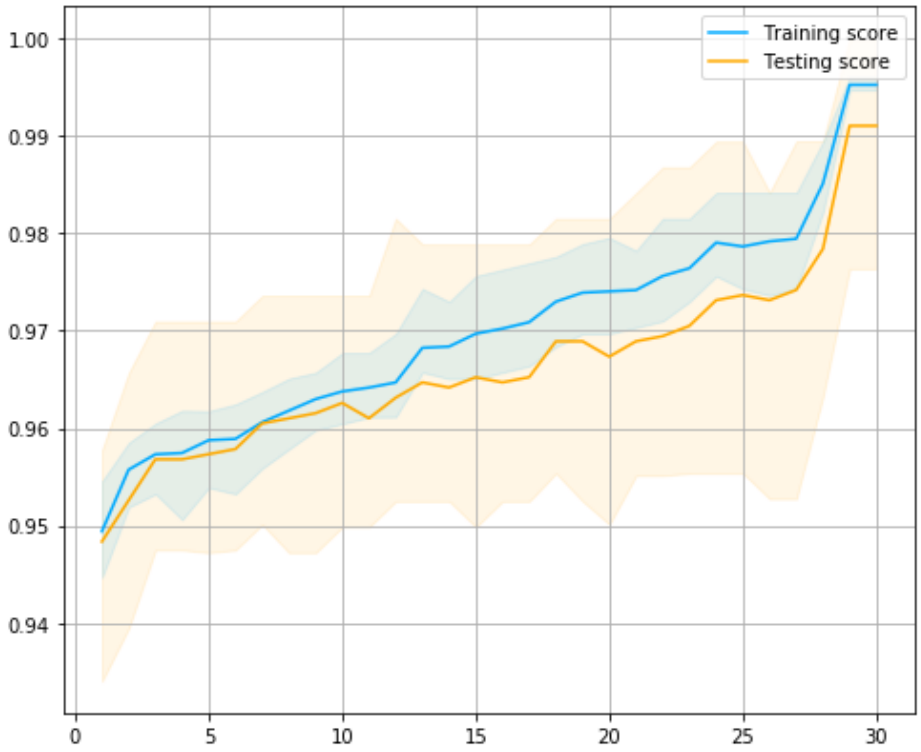
4.1.3.2 *Attribute Reduction*

Jumlah observasi yang cukup besar menyebabkan perlunya *attribute reduction* pada dataset ini. Menggunakan seluruh atribut akan memakan waktu yang sangat lama untuk memproses data besar, terlebih jika memproses data ini dilakukan berkali-kali pada model *tuning* dan model *fitting*.



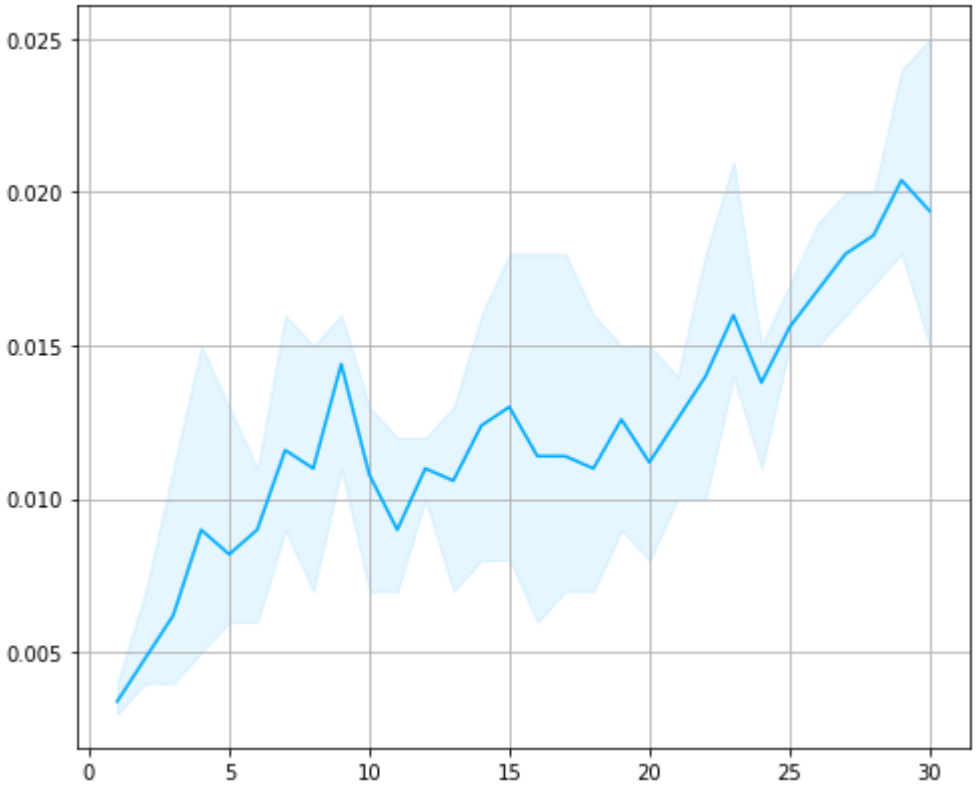
Gambar 13: Scree Plot Credit Card Fraud Dataset

Berdasarkan *scree plot* pada Gambar 13, sebesar 91% informasi dari dataset ini dapat dijelaskan hanya dengan menggunakan 6 komponen utama PCA.



Gambar 14: Training & Testing Score Credit Card Fraud dataset berdasarkan jumlah komponen utama PCA

Gambar 14 menunjukkan bahwa dengan menggunakan seluruh komponen menjamin akurasi training dan testing yang tertinggi. Gambar 14 juga menunjukkan bahwa 2 komponen pertama berkontribusi besar terhadap akurasi model, dengan rata-rata akurasi sebesar 95.5% dengan perbedaan kurang dari 4% dibanding dengan menggunakan seluruh komponen.



Gambar 15: Running time dari satu fit berdasarkan jumlah komponen utama PCA

Running time adalah aspek terpenting pada dataset ini. Gambar 15 menunjukkan bahwa *running time* tumbuh secara linear berdasarkan jumlah komponen. Menggunakan 2 komponen utama PCA memiliki perbedaan *running time* sebesar 3.73 kali dari menggunakan seluruh komponen.

Berdasarkan hasil analisis Gambar 13, Gambar 14 dan Gambar 15, dapat disimpulkan bahwa jumlah komponen teroptimal untuk digunakan adalah 2 komponen. Data mengenai Gambar 13, Gambar 14 dan Gambar 15 dapat dilihat pada Lampiran 8.

4.1.3.3 *Dataset Rebalancing*

Walaupun *attribute reduction* telah dilaksanakan, data masih terlalu besar untuk melakukan *tuning*, *fitting*, dan *cross validation* berkali-kali. Maka sub-dataset perlu dibangun dari dataset asli.

Dataset dipisah berdasarkan kelasnya, yaitu dataset kelas minoritas dan dataset kelas mayoritas. Dataset yang lebih besar, atau dataset kelas mayoritas akan dibagi menjadi 200 kelompok, dengan masing-masing kelompok memiliki size yang sama, yaitu sekitar 1420 sampel per kelompoknya. Dataset baru dibangun berdasarkan dataset kelas minoritas dengan satu kelompok dataset kelas mayoritas, dengan total 200 dataset baru dengan kelas minoritas yang sama di setiap datasetnya namun dengan sampel mayoritas yang berbeda. Setiap dataset akan memiliki *imbalance ratio* sebesar 1:3 (dibanding 1:577 pada dataset asli). Model *tuning* dan *fitting* dilakukan terhadap masing-masing dataset baru ini. Pendekatan ini memiliki keuntungan:

1. Tidak ada information loss.
2. Berupa *cross validation* untuk kelas mayoritas.
3. Mengurangi *overfitting*.

Tabel 5:Perbedaan waktu dari total fitting pada original dataset dan 200 datasets approach

	original dataset	*200 datasets approach*
time(s)	>15000	~6000

Tabel 5 menunjukkan bahwa pendekatan 200 dataset (*200 datasets approach*) dapat menyelesaikan 30 kombinasi pada dataset Credit Card Fraud dalam 6.000 detik, sedangkan tanpa *dataset rebalancing* membutuhkan waktu lebih dari 15.000 detik.

4.2 *Model Tuning & Fitting*

Setelah melakukan data preprocessing, setiap dataset melewati 5-fold *cross validation*, dengan pengecualian Credit Card Fraud Dataset yang harus melewati dataset rebalancing terlebih dahulu. 5-fold *cross validation* memberi distribusi 80% data training dan 20% data testing dengan data testing yang berbeda pada tiap iterasinya. Untuk setiap iterasi *cross validation*, data training diseimbangkan dengan melakukan resampling. Hasil resampling adalah data training yang telah diseimbangkan. Data training yang telah diseimbangkan ini akan diklasifikasikan melalui algoritma *machine learning* dengan berbagai jenis parameter:

1. Regresi Logistik: C dan regularization. C parameter berguna sebagai pengontrol dari regulariasi Ridge dan Lasso Regression. C parameter adalah invers lambda atau $C = \frac{1}{\lambda}$.
   - C: {0.01, 0.1, 0.5, 0.75, 1, 10, 100}
   - Regularization: Lasso Regression dan Ridge Regression
2. Support Vector Machines: C, kernel trick, dan penalty function. Kernel trick adalah fungsi pemetaan yang digunakan untuk mengubah ruang input ke ruang lain dengan tujuan mempermudah pemisahan class dengan *hyperplane*.
   - C: {0.01, 0.1, 0.5, 1, 10, 100}
   - Kernel trick: Radial Basis Function, Polynomial, Sigmoid, Linear
   - Regularization: Lasso Regression dan Ridge Regression
3. MultiLayer Perceptron: fungsi aktivasi, jumlah hidden layer dan jumlah neuron pada hidden layer.
   - Fungsi aktivasi: Logistik, tanh, Rectified Linear Unit (ReLU)
   - Jumlah hidden layer: {1, 2}
   - Jumlah neuron pada hidden layer: $\{|X|, \frac{|X|}{2}\}$ dengan $|X|$ adalah panjang fitur vektor X.
4. Decision Tree: Sampel minimal untuk leaf dan kedalaman maksimal.
   - Sampel minimal untuk leaf: {5, 10, 15}
   - Kedalaman maksimal : {3, 4, 5, 6, 7}

5. KNN: K atau jumlah tetangga terdekat, dan bobot untuk masing-masing tetangga.
   - K: {3, 5, 7}
   - Bobot: Uniform, weighted (Weighted-KNN).

Setiap pengklasifikasi melakukan 3-fold *cross validation* pada data training yang telah diseimbangkan (*balanced dataset*) dengan setiap kombinasi parameter yang ada (Contoh kombinasi parameter untuk MLP: Fungsi aktivasi Logistik, jumlah hidden layer 1, dan jumlah neuron pada hidden layer $\frac{|X|}{2}$). Kombinasi parameter dengan *recall* tertinggi pada 3-fold *cross validation* tersebut dipilih untuk melakukan uji performa pada 20% data testing asli.

Nilai ukur yang digunakan untuk setiap hasil testing dari tidap kombinasi adalah akurasi (secara umum), *precision-0* (akurasi prediksi kelas mayoritas), *precision-1* (akurasi prediksi kelas minoritas), *recall-0* (tingkat pengenalan kelas mayoritas), *recall-1* (tingkat pengenalan kelas minoritas), f1-0 (nilai f1 dari kelas mayoritas), f1-1 (nilai f1 dari kelas minoritas). Namun pada kasus imbalanced class, nilai ukur yang terpenting adalah *recall-1*.

Merata-ratakan hasil dari ketiga dataset, algoritma *machine learning* memilki performa sebagai berikut:

clf	*accuracy*	*precision-0*	*precision-1*	*recall-0*	*recall-1*	*f1-0*	*f1-1*
lr	0.948	0.972	0.902	0.948	0.942	0.960	0.919
svm	0.955	0.975	0.916	0.957	0.946	0.966	0.930
mlp	0.956	0.976	0.921	0.958	0.946	0.966	0.931
dt	0.921	0.947	0.893	0.935	0.870	0.938	0.870
knn	0.949	0.977	0.895	0.944	0.951	0.960	0.921

Tabel 6: Performa algoritma machine learning

Tabel 6 menunjukkan bahwa berdasarkan *recall-1*, seluruh algoritma *machine learning* memiliki performa yang serupa (0.942 – 0.951) dengan pengecualian pada algoritma Decision Tree yang memiliki *recall-1* 0.870.

Dengan merata-ratakan hasil dari ketiga dataset, teknik resampling memiliki performa sebagai berikut:

Res	*accuracy*	*precision-0*	*precision-1*	*recall-0*	*recall-1*	*f1-0*	*f1-1*
Smote	0.952	0.969	0.920	0.958	0.931	0.963	0.923
bsmote	0.931	0.976	0.854	0.919	0.949	0.945	0.895
adasyn	0.928	0.978	0.844	0.914	0.953	0.944	0.891
Ros	0.951	0.969	0.919	0.957	0.929	0.963	0.922
Rus	0.948	0.969	0.904	0.951	0.930	0.960	0.914
Tl	0.955	0.963	0.946	0.969	0.914	0.966	0.927

Tabel 7: Performa teknik resampling

Tabel 7 menunjukkan bahwa berdasarkan *recall-1*, ADASYN adalah teknik resampling yang memiliki *detection rate* tertinggi untuk kelas minoritas dengan nilai 0.953.

Dengan merata-ratakan hasil dari ketiga dataset, berikut adalah 10-tertinggi kombinasi dari algoritma *machine learning* diurut berdasarkan *recall-1* (*detection rate* kelas minoritas):

Res	*clf*	*accuracy*	*precision-0*	*precision-1*	*recall-0*	*recall-1*	*f1-0*	*f1-1*
adasyn	mlp	0.936	0.984	0.852	0.921	0.969	0.951	0.904
adasyn	lr	0.927	0.981	0.819	0.909	0.967	0.943	0.885
adasyn	svm	0.939	0.983	0.855	0.927	0.966	0.953	0.905
adasyn	knn	0.931	0.984	0.840	0.913	0.966	0.946	0.897
bsmote	lr	0.929	0.981	0.824	0.912	0.965	0.945	0.888
bsmote	mlp	0.943	0.982	0.870	0.932	0.964	0.956	0.912
bsmote	svm	0.944	0.982	0.869	0.934	0.963	0.957	0.913
bsmote	knn	0.941	0.982	0.869	0.928	0.963	0.953	0.912
Rus	knn	0.945	0.979	0.875	0.939	0.954	0.958	0.912
smote	knn	0.951	0.977	0.900	0.948	0.951	0.962	0.924

Tabel 8: 10-tertinggi kombinasi algoritma machine learning dan teknik resampling diurut berdasarkan recall-1

Performa dari setiap kombinasi dapat dilihat pada Lampiran 4, Lampiran 5 dan Lampiran 6

4.3 Analisis Hasil

Borderline-SMOTE dan ADASYN memiliki *recall* tertinggi dalam memprediksi kelas minoritas, namun dengan mengorbankan akurasi. Hal ini disebabkan karena Borderline-SMOTE dan ADASYN membuat banyak data sintetik di sekitar kelas mayoritas, memberi batas yang jelas antara kelas minoritas dan mayoritas. Pada ketiga dataset, Borderline-SMOTE dan ADASYN secara konsisten memiliki *recall* tertinggi.

Tomek Links memiliki *recall* terendah dalam memprediksi kelas minoritas jika dibandingkan dengan teknik resampling yang lain. Hal ini disebabkan karena Tomek Links tidak dapat menyeimbangkan jumlah kelas mayoritas dan minoritas. Secara praktis, teknik ini hanya merupakan teknik penghapusan outlier ketika teknik resampling lain telah dilakukan.

KNN memiliki *recall* yang baik dikarenakan jumlah tetangga minoritas akan sangat dominan ketika class telah diseimbangkan dengan teknik resampling. Namun berdampak sangat buruk terhadap presisi dari kelas minoritas, menghasilkan banyak *false positive*.

Performa MLP dan SVM adalah yang terbaik pada algoritma klasifikasi *machine learning*, namun dengan *running time* yang tertinggi pula. Kedua metode ini secara konsisten memiliki *recall* terbaik pada ketiga dataset yang diberikan, menjadikannya pengklasifikasi yang ideal pada masalah ketidakseimangan data.

Decision Tree adalah algoritma *machine learning* yang memiliki performa terburuk pada ketiga dataset ini, hal ini dikarenakan decision tree adalah algoritma klasifikasi yang lebih baik digunakan pada dataset dengan atribut kategorik, sedangkan ketiga dataset yang digunakan seluruhnya memiliki atribut kontinu.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan, peneliti menarik beberapa kesimpulan sebagai berikut:

1. ADASYN dan Borderline-SMOTE adalah teknik resampling terbaik untuk mengenali kelas minoritas, namun dengan menurunkan sedikit akurasi. SMOTE berada di bawah ADASYN dan Borderline-SMOTE pada *recall-1*, namun SMOTE jauh lebih unggul pada *precision-1* dan akurasi pada umumnya, yang menghasilkan nilai f1-1 yang lebih tinggi dari ADASYN dan Borderline-SMOTE. SMOTE juga lebih baik pada *recall-0*, yang ideal jika *false positive* juga sangat tidak diinginkan pada suatu dataset.
2. KNN memiliki *recall* terbaik dibanding seluruh pengklasifikasi lain ketika class telah diseimbangkan dengan teknik resampling, namun MLP dan SVM memiliki akurasi yang lebih tinggi dengan perbedaan *recall* yang sangat tipis dari KNN (2%). MLP dan SVM juga memiliki *precision-1* yang jauh lebih tinggi dari KNN, menghasilkan nilai f1-1 yang lebih tinggi. Secara umum, MLP dan SVM memiliki performa yang lebih baik pada ketiga dataset yang diberikan.
3. Kombinasi algoritma *machine learning* dan teknik resampling yang terbaik adalah adasyn_mlp untuk *recall* tertinggi, smote_knn untuk nilai f1 tertinggi (keseimbangan antara *recall* dan *precision*), dan tl_mlp untuk akurasi pada umumnya.
4. Dataset dengan class yang tingkat ketidakseimbangannya sangat tinggi, seperti Credit Card Fraud Dataset, lebih mudah untuk diproses jika dataset dikelompokkan seperti pada bagian **4.1.3.3 Dataset Rebalancing**. Pendekatan ini menunjukkan jauhnya perbedaan pada *recall-1* jika dibandingkan data diolah langsung sebelum dikelompokkan.

5.2 Saran

Penelitian ini hanya membahas tentang ketidakseimbangan kelas biner, untuk ketidakseimbangan *multiclass*, di mana kelas lebih dari dua, tidak dibahas pada penelitian ini. Teknik undersampling seperti SBC dan NearMiss juga tidak dibahas pada penelitian ini. Maka saran untuk penelitian selanjutnya adalah membahas mengenai ketidakseimbangan kelas di mana kelasnya lebih dari dua (*multiclass*), dan menggunakan teknik-teknik resampling atau algoritma *machine learning* yang belum dibahas pada penelitian ini.

Daftar Pustaka

Ahmed, N. K., Atiya, A. F., Gayar, N. E., & El-Shishiny, H. (2010). An Empirical Comparison of Machine Learning Models for Time Series Forecasting. *Econometric Reviews, 29*(5-6), 594-621.

Amin, A., Anwar, S., Adnan, A., Nawaz, M., Howard, N., Qadir, J., & Hawalah, A. (2016). Comparing Oversampling Techniques to Handle the Class Imbalance Problem: A Customer Churn Prediction Case Study. *IEEE Access, 4*, 7940-7957.

Anand, A., Pugalenthi, G., Fogel, G. B., & Suganthan, P. N. (2010). An approach for classification of highly imbalanced data using weighting and undersampling. *Amino acids*, *39*(5), 1385-1391.

Andrew, N. (2000). CS229 Lecture Notes.

Baars, H., & Kemper, H. G. (2008). Management support with structured and unstructured data—an integrated business intelligence framework. *Information Systems Management*, *25*(2), 132-148.

Batista, G. E., Prati, R. C., & Monard, M. C. (2004). A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *ACM SIGKDD explorations newsletter*, *6*(1), 20-29.

Beal, V. (2019). *Unstructured Data.* Retrieved 6 20, 2019, from https://www.webopedia.com/TERM/U/unstructured_data.html

Berman, F., Rutenbar, R., Hailpern, B., Christensen, H., Davidson, S., Estrin, D., . . . Szalay, A. S. (2018). Realizing the Potential of Data Science. *Communications Of The Acm, 61*(4), 67-72.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Oxford university press.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Springer.

Burnaev, E., Erofeev, P., & Papanov, A. (2015). Influence of Resampling on Accuracy of Imbalanced Classification. *In Eighth International Conference on Machine Vision (ICMV 2015), 9875*, 987521.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of artificial intelligence research, 16*, 321-357.

Chen, M., Mao, S., & & Liu, Y. (2014). Big Data: A survey. *Mobile networks and applications, 19*(2), 171-209.

Dai, Q.-y., Zhang, C.-p., & Wu, H. (2016). Research of Decision Tree Classification Algorithm in Data Mining. *International Journal of Database Theory and Application, 9*(5), 1-8.

Desjardins, J. (2019, April 17). *How much data is generated each day?* World Economic Forum. Retrieved June 14, 2019, from https://www.weforum.org/agenda/2019/04/how-much-data-is-generated-each-day-cf4bddf29f/

Dhar, V. (2012). Data Science and Prediction. *Communications of the ACM, 56*(12), 64-73.

Diri, B., & Albayrak, S. (2008). Visualization and analysis of classifiers performance in multi-class medical data. *Expert Systems with Applications, 34*(1), 628-634.

Domingos, P. (2011). A Few Useful Things to Know about Machine Learning. 78-87.

Elgendy, N., & Elragal, A. (2014). Big data analytics: a literature review paper. *Industrial Conference on Data Mining*, 214-227.

Ethem, A. (2009). *Introduction to Machine Learning.* MIT press.

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., & Herrera, F. (2011). A review on ensembles for the class imbalance problem: bagging-, boosting-,

and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42*(4), 463-484.

Guenther, N., & Schonlau, M. (2016). Support vector machines. *The Stata Journal, 16*(4), 917-937.

Han, H., Wang, W. Y., & Mao, B. H. (2005, August). Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. *International conference on intelligent computing*, 878-887.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation.* Prentice Hall PTR.

He, H., & Garcia, E. A. (2008). Learning from Imbalanced Data. *IEEE Transactions on Knowledge & Data Engineering*(9), 1263-1284.

He, H., Bai, Y., Garcia, E. A., & Li, S. (2008, June). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 1322-1328.

Hechenbichler, S. (2004). Weighted k-Nearest-Neighbor Techniques and Ordinal. *Sonderforschungsbereich 386*.

Hosmer, D. W., & Lemeshow, S. (2000). *Applied Logistic Regression (Second Edition).* Canada: Wiley-Interscience Publication.

Howard, A. (2013). *Elementary Linear Algebra, Binder Ready Version: Applications Version.* John Wiley & Sons.

Jolliffe, I. T., & Cadima, J. (2016). rincipal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*.

Jordan, I. M., & Mitchell, M. T. (2015). Machine learning: Trends, perspectives, and prospects. *Science, 349*(6245), 255-260.

Kotsiantis, S., Kanellopoulos, D., & Pintelas, P. (2006). Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering, 30*(1), 25-36.

Kumar, A., & Sheshadri, H. (2012). On the Classification of Imbalanced Datasets. *International Journal of Computer Applications (0975 – 8887), 44*(8), 1-7.

LinkedIn Economic Graph Team. (2018). *Linkedin 2018 Emerging Jobs Report.* LinkedIn.

Liu, A. Y.-c. (2004). The effect of oversampling and undersampling on classifying imbalanced text datasets. *The University of Texas at Austin*.

Lohr, S. (2012). *The age of big data.* New York: New York Times.

McAfee, A., Brynjolfsson, E., Davenport, T. H., Patil, D. J., & Barton, D. (2012). Big data: the management revolution. *Harvard business review, 90*(10), 60-68.

More, A. (2016). Survey of resampling techniques for improving classification performance in unbalanced datasets. *arXiv preprint arXiv:1608.06048*.

MULTILAYER, A. N. (1998). Artificial Neural Networks (The Multilayer Perceptron) - A Review of Applications in the Atmospheric Sciences. *Atmospheric Environment, 32*(14-15), 2627-2636.

Ng, A., & Katanforoosh, K. (2018). CS 229 Lecture Notes. In A. NG. California: Stanford Edu.

Osisanwo, F., Akinsola, J., Awodele, O., Hinmikaiye, Olakanmi, & Akinjobi. (2017). Supervised Machine Learning Algorithms: Classification and Comparison. *International Journal of Computer Trends and Technology (IJCTT), 48*(3), 128-138.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, M. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research 12*.

Pereira, J. M., Basto, M., & Silva, A. F. (2016). The Logistic Lasso and Ridge Regression in Predicting Corporate Failure. *Procedia Economics and Finance*, 634-641.

Provost, F. (2000, July). Machine Learning from Imbalanced Data Sets 101. *Proceedings of the AAAI'2000 workshop on imbalanced data sets, 68*, 1-3.

Provost, F., & Fawcett, T. (2013). Data Science and its Relationship to Big Data and Data-Driven Decision Making. *Big data, 1*(1), 51-59.

Quinlan, J. (1986). Induction of Decision Trees. *Machine Learning, 1*, 81-106.

Rahman, M. M., & Davis, D. N. (2013). Addressing the Class Imbalance Problem in Medical Datasets. *International Journal of Machine Learning and Computing, 3*(2), 224.

Schuldt, C., Laptev, I., & Caputo, B. (2004). Recognizing human actions: a local SVM approach. *Proceedings of the 17th International Conference on Pattern Recognition, 3*, 32-36.

Snijders, C., Matzat, U., & Reips, U.-D. (2012). 'Big Data'. Big gaps of knowledge in the field of Internet. *International Journal of Internet Science, 7*, 1-5.

Statistics Solution. (2016). *Statistics Solution*. Retrieved August 14, 2019, from https://www.statisticssolutions.com/sample-size-calculation-and-sample-size-justification-resampling/

Swets, J. A. (1988). Measuring the Accuracy of Diagnostic Systems. *Science, 240*(4857), 1285-1293.

Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 267-288.

Tomek, I. (1976). Two Modifications of CNN. *IEEE Trans. Systems, Man and Cybernetics, 6*, 769-772.

Tsangaratos, P., & Ilia, I. (2016). Comparison of a logistic regression and Naïve Bayes classifier in landslide susceptibility assessments: The influence of models complexity and training dataset size. *Catena, 145*, 164-179.

Vafeiadis, T., Diamantaras, K., Sarigiannidis, G., & Chatzisavvas, K. (55). A comparison of machine learning techniques for customer. *Simulation Modelling Practice and Theory, 55*(1), 1-9.

Vapnik, V., & Cortes, C. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273-297.

Visa, S., & Ralescu, A. (2005, April). Issues in Mining Imbalanced Data Sets - A Review Paper. *Proceedings of the sixteen midwest artificial intelligence and cognitive science conference, 2005*, 67-73.

Weglarz, G. (2004). Two Worlds of Data - Unstructured and Structured. *DM Review, 14*, 19-23.

Yen, S. J., & Lee, Y. S. (2006). Under-Sampling Approaches for Improving Prediction of the Minority Class in an Imbalanced Dataset. *Intelligent Control and Automation*, 731-740.

Zhou, Z. H., & Liu, X. Y. (2006). Training Cost-Sensitive Neural Networks with Methods Addressing the Class Imbalance Problem. *IEEE Transactions on Knowledge & Data Engineering*(1), 63-77.

Zurada, J. M. (1992). *Introduction to artificial neural systems.* St. Paul: West publishing company.

LAMPIRAN

Lampiran 1: Shapiro-Wilk Normality Test pada Spambase Dataset

name	p-value	name	p-value
word_freq_make	0.000000e+00	word_freq_money	0.000000e+00
word_freq_address	0.000000e+00	word_freq_hp	0.000000e+00
word_freq_all	0.000000e+00	word_freq_hpl	0.000000e+00
word_freq_3d	0.000000e+00	word_freq_george	0.000000e+00
word_freq_our	0.000000e+00	word_freq_650	0.000000e+00
word_freq_over	0.000000e+00	word_freq_lab	0.000000e+00
word_freq_remove	0.000000e+00	word_freq_labs	0.000000e+00
word_freq_internet	0.000000e+00	word_freq_telnet	0.000000e+00
word_freq_order	0.000000e+00	word_freq_857	0.000000e+00
word_freq_mail	0.000000e+00	word_freq_data	0.000000e+00
word_freq_receive	0.000000e+00	word_freq_415	0.000000e+00
word_freq_will	0.000000e+00	word_freq_85	0.000000e+00
word_freq_people	0.000000e+00	word_freq_technology	0.000000e+00
word_freq_report	0.000000e+00	word_freq_1999	0.000000e+00
word_freq_addresses	0.000000e+00	word_freq_parts	0.000000e+00
word_freq_free	0.000000e+00	word_freq_pm	0.000000e+00
word_freq_business	0.000000e+00	word_freq_direct	0.000000e+00
word_freq_email	0.000000e+00	word_freq_cs	0.000000e+00
word_freq_you	0.000000e+00	word_freq_meeting	0.000000e+00
word_freq_credit	0.000000e+00	word_freq_original	0.000000e+00
word_freq_your	0.000000e+00	word_freq_project	0.000000e+00
word_freq_font	0.000000e+00	word_freq_re	0.000000e+00

word_freq_000	0.000000e+00	word_freq_edu	0.000000e+00
word_freq_conference	0.000000e+00	word_freq_table	0.000000e+00
char_freq_;	0.000000e+00	char_freq_$	0.000000e+00
char_freq_(	0.000000e+00	char_freq_#	0.000000e+00
char_freq_[	0.000000e+00	capital_run_length_average	7.980520e-33
char_freq_!	0.000000e+00	capital_run_length_longest	2.182583e-29
		capital_run_length_total	2.402390e-25

Lampiran 2: Shapiro-Wilk Normality Test pada Image Segmentation Dataset

name	p-value
Vedge-mean	8.299744e-39
Vedge-sd	9.430739e-43
Hedge-mean	1.105662e-37
Hedge-sd	5.324934e-44
Intensity-mean	1.608171e-37
Rawred-mean	3.707636e-37
Rawblue-mean	1.944802e-36
Rawgreen-mean	2.243075e-39
Exred-mean	6.128499e-24
Exblue-mean	1.765567e-22
Exgreen-mean	8.926998e-16
Value-mean	1.945458e-36
Saturation-mean	2.075923e-33
Hue-mean	1.117830e-40

Lampiran 3: Shapiro-Wilk Normality Test pada Credit Card Fraud Dataset

name	p-value	name	p-value
Time	0.000000e+00	V17	0.000000e+00
V1	0.000000e+00	V18	0.000000e+00
V2	0.000000e+00	V19	0.000000e+00
V3	0.000000e+00	V15	0.000000e+00
V4	0.000000e+00	V16	0.000000e+00
V5	0.000000e+00	V20	0.000000e+00
V6	0.000000e+00	V21	0.000000e+00
V7	0.000000e+00	V22	0.000000e+00
V8	0.000000e+00	V23	0.000000e+00
V9	0.000000e+00	V24	0.000000e+00
V10	0.000000e+00	V25	0.000000e+00
V11	0.000000e+00	V26	0.000000e+00
V12	0.000000e+00	V27	0.000000e+00
V13	7.500075e-30	V28	0.000000e+00
V14	0.000000e+00	Amount	0.000000e+00

Lampiran 4: Performa dari seluruh kombinasi pada Spambase Dataset

res	clf	accuracy	precision-0	precision-1	recall-0	recall-1	f1-0	f1-1
smote	lr	0.9129	0.9499	0.8625	0.9045	0.9259	0.9265	0.8929
smote	svm	0.9303	0.9564	0.8932	0.9278	0.9343	0.9419	0.9132
smote	mlp	0.9334	0.9621	0.8931	0.9271	0.9432	0.9442	0.9174
smote	dt	0.8821	0.9234	0.8262	0.8790	0.8869	0.9005	0.8552
smote	knn	0.9138	0.9626	0.8511	0.8930	0.9460	0.9264	0.8959
bsmote	lr	0.8904	0.9616	0.8072	0.8539	0.9471	0.9045	0.8714
bsmote	svm	0.9146	0.9683	0.8471	0.8887	0.9549	0.9267	0.8977
bsmote	mlp	0.9164	0.9655	0.8532	0.8944	0.9504	0.9286	0.8991
bsmote	dt	0.8159	0.9325	0.7077	0.7544	0.9114	0.8323	0.7950
bsmote	knn	0.8952	0.9706	0.8090	0.8535	0.9599	0.9082	0.8779
adasyn	lr	0.8954	0.9591	0.8186	0.8650	0.9426	0.9095	0.8761
adasyn	svm	0.9188	0.9696	0.8540	0.8944	0.9565	0.9305	0.9023
adasyn	mlp	0.9192	0.9700	0.8544	0.8948	0.9571	0.9309	0.9028
adasyn	dt	0.8430	0.9458	0.7386	0.7871	0.9298	0.8589	0.8230
adasyn	knn	0.8932	0.9748	0.8024	0.8463	0.9660	0.9059	0.8765
ros	lr	0.9109	0.9497	0.8585	0.9013	0.9259	0.9248	0.8908
ros	svm	0.9299	0.9574	0.8910	0.9260	0.9359	0.9414	0.9128
ros	mlp	0.9332	0.9610	0.8939	0.9278	0.9415	0.9441	0.9170
ros	dt	0.8784	0.9219	0.8201	0.8747	0.8841	0.8974	0.8505
ros	knn	0.9151	0.9613	0.8551	0.8966	0.9437	0.9277	0.8971
rus	lr	0.9124	0.9485	0.8630	0.9052	0.9237	0.9263	0.8922
rus	svm	0.9279	0.9563	0.8881	0.9239	0.9343	0.9397	0.9104
rus	mlp	0.9295	0.9598	0.8872	0.9228	0.9398	0.9409	0.9127
rus	dt	0.8725	0.9294	0.8022	0.8560	0.8981	0.8907	0.8467
rus	knn	0.9107	0.9657	0.8422	0.8847	0.9510	0.9233	0.8931
tl	lr	0.9133	0.9386	0.8766	0.9174	0.9070	0.9279	0.8914
tl	svm	0.9332	0.9491	0.9095	0.9408	0.9214	0.9448	0.9153
tl	mlp	0.9330	0.9501	0.9078	0.9393	0.9231	0.9446	0.9152
tl	dt	0.8810	0.9081	0.8446	0.8966	0.8568	0.9016	0.8490
tl	knn	0.9210	0.9568	0.8722	0.9113	0.9359	0.9334	0.9028

Lampiran 5: Performa dari seluruh kombinasi pada Image Segmentation Dataset

res	clf	accuracy	precision-0	precision-1	recall-0	recall-1	f1-0	f1-1
smote	lr	0.9953	0.9964	0.9908	0.9978	0.9847	0.9971	0.9877
smote	svm	0.9965	0.9964	0.9969	0.9993	0.9847	0.9978	0.9907
smote	mlp	0.9953	0.9978	0.9851	0.9964	0.9908	0.9971	0.9878
smote	dt	0.9854	0.9971	0.9392	0.9849	0.9878	0.9909	0.9628
smote	knn	0.9907	0.9985	0.9587	0.9899	0.9939	0.9942	0.9760
bsmote	lr	0.9673	0.9978	0.8618	0.9618	0.9908	0.9794	0.9209
bsmote	svm	0.9883	0.9985	0.9480	0.9870	0.9938	0.9927	0.9703
bsmote	mlp	0.9942	0.9986	0.9767	0.9942	0.9939	0.9964	0.9850
bsmote	dt	0.9872	0.9993	0.9395	0.9849	0.9970	0.9920	0.9674
bsmote	knn	0.9901	0.9985	0.9562	0.9892	0.9939	0.9938	0.9746
adasyn	lr	0.9702	0.9985	0.8713	0.9647	0.9938	0.9813	0.9278
adasyn	svm	0.9883	0.9985	0.9480	0.9870	0.9938	0.9927	0.9703
adasyn	mlp	0.9942	0.9986	0.9766	0.9942	0.9939	0.9964	0.9850
adasyn	dt	0.9866	0.9985	0.9393	0.9849	0.9939	0.9916	0.9658
adasyn	knn	0.9883	0.9985	0.9477	0.9870	0.9939	0.9927	0.9702
ros	lr	0.9936	0.9964	0.9823	0.9957	0.9847	0.9960	0.9832
ros	svm	0.9953	0.9964	0.9908	0.9978	0.9847	0.9971	0.9877
ros	mlp	0.9942	0.9978	0.9796	0.9949	0.9908	0.9964	0.9849
ros	dt	0.9837	0.9956	0.9361	0.9841	0.9817	0.9898	0.9582
ros	knn	0.9901	0.9978	0.9586	0.9899	0.9908	0.9938	0.9744
rus	lr	0.9907	0.9971	0.9647	0.9913	0.9878	0.9942	0.9760
rus	svm	0.9947	0.9971	0.9848	0.9964	0.9878	0.9968	0.9863
rus	mlp	0.9901	0.9971	0.9617	0.9906	0.9878	0.9938	0.9744
rus	dt	0.9697	0.9934	0.8825	0.9690	0.9725	0.9810	0.9247
rus	knn	0.9761	1.0000	0.8898	0.9704	1.0000	0.9850	0.9414
tl	lr	0.9971	0.9971	0.9970	0.9993	0.9878	0.9982	0.9923
tl	svm	0.9971	0.9964	1.0000	1.0000	0.9847	0.9982	0.9923
tl	mlp	0.9977	0.9971	1.0000	1.0000	0.9878	0.9986	0.9938
tl	dt	0.9860	0.9914	0.9639	0.9913	0.9634	0.9913	0.9632
tl	knn	0.9936	0.9971	0.9789	0.9950	0.9877	0.9960	0.9833

Lampiran 6: Performa dari seluruh kombinasi pada Credit Card Fraud Dataset

res	clf	accuracy	precision-0	precision-1	recall-0	recall-1	f1-0	f1-1
smote	lr	0.9589	0.9692	0.9283	0.9763	0.9071	0.9727	0.9172
smote	svm	0.9530	0.9717	0.9004	0.9655	0.9156	0.9685	0.9072
smote	mlp	0.9579	0.9689	0.9257	0.9753	0.9062	0.9720	0.9154
smote	dt	0.9234	0.9208	0.9581	0.9849	0.7399	0.9507	0.8242
smote	knn	0.9494	0.9709	0.8897	0.9614	0.9135	0.9660	0.9007
bsmote	lr	0.9284	0.9844	0.8028	0.9189	0.9567	0.9503	0.8718
bsmote	svm	0.9296	0.9794	0.8125	0.9256	0.9417	0.9515	0.8713
bsmote	mlp	0.9171	0.9810	0.7786	0.9070	0.9475	0.9422	0.8533
bsmote	dt	0.8862	0.9271	0.8742	0.9260	0.7672	0.9155	0.7887
bsmote	knn	0.9387	0.9772	0.8422	0.9402	0.9342	0.9582	0.8850
adasyn	lr	0.9140	0.9867	0.7665	0.8972	0.9641	0.9393	0.8518
adasyn	svm	0.9104	0.9806	0.7639	0.8981	0.9471	0.9370	0.8438
adasyn	mlp	0.8946	0.9836	0.7258	0.8739	0.9565	0.9248	0.8230
adasyn	dt	0.8931	0.9259	0.8865	0.9369	0.7625	0.9228	0.7944
adasyn	knn	0.9129	0.9777	0.7702	0.9044	0.9383	0.9394	0.8449
ros	lr	0.9598	0.9681	0.9351	0.9787	0.9034	0.9733	0.9185
ros	svm	0.9554	0.9699	0.9138	0.9707	0.9098	0.9702	0.9110
ros	mlp	0.9589	0.9682	0.9316	0.9774	0.9038	0.9727	0.9170
ros	dt	0.9240	0.9209	0.9595	0.9854	0.7406	0.9511	0.8255
ros	knn	0.9483	0.9707	0.8862	0.9600	0.9132	0.9653	0.8987
rus	lr	0.9594	0.9677	0.9347	0.9786	0.9022	0.9730	0.9177
rus	svm	0.9455	0.9718	0.8742	0.9551	0.9168	0.9633	0.8943
rus	mlp	0.9600	0.9648	0.9464	0.9825	0.8927	0.9735	0.9181
rus	dt	0.9237	0.9230	0.9510	0.9824	0.7483	0.9507	0.8268
rus	knn	0.9496	0.9700	0.8927	0.9627	0.9106	0.9662	0.9009
tl	lr	0.9629	0.9606	0.9712	0.9911	0.8786	0.9756	0.9222
tl	svm	0.9600	0.9637	0.9489	0.9837	0.8890	0.9736	0.9175
tl	mlp	0.9632	0.9597	0.9757	0.9926	0.8754	0.9758	0.9225
tl	dt	0.9254	0.9161	0.9784	0.9932	0.7231	0.9526	0.8238
tl	knn	0.9625	0.9633	0.9605	0.9876	0.8873	0.9753	0.9220

Lampiran 7: Performa model pada komponen-komponen utama PCA dari Image Segmentation dataset

n_components	variance	training accuracy	testing accuracy	running_time
1	0.323265	0.777183	0.777948	0.012590
2	0.441720	0.780186	0.782096	0.011792
3	0.516322	0.832697	0.832314	0.011193
4	0.563684	0.871725	0.872926	0.012992
5	0.605417	0.873472	0.874454	0.015395
6	0.644951	0.872871	0.870087	0.016589
7	0.674542	0.875382	0.872489	0.018989
8	0.702507	0.877511	0.872707	0.020189
9	0.727592	0.896070	0.891921	0.037179
10	0.751075	0.911354	0.908952	0.031720
11	0.773298	0.912937	0.912009	0.031580
12	0.793020	0.914847	0.913755	0.027785
13	0.811804	0.915502	0.913100	0.030981
14	0.827974	0.917522	0.914410	0.029182
15	0.842466	0.918668	0.917031	0.046370
16	0.855382	0.917959	0.915066	0.079464
17	0.867154	0.917522	0.915721	0.053370
18	0.877956	0.917194	0.914847	0.066362
19	0.887448	0.917413	0.915939	0.051771
20	0.896320	0.917303	0.914847	0.048370
21	0.904811	0.919323	0.917249	0.054766
22	0.911926	0.920360	0.918341	0.056169
23	0.918549	0.919978	0.917904	0.056966

24	0.924953	0.920360	0.918777	0.058164
25	0.930908	0.922162	0.919432	0.073358
26	0.936661	0.922707	0.920961	0.071760
27	0.942210	0.922653	0.920961	0.076558
28	0.947246	0.922707	0.920087	0.074358
29	0.951330	0.922707	0.919651	0.082952
30	0.955232	0.923199	0.919651	0.084152
31	0.958873	0.925655	0.924891	0.097742
32	0.962298	0.926474	0.923362	0.112439
33	0.965654	0.927566	0.923144	0.118132
34	0.968938	0.928493	0.925546	0.129240
35	0.971986	0.929367	0.924891	0.128927
36	0.974909	0.929640	0.925546	0.112369
37	0.977580	0.929476	0.926638	0.122774
38	0.979902	0.930186	0.927511	0.118546
39	0.982210	0.930950	0.927729	0.121531
40	0.984366	0.930950	0.927511	0.128527
41	0.986214	0.931004	0.926856	0.135323
42	0.987758	0.930895	0.927511	0.145451
43	0.989240	0.930950	0.927729	0.144523
44	0.990653	0.931878	0.927293	0.148472
45	0.991921	0.931550	0.927948	0.143946
46	0.993054	0.933133	0.926638	0.143134
47	0.994155	0.934498	0.928821	0.149327
48	0.995234	0.934716	0.928821	0.154321

49	0.996197	0.934716	0.928384	0.160680
50	0.997104	0.935480	0.928603	0.158643
51	0.997926	0.935044	0.929913	0.169452
52	0.998618	0.935535	0.930131	0.176796
53	0.999261	0.935644	0.929913	0.174957
54	0.999631	0.935862	0.929913	0.184309
55	0.999877	0.937445	0.930786	0.194573
56	0.999994	0.937500	0.931004	0.190899
57	1.000000	0.937500	0.930786	0.207372

Lampiran 8: Performa model pada komponen-komponen utama PCA dari Credit Card Fraud dataset

n_components	variance	training accuracy	testing accuracy	running_time
1	0.695250	0.949421	0.948365	0.003596
2	0.758855	0.955743	0.952594	0.005196
3	0.815645	0.957324	0.956809	0.008595
4	0.854299	0.957456	0.956807	0.006995
5	0.883382	0.958773	0.957335	0.005995
6	0.898010	0.958905	0.957860	0.012792
7	0.910138	0.960617	0.960496	0.008793
8	0.920838	0.961802	0.961016	0.008194
9	0.928967	0.963119	0.961541	0.008192
10	0.936481	0.963909	0.962597	0.011993
11	0.943320	0.964173	0.961019	0.008594
12	0.949931	0.964700	0.963133	0.010992
13	0.955761	0.968256	0.964706	0.009194
14	0.961171	0.968388	0.964181	0.009994
15	0.966261	0.969705	0.965232	0.009200
16	0.970950	0.970232	0.964710	0.011592
17	0.975276	0.970891	0.965237	0.012401
18	0.979410	0.972998	0.968925	0.009193
19	0.982674	0.973921	0.968926	0.012791
20	0.985642	0.974053	0.967347	0.012591
21	0.988301	0.974184	0.968926	0.014590
22	0.990686	0.975633	0.969454	0.015989
23	0.992820	0.976424	0.970508	0.013193

24	0.994726	0.979058	0.973144	0.013991
25	0.996245	0.978663	0.973672	0.015192
26	0.997576	0.979189	0.973145	0.016390
27	0.998600	0.979453	0.974198	0.016785
28	0.999215	0.985117	0.978408	0.019390
29	0.999698	0.995258	0.991055	0.019388