BAB II
TINJAUAN PUSTAKA

2.1. Penelitian Sebelumnya

Pada sub bab ini, dipaparkan mengenai penelitian-penelitian yang berkaitan dengan topik penelitian. Penelitian-penelitian tersebut diantaranya berkaitan dengan metode *Extreme Learning Machine* untuk klasifikasi serta faktor-faktor yang mempengaruhi terjadinya Diabetes Melitus.

Pada penelitian tentang diabetes yang dilakukan oleh (Djihanga, 2020) dalam makalah yang berjudul "Screening Diabetes Mellitus Gestasional di Negeri Berkembang", telah dilakukan analisis skrining faktor-faktor penyebab diabetes mellitus gestasional di negara-negara berkembang. Disimpulkan bahwa faktor risiko yang mempengaruhi diabetes melitus gestasional adalah, usia, riwayat diabetes melitus pada keluarga, *Body Mass Index*, dan kadar gula darah pada ibu hamil. Pada penelitian lain tentang faktor-faktor yang mempengaruhi terjadinya diabetes yang dilakukan oleh (Fanani & Sulaiman, 2021) dalam makalah yang berjudul "Faktor Obesitas dan Faktor Keturunan dengan Kejadian Kasus Diabetes Mellitus", telah dilakukan evaluasi beberapa faktor penyebab diabetes mellitus yang dititik beratkan pada obesitas dan genetik atau keturunan. Disimpulkan bahwa faktor risiko seperti obesitas dan genetik atau keturunan memiliki hubungan yang signifikan dengan kejadian Diabetes Mellitus.

Penelitian lain yang membahas tentang perbandingan performa *ELM* dengan metode lain yang dilakukan oleh (Ahmad et al., 2018) dalam makalah yang berjudul "*Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection*", telah dilakukan proses pembandingan pada teknik pembelajaran mesin yang terkenal yaitu *SVM, Random Forest,* dan *Extreme Learning Machine*, untuk mendeteksi terjadinya penyusupan atau intrusi pada sistem dan jaringan. Disimpulkan bahwa *ELM* mengungguli pendekatan lain dalam akurasi, presisi, dan penarikan kembali pada sampel data lengkap yang terdiri dari 65.535 catatan aktivitas yang berisi aktivitas normal dan

intrusive, oleh karena itu, *ELM* adalah teknik yang cocok untuk sistem deteksi intrusi yang dirancang untuk menganalisis sejumlah besar data.

Pada penelitian lain yang membahas tentang metode *ELM* untuk mendiagnosis penyakit tumor otak, yang dilakukan oleh (Rakhman Wahid et al., 2021) dalam makalah yang berjudul "*Brain Tumor Classification with Hybrid Algorithm Convolutional Neural Network-Extreme Learning Machine*", telah dilakukan implementasi *Extreme Learning Machine* untuk melakukan klasifikasi tumor otak. Disimpulkan bahwa *ELM* yang menggunakan lebih banyak node pada lapisan tersembunyi lebih cenderung mendapatkan hasil akurasi yang lebih baik. Pada penelitian ini didapatkan rata-rata presisi, skor F1, dan recall sebesar 0,915 dan akurasi 91,4% pada model *CNN-ELM* dengan 8 filter dan 6000 node di lapisan tersembunyi. Berbagai teknik pra-proses citra juga akan mempengaruhi hasil akhir yang didapatkan.

Pada penelitian lain tentang penggunaan metode *ELM* untuk mendiagnosis penyakit Diabetes Melitus yang dilakukan oleh (Pangaribuan & Suharjito, 2014) dalam makalah yang berjudul "*Diagnosis of Diabetes Mellitus Using Extreme Learning Machine*", telah dilakukan implementasi *Extreme Learning Machine* untuk mendeteksi Diabetes. Disimpulkan bahwa *ELM* memiliki akurasi yang tinggi dan kecepatan yang sangat baik dalam mendiagnosis Diabetes Mellitus. Dari segi kecepatan, kinerja *ELM* lebih baik daripada kinerja *backpropagation*, dimana kecepatan diperoleh saat proses prediksi dimulai per setiap data. Performa *ELM* dalam hal akurasi juga lebih baik dibandingkan performa *backpropagation*, dimana akurasi dapat dilihat dari nilai output yang telah dianalisa dengan melihat *error rate* dari *MSE*.

Pada penelitian yang lain tentang *ELM* yang digunakan untuk mendiagnosis Diabetes yang dilakukan oleh (Shanthi et al., 2019) dalam makalah yang berjudul "*Diagnosis of Diabetes using an Extreme Learning Machine Algorithm based Model*", telah dilakukan studi tentang tipe-tipe diabetes serta dilakukan pengembangan suatu model untuk mendiagnosis Diabetes tipe 2 menggunakan *Extreme Learning Machine*. Disimpulkan bahwa untuk jumlah kehamilan yang lebih sedikit, persentase kasus diabetes tinggi jika usia kurang dari 40 tahun dan

persentasenya terus menurun seiring dengan meningkatnya jumlah kehamilan. Ketika analisis yang sama dilakukan untuk kasus di atas 45 tahun, ditemukan bahwa persentasenya tetap baik dalam kisaran yang sempit. Model yang diusulkan diuji dengan kumpulan data yang sama dan persentase prediksi yang benar yang dibuat oleh model adalah 84%. Hasilnya, model berbasis algoritma ELM ini dapat membantu para ahli medis untuk mendeteksi pasien dengan Diabetes tipe II yang mencurigakan.

2.2. Diabetes

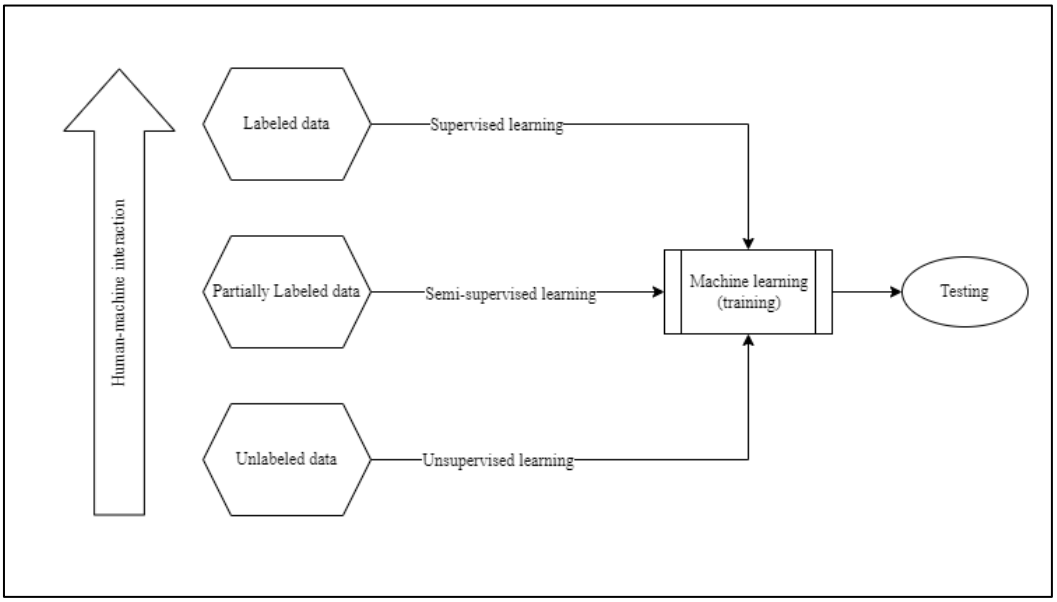
Diabetes Melitus merupakan penyakit yang tidak menular yang disebabkan oleh kerusakan pankreas atau berkurangnya insulin yang diproduksi oleh pankreas sehingga terjadi peningkatan kadar gula didalam darah atau resistensi insulin yang menjadi masalah kesehatan terbesar dunia saat ini, dan menjadi salah satu faktor penyebab turunnya kualitas sumber daya manusia (Isnaini & Ratnasari, 2018). Penderita Diabetes Melitus seringkali tidak menyadari bahwa dirinya telah menderita Diabetes Melitus dalam waktu yang lama, hal ini sangat berbahaya, dan ketika disadari telah terjadi komplikasi. Komplikasi yang menyebabkan Diabetes Melitus sering disebut sebagai silent killer (Renagalih Amarta et al., 2021).

Oleh karena itu perlu diketahui apa saja yang dapat menjadi faktor risiko timbulnya penyakit Diabetes Melitus. Jumlah penderita Diabetes Melitus di Indonesia dapat ditekan, dan komplikasi penyakit Diabetes Melitus dapat dicegah sedini mungkin (Renagalih Amarta et al., 2021). Beberapa penelitian menunjukkan bahwa faktor-faktor yang memiliki pengaruh besar untuk mendeteksi Diabetes adalah konsentrasi glukosa plasma, umur, tekanan darah diastolic, indeks massa tubuh, kehamilan, ketebalan lipatan kulit trisep, fungsi silsilah diabetes, dan insulin serum (el Jerjawi Nesreen Samer & Abu-Naser, 2018).

2.3. *Machine Learning*

Machine Learning (ML) adalah bagian dari kecerdasan buatan, yang membangun model matematika berdasarkan data sampel, yang dikenal sebagai "data pelatihan", untuk membuat prediksi atau keputusan tanpa diprogram secara eksplisit untuk melakukan tugas tersebut (X.-D. Zhang, 2020). Jenis-jenis

permasalahan yang umumnya diselesaikan dengan pendekatan Machine learning adalah klasterisasi dan klasifikasi. Klasterisasi adalah aktivitas yang bertujuan mengelompokan data berdasarkan kedekatan fitur yang dimilikinya, sedangkan klasifikasi bertujuan untuk memisahkan data menjadi kelas-kelas tertentu. Perbedaan yang mendasar antara 2 buah permasalahan ini adalah, pada proses klasterisasi, data-data dikelompokan tanpa pelabelan, sedangkan klasifikasi mengelompokan data-data menjadi label tertentu (Putu, 2018).



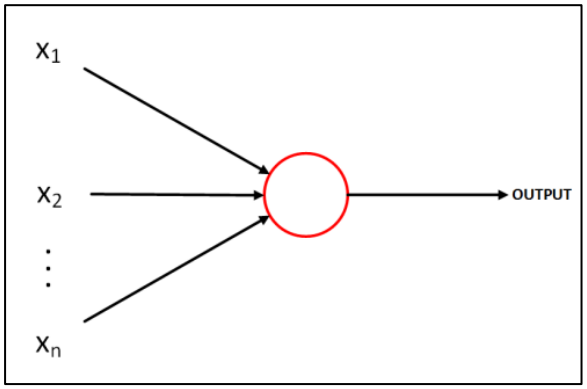
Gambar 2.1 Kategori Algoritma Pembelajaran Mesin Menurut Data Pelatihan **(el Naqa & Murphy, 2015)**

Pembelajaran mesin dapat dibagi menurut sifat pelabelan data menjadi supervised, unsupervised, dan semi-supervised seperti yang ditunjukkan pada Gambar 2.1. Pembelajaran terawasi digunakan untuk memperkirakan pemetaan yang tidak diketahui (input, output) dari sampel yang diketahui (input, output), di mana output diberi label (misalnya, klasifikasi dan regresi). Dalam pembelajaran tanpa pengawasan, hanya sampel masukan yang diberikan ke sistem pembelajaran (misalnya, pengelompokan dan estimasi fungsi kepadatan probabilitas). Pembelajaran semi-diawasi adalah kombinasi dari keduanya terawasi dan tidak terawasi dimana sebagian data diberi label sebagian dan bagian berlabel digunakan

untuk menyimpulkan bagian yang tidak berlabel (misalnya, sistem pencarian teks/gambar) (el Naqa & Murphy, 2015).

2.4. Jaringan Syaraf Tiruan

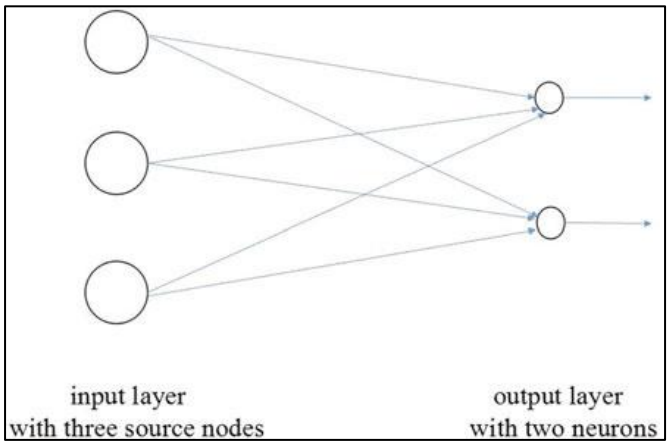
Jaringan saraf tiruan (JST) adalah alat kecerdasan buatan yang kuat yang mampu memecahkan masalah sulit dengan cara yang menyerupai kecerdasan manusia. JST tidak memerlukan pengetahuan sebelumnya tentang hubungan antara data yang terlibat dalam masalah sebelum menyelesaikannya, tidak seperti banyak pendekatan statistik dan probabilistik. Seperti model regresi nonlinier, JST dapat memecahkan masalah regresi nonlinier dengan akurasi yang sangat tinggi karena strukturnya mirip dengan otak manusia (Lawal & Idris, 2020).



Gambar 2.2 Percepton (Satria Wibawa, 2017)

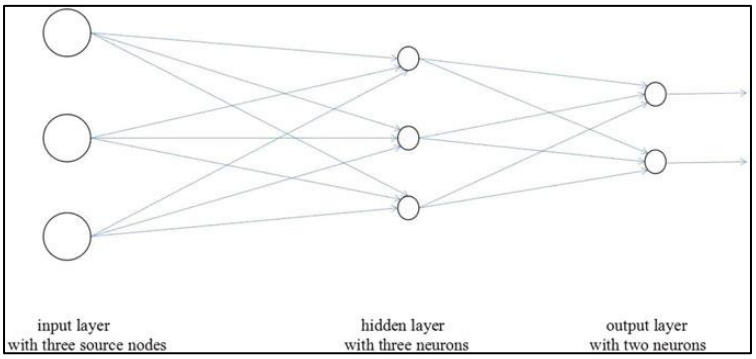
Sama seperti manusia, jaringan saraf tiruan terdiri dari beberapa neuron dan ada hubungan antara neuron-neuron tersebut. Beberapa neuron akan mentransformasikan informasi yang diterimanya melalui sambungan keluaran menuju neuron-neuron yang lain. Dengan kata lain, neuron adalah sebuah unit pemroses informasi yang merupakan dasar operasi jaringan saraf tiruan. Neuron ini dimodelkan dari penyederhanaan sel saraf manusia yang sebenarnya. Gambar 2.2 merupakan contoh suatu neuron. Dengan neuron yang sederhana tersebut, jaringan saraf tiruan dapat dibentuk menjadi beberapa arsitektur seperti *perceptron*, *feed-forward network*, *backpropagation*, *adaline* dan *madaline*, dsb (Satria Wibawa, 2017).

Jaringan syaraf tiruan tertua dan paling sederhana adalah jaringan syaraf tiruan *feedforward* satu lapis. Ini terdiri dari lapisan input node sumber dan lapisan *output* neuron, dimana node sumber diproyeksikan langsung ke lapisan *output* neuron. Kata "lapisan tunggal" berarti bahwa jaringan saraf hanya memiliki satu lapisan. Lapisan node sumber tidak dihitung karena tidak ada perhitungan yang dilakukan (Z. Zhang, 2018). Gambar 2.3 merupakan jaringan syaraf tiruan *feedforward* satu lapis.



Gambar 2.3 Jaringan Syaraf Tiruan Satu Lapis (Z. Zhang, 2018)

Sedangkan sebuah jaringan *feedforward multilayer* terdiri atas lapisan masukan dan lapisan luaran serta lapisan terselubung (*hidden layer*). Namun jumlah lapisan terselubung pada *MLP* bisa bermacam-macam. Jumlah lapisan masukan sama dengan jumlah variabel bebas dalam model. Sedangkan jumlah neuron pada variabel luarannya sama dengan variabel terikat pada model. Gambar 2.4 menunjukkan model *Multi Layer Perceptron* (*MLP*) (Setialaksana et al., 2020).



Gambar 2.4 Jaringan Syaraf Tiruan *Multilayer* (Z. Zhang, 2018)

Secara umum, jaringan syaraf tiruan dapat dibagi menjadi tiga bagian, yang diberi nama lapisan, yang dikenal sebagai (da Silva et al., 2017):

1. *Input Layer*

   Lapisan ini bertanggung jawab untuk menerima informasi (data), sinyal, fitur, atau pengukuran dari lingkungan eksternal. *Input* ini (sampel atau pola) biasanya dinormalisasi dalam nilai batas yang dihasilkan oleh fungsi aktivasi. Normalisasi ini menghasilkan presisi numerik yang lebih baik untuk operasi matematika yang dilakukan oleh jaringan.

2. *Hidden Layer*

   Lapisan ini terdiri dari neuron yang bertanggung jawab untuk mengekstraksi pola yang terkait dengan proses atau sistem yang dianalisis. Lapisan ini melakukan sebagian besar pemrosesan internal dari jaringan.

3. *Output Layer*

   Lapisan ini juga terdiri dari neuron, dan dengan demikian bertanggung jawab untuk memproduksi dan menampilkan keluaran akhir jaringan, yang dihasilkan dari pemrosesan yang dilakukan oleh neuron pada lapisan sebelumnya.

$$Y_j = f\left(\theta_j + \sum_{i=1}^{n} w_{ji} X_i\right) \tag{2.1}$$

Proses JST dapat diringkas dalam persamaan sederhana seperti yang disajikan dalam persamaan 2.1. Dimana θ_j adalah bias pada lapisan tersembunyi, n adalah jumlah neuron di lapisan tersembunyi, w_{ji} bobot koneksi antara variabel input dan lapisan tersembunyi, X_i adalah variabel masukan, Y_j adalah variabel keluaran, dan f adalah fungsi aktivasi (Lawal & Idris, 2020).

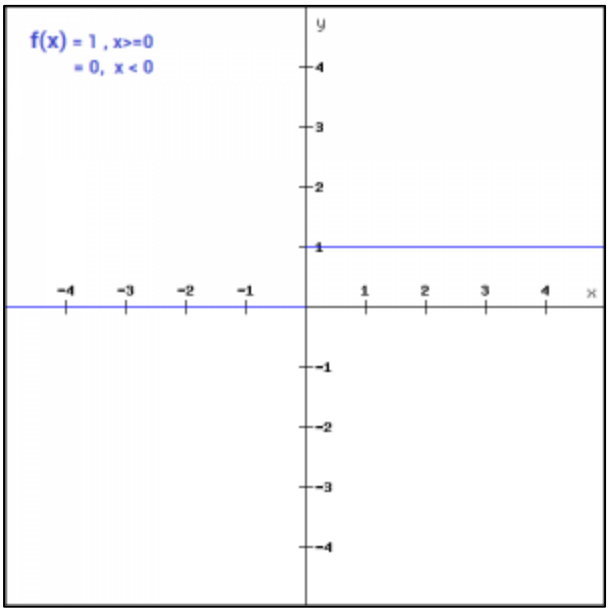
2.5. Fungsi Aktivasi

Fungsi aktivasi merupakan fungsi yang digunakan pada jaringan saraf untuk mengaktifkan atau tidak mengaktifkan neuron. Karakteristik yang harus dimiliki oleh fungsi aktivasi jaringan perambatan balik antara lain harus kontinyu, terdiferensialkan, dan tidak menurun secara monotonis (*monotonically non-decreasing*). Lebih lanjut, untuk efisiensi komputasi, turunan fungsi tersebut mudah didapatkan dan nilai turunannya dapat dinyatakan dengan fungsi aktivasi itu sendiri (Budhiarti Nababan & Zarlis, 2015). Berikut ini merupakan beberapa fungsi aktivasi yang sering digunakan (Sharma et al., 2020):

1. Fungsi Undak Biner

   Fungsi Undak Biner adalah fungsi aktivasi paling sederhana yang ada dan dapat diimplementasikan dengan pernyataan *if-else* sederhana dengan Python. Pada pengklasifikasi biner, umumnya fungsi undak biner digunakan. Gambar 2.5 merupakan grafik fungsi Undak Biner. Fungsi undak biner dirumuskan pada persamaan 2.2 berikut:

$$f(x) = \begin{cases} 1\ jika\ x \geq 0 \\ 0\ jika\ x < 0 \end{cases} \quad (2.2)$$

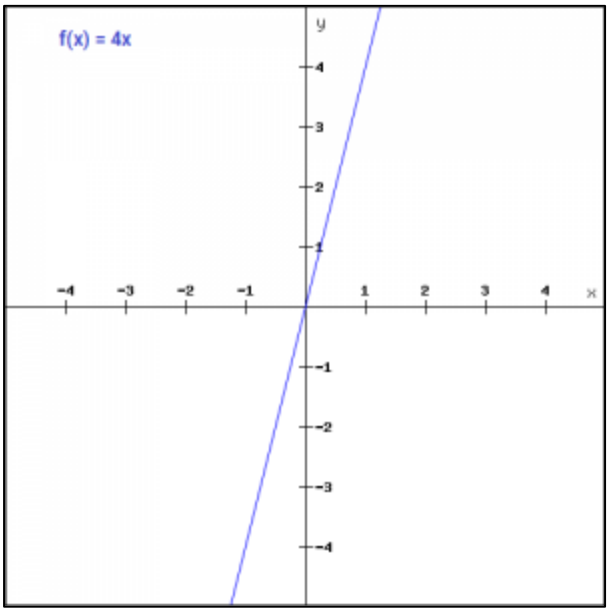


Gambar 2.5 Fungsi Undak Biner

2. Fungsi Linear

Fungsi aktivasi linear berbanding lurus dengan input. Kelemahan utama dari fungsi undak biner adalah memiliki gradien nol, karena tidak ada komponen x dalam fungsi undak biner. Nilai variabel dapat berupa nilai konstan yang dipilih oleh pengguna. Untuk menghindarinya, dapat digunakan fungsi linier. Gambar 2.6 merupakan grafik fungsi Linear. Fungsi linear dirumuskan pada persamaan 2.3 berikut:

$$f(x) = ax \tag{2.3}$$



Gambar 2.6 Fungsi Linear

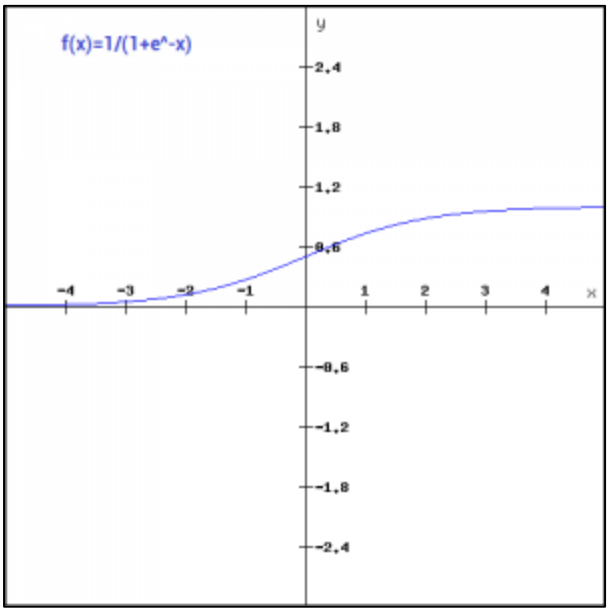
3. Fungsi Sigmoid

   Ini adalah fungsi aktivasi yang paling banyak digunakan karena merupakan fungsi non-linier. Fungsi sigmoid mengubah nilai dalam rentang 0 hingga 1. Fungsi sigmoid dirumuskan pada persamaan 2.4 berikut:

$$f(x) = \frac{1}{e^{-x}} \tag{2.4}$$

   Fungsi sigmoid terdiferensialkan secara terus menerus dan fungsi berbentuk seperti huruf S. Gambar 2.7 merupakan grafik fungsi Sigmoid. Turunan dari fungsi tersebut bisa dilihat pada persamaan 2.5 berikut:

$$f'(x) = 1 - sigmoid(x) \tag{2.5}$$

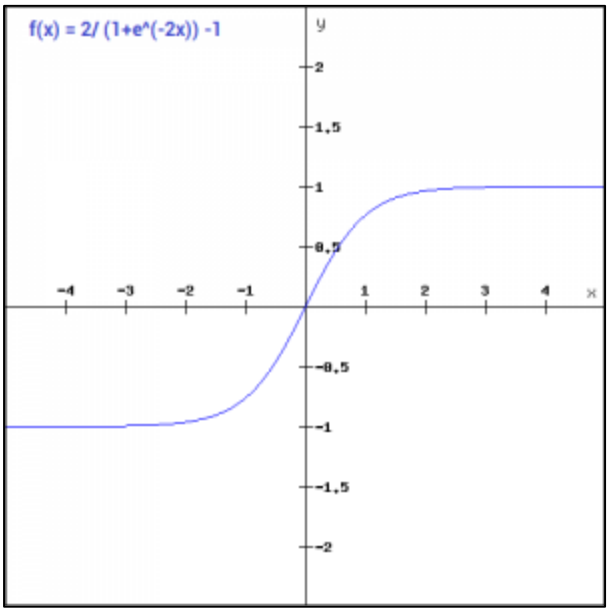


Gambar 2.7 Fungsi Sigmoid

4. Fungsi Tanh

Ini adalah fungsi Tangen Hiperbolik. Fungsi Tanh mirip dengan fungsi sigmoid tetapi simetris dengan sekitar titik asal. Ini menghasilkan tanda keluaran yang berbeda dari lapisan sebelumnya yang akan diumpankan sebagai input ke lapisan berikutnya. Gambar 2.8 merupakan grafik fungsi Sigmoid. Fungsi tanh dirumuskan pada persamaan 2.6 berikut:

$$f(x) = 2sigmoid(2x) - 1 \qquad (2.6)$$

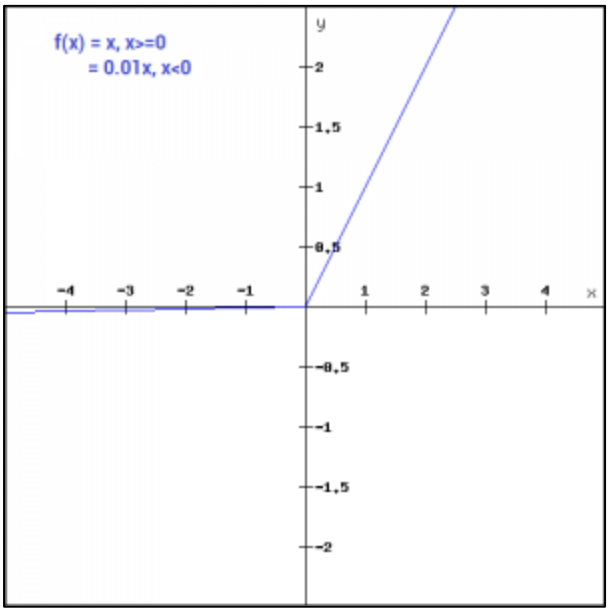


Gambar 2.8 Fungsi Tanh

5. Fungsi ReLU

ReLU adalah singkatan dari rectified linear unit dan merupakan fungsi aktivasi non-linier yang banyak digunakan dalam jaringan saraf. Keunggulan menggunakan fungsi ReLU adalah bahwa semua neuron tidak diaktifkan secara bersamaan. Ini berarti bahwa neuron akan dinonaktifkan hanya ketika output dari transformasi linier adalah nol. Gambar 2.9 merupakan grafik fungsi ReLU. Fungsi ReLU dirumuskan pada persamaan 2.7 berikut:

$$f(x) = \max(0, x) \tag{2.7}$$



Gambar 2.9 Fungsi ReLU

2.6. Normalisasi

Normalisasi data adalah langkah pra-pemrosesan penting yang melibatkan transformasi fitur dalam rentang yang sama sehingga nilai fitur numerik yang lebih besar tidak dapat mendominasi nilai fitur numerik yang lebih kecil. Tujuan utamanya adalah untuk meminimalkan bias fitur-fitur yang kontribusi numeriknya lebih tinggi dalam kelas pola pembeda (Singh & Singh, 2020). Metode normalisasi yang sering digunakan adalah *min-max normalization*, metode ini mengubah data ke range baru lain yaitu antara 0 sampai 1. *Min-max normalization* dapat dirumuskan sebagai berikut:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2.8}$$

Keterangan dari fungsi 2.8 adalah, x' adalah hasil normalisasi, x adalah data asli, x_{max} adalah nilai maksimum dari semua data asli, dan x_{min} adalah nilai minimum dari semua data asli (Sakinah et al., 2018).

2.7. Standardisasi

Standardisasi adalah jenis normalisasi dari suatu variabel acak, sehingga variabel tersebut memperoleh nilai rata-rata yang diharapkan 0 dan standar deviasi 1. Metode penskalaan ini berguna ketika data mengikuti distribusi normal (distribusi Gaussian), jika data tidak mengikuti distribusi normal maka hal ini akan menimbulkan masalah (Jamal et al., 2014). Operasi ini dilakukan sesuai dengan rumus uji Z pada persamaan 2.9 :

$$Z = \frac{x - \mu}{\sigma} \tag{2.9}$$

Dimana x adalah nilai variabel yang diamati, μ adalah nilai yang diharapkan, rata-rata, lalu σ adalah standar deviasi (Dzierżak, 2019).

2.8. *Oversampling* Data

Klasifikasi pada *imbalanced dataset* cukup sulit dilakukan karena terlalu sedikit data dari kelas minoritas untuk dapat menghasilkan model klasifikasi yang efektif. Untuk mengatasi hal tersebut, salah satu cara yang bisa dilakukan adalah melakukan oversampling yaitu melakukan duplikasi data dari kelas minoritas pada data latih agar menghasilkan model yang lebih baik.

Synthetic Minority Oversampling Technique (*SMOTE*) memilih data pada kelas minoritas secara acak kemudian mencari k data kelas minoritas terdekat dari data tersebut. Dari k data kelas minoritas tersebut, akan dipilih satu data secara acak yang selanjutnya dihubungkan dengan data awal yang dipilih untuk membentuk segmen garis pada ruang fitur. Data sintesis dihasilkan menggunakan kombinasi convex antara dua data yang dipilih secara acak tadi. Pendekatan ini dinilai cukup efektif karena data sintesis yang dibentuk relatif dekat dalam ruang fitur dengan data yang ada dari kelas minoritas (Sabilla & Vista, 2021).

Misalkan diberikan data dengan jumlah variabel p maka jarak antara $X^T = [x_1, x_2, \dots, x_n]$ dan $Z^T = [z_1, z_2, \dots, z_n]$ adalah $d(x, y) = \sqrt{(x_1 - z_1)^2 + (x_2 - z_2)^2 + \cdots + (x_n - z_n)^2}$. Untuk membangkitkan data dengan metode SMOTE maka digunakan persamaan 2.10 berikut:

$$x_{syn} = x_i + (x_{knn} - x_i)\gamma \quad (2.10)$$

x_{syn} merupakan pengamatan baru hasil pembangkitan, x_i adalah pengamatan ke-i, x_{knn} merupakan x terdekat dari x_i, serta gamma merupakan bilangan acak antara 0 dan 1. Untuk data nominal maka akan diisi dengan nilai mayoritas pada k-tetangga terdekat. Perhitungan jarak pada *SMOTE* apabila ada variabel kategorik maka akan diganti dengan kuadrat median standar deviasi variabel kontinyu kelas minoritas jika nilai kategorik pada pengamatan ke-i dan j berbeda (Syukron et al., 2020).

2.9. *Extreme Learning Machine*

Extreme learning machine (*ELM*) adalah algoritma pembelajaran baru untuk *single-hidden layer feedforward neural networks* (*SLFNs*) yang pertama kali diperkenalkan oleh Huang (2004), yang memiliki beberapa fitur menarik dan signifikan yang berbeda dari algoritma pembelajaran tradisional berbasis gradien populer untuk jaringan saraf *feedforward* (Huang et al., 2004):

1. Kecepatan belajar *ELM* sangat cepat. *ELM* dapat melatih *SLFN* jauh lebih cepat daripada pembelajaran klasik algoritma.
2. Berbeda dengan algoritma pembelajaran berbasis gradien klasik tradisional yang bermaksud untuk mencapai kesalahan pelatihan minimum tetapi tidak mempertimbangkan besarnya bobot, *ELM* cenderung tidak hanya mencapai kesalahan pelatihan terkecil tetapi juga norma bobot terkecil.
3. Berbeda dengan algoritma pembelajaran berbasis gradien klasik tradisional yang hanya berfungsi untuk fungsi aktivasi yang dapat didiferensiasikan, algoritma pembelajaran *ELM* dapat digunakan untuk melatih *SLFN* dengan fungsi aktivasi yang tidak dapat dibedakan.
4. Algoritma pembelajaran *ELM* terlihat jauh lebih sederhana daripada kebanyakan algoritma pembelajaran untuk jaringan saraf *feedforward*.

Untuk N dengan jumlah sampel yang berbeda (x_i, t_i), dimana $x_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T \in R^n$ dan $t_i = [t_{i1}, t_{i2}, \dots, t_{in}]^T \in R^m$ *SLFNs* standar dengan

jumlah layer tersembunyi sebanyak Ñ dengan fungsi aktivasi $g(x)$ secara matematis dapat dimodelkan pada persamaan 2.11 sebagai berikut (Huang et al., 2006):

$$\sum_{i=1}^{\tilde{N}} \beta_i g_i(w_i \cdot x_j + b_i) = o_j \qquad j = 1, \dots, N \tag{2.11}$$

Dimana $W_i = [W_{i1}, W_{i2}, \dots, W_{in}]^T$ adalah matriks bobot yang menghubungkan antara neuron input dan neuron pada layar tersembunyi, $\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{in}]^T$ adalah vektor bobot yang menghubungkan neuron pada layer tersembunyi dengan neuron output dan b_i adalah bias dari neuron pada layer tersembunyi. $w_i \cdot x_j$ adalah hasil perkalian dari w_i dan x_j. Jika diasumsikan bahwa error yang didapatkan adalah 0 berarti $\sum_{j=1}^{\tilde{N}} \|o_j - t_j\| = 0$, sehingga persamaan menjadi seperti persamaan 2.12 berikut:

$$\sum_{i=1}^{\tilde{N}} \beta_i g_i(w_i \cdot x_j + b_i) = t_j \qquad j = 1, \dots, N \tag{2.12}$$

Persamaan N di atas dapat ditulis secara ringkas pada persamaan 2.13 sebagai berikut:

$$H\beta = T \tag{2.13}$$

Dimana untuk detail persamaan 2.13 bisa dilihat pada persamaan 2.14, 2.15, dan 2.16 dibawah :

$$H = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_{\tilde{N}} \cdot x_1 + b_{\tilde{N}}) \\ \vdots & \dots & \vdots \\ g(w_1 \cdot x_N + b_1) & \cdots & g(w_{\tilde{N}} \cdot x_N + b_{\tilde{N}}) \end{bmatrix}_{N \, x \, \tilde{N}} \tag{2.14}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \text{ x m}} \tag{2.15}$$

$$T = \begin{bmatrix} t_1^T \\ \vdots \\ t_{\tilde{N}}^T \end{bmatrix}_{N \, x \, m} \tag{2.16}$$

Pada persamaan 2.14 diatas, H disebut matriks keluaran lapisan tersembunyi dari jaringan saraf, kolom ke-i dari H adalah output simpul tersembunyi ke-i sehubungan dengan input $x_1, x_2, \dots, x_N$.

Proses pelatihan pada metode *ELM* bertujuan untuk mencari bobot β. Dengan memodifikasi persamaan 2.11 maka nilai β dapat diperoleh

$$\beta = H^{+}T \tag{2.17}$$

Pada persamaan 2.17 diatas, H^{+} adalah invers umum Moore–Penrose dari matriks H. Dengan demikian, metode pembelajaran *ELM* yang diberi data training $N = \{(x_i, t_i) | x_i \in R^n, t_i \in R^m, i = 1, \dots, N\}$, menggunakan fungsi aktivasi $g(x)$, dan neuron pada layer tersembunyi sebanyak Ñ.

Step 1 : Tetapkan bobot input secara acak w_i dan bias b_i, $i = 1, \dots, \tilde{N}$

Step 2 : Hitung matriks keluaran lapisan tersembunyi H

Step 3 : Hitung berat keluaran β

Jumlah neuron pada algoritma *ELM* mempengaruhi hasil perhitungan yang nantinya akan dilakukan, semakin banyak jumlah neuron, semakin banyak pula penghubung antara lapisan masukkan ke lapisan keluaran sehingga pengenalan pola yang dilakukan oleh sistem akan semakin baik. Jumlah fitur juga mempengaruhi perhitungan, karena jika terlalu sedikit fitur yang digunakan maka pengenalan pola yang diperoleh terlalu sedikit sehingga sistem akan kesulitan mempelajari pola tersebut. Pengujian jumlah data latih dan data uji juga akan mempengaruhi hasil evaluasi, hal ini terjadi karena jika jumlah data latih terlalu sedikit maka sistem akan kesulitan untuk melakukan pengenalan pola (Maulidya Ashar et al., 2018).

2.10. Matriks Konfusi

Evaluasi performa model diperlukan untuk menilai seberapa baik model pada tahap pengujian yang telah dilakukan. Pengujian model bisa dilakukan dengan berbagai cara, pada penelitian ini penulis menggunakan matriks konfusi untuk mengevaluasi performa dari model *ELM* yang telah dibuat.

Tabel 2.1 Matriks Konfusi

		Nilai Asli	
	label	Positif (1)	Negatif (0)
Nilai Prediksi	Positif (1)	TP	FP
	Negatif (0)	FN	TN

Tabel 2.1 merupakan tabel dari matriks konfusi, dimana *TP* (*True Positive*) adalah jumlah data dengan label positif (1) dengan benar terklasifikasi dengan hasil klasifikasi positif (1) juga, *FP* (*False Positive*) adalah jumlah data dengan label negatif (0) terklasifikasi dengan hasil klasifikasi positif (1), *FN* (*False Negative*) adalah jumlah data dengan label positif (1) terklasifikasi dengan hasil klasifikasi negative (0), dan *TN* (*True Negative*) adalah jumlah data dengan laberl negative (0) dengan benar terklasifikasi dengan hasil klasifikasi negative (0) juga. Hasil dari matriks konfusi dapat digunakan untuk mengukur nilai *accuracy*, *precission*, *recall*, *specificity*, dan *F1 Score*, dengan masing-masing memiliki perhitungan sebagai berikut :

1. *Accuracy* merupakan rasio prediksi benar (positif dan negatif) dengan keseluruhan data. Persamaan *accuracy* dapat dilihat pada Rumus 2.18.

$$Accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)} \quad (2.18)$$

2. *Precission* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif. Persamaan *precission* dapat dilihat pada Rumus 2.19.

$$Precission = \frac{(TP)}{(TP + FP)} \quad (2.19)$$

3. *Recall* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. Persamaan *recall* dapat dilihat pada Rumus 2.20.

$$Recall = \frac{(TP)}{(TP + FN)} \quad (2.20)$$

4. *Specificity* merupakan kebenaran memprediksi negative dibandingkan dengan keseluruhan data negative. Persamaan *specificity* dapat dilihat pada Rumus 2.21.

$$Specificity = \frac{(TN)}{(TN + FP)} \quad (2.21)$$

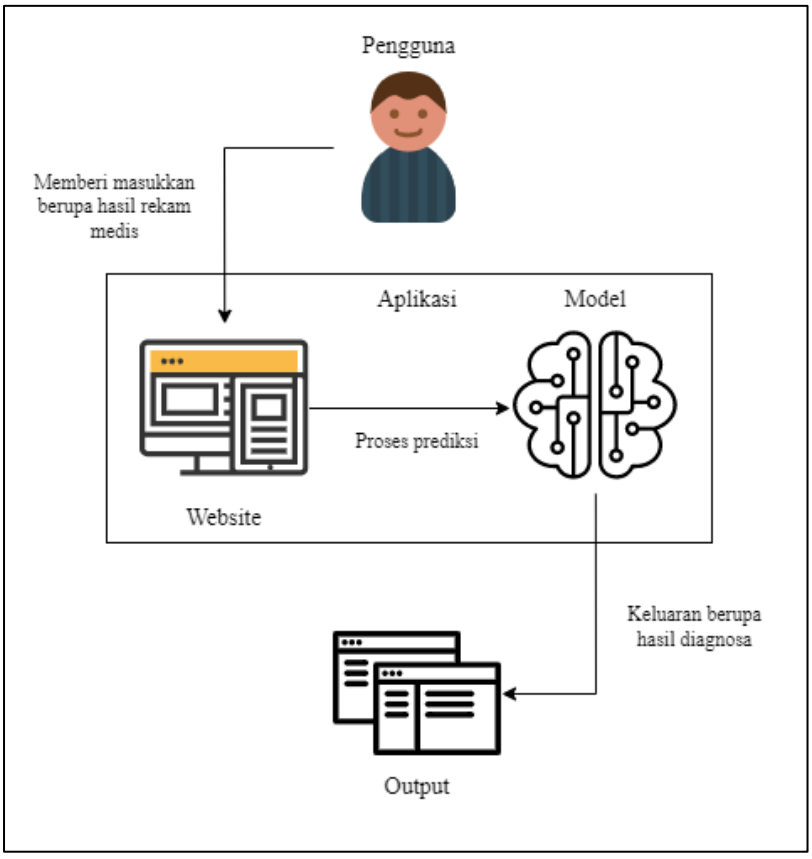
5. *F1 Score* merupakan perbandingan rata-rata *precission* dan *recall.* Persamaan *F1 score* dapat dilihat pada Rumus 2.22.

$$F1\ Score = \ 2\ x\ \frac{(recall\ x\ precission)}{(recall + precission)} \quad (2.22)$$

2.11. *Machine Learning as a Service*

Machine learning (ML) dan *artificial intelligence (AI)* telah dengan cepat mengubah banyak industri seperti *e-commerce*, transportasi dan perawatan kesehatan, ML dan AI secara tradisional dilatih dalam satu mesin menggunakan alat seperti Scikit-learn atau R dengan model terlatih yang digunakan secara manual. Namun, di era layanan *Big Data* dan *real time* sesuai permintaan, ini sangat membatasi skala penerapan ML. Menyadari adanya kebutuhan, banyak perusahaan seperti Facebook dan penyedia *cloud* seperti Microsoft, Amazon, dan Google telah membangun platform pembelajaran mesin untuk mengotomatisasi dan menskalakan pembelajaran mesin (Erran Li et al., 2017).

Machine Learning as a Service adalah arsitektur pengiriman komputasi modern yang membuat komponen ML tersedia melalui antarmuka berorientasi manusia (misalnya, browser web) dan berorientasi aplikasi (misalnya, API) melalui Internet. Layanan ini menyediakan API yang andal untuk klasifikasi, regresi, analisis klaster, deteksi anomali, dan penemuan asosiasi melalui koneksi Internet (Perner, 2017). Gambar 2.10 merupakan rancangan *machine learning as a service* yang diusulkan penulis.



Gambar 2.10 Rancangan *Machine Learning as a Service* yang Diusulkan Penulis