BAB II

TINJAUAN PUSTAKA

2.1. *Hepatocelluler Carsinoma*

Penyakit kanker adalah salah satu penyebab kematian utama di seluruh dunia. Kanker menjadi penyebab kematian sekitar 8,2 juta orang. Menurut Data GLOBOCAN, *International Agency for Research on Cancer* (IARC) dikatakan bahwa pada tahun 2012 terdapat 14.067.894 kasus baru penyakit kanker dan 8.201.575 kematian akibat penyakit kanker di seluruh dunia. Setiap tahunnya kematian akibat kanker sebagian besar disebabkan oleh kanker hati, paru, perut, kolorektal serta kanker payudara (KemenKes, 2015).

Kanker merupakan salah satu penyakit yang disebabkan oleh pertumbuhan sel yang tidak normal. Proses pada semua jenis kanker yaitu sel tubuh akan membelah tanpa berhenti dan akan menyebar ke jaringan yang terdapat disekitarnya. Awal terbentuknya kanker adalah hampir dimanapun yang ada pada tubuh manusia yang terdiri dari triliunan sel, sel pada manusia tersebut akan tumbuh dan membelah menjadi sel baru. Namun pada saat kanker berkembang, sel-sel tersebut akan menjadi semakin tidak normal dan sel-sel tua atau sel-sel yang rusak akan bertahan hidup ketika mereka harus mati, serta sel-sel baru akan terbentuk saat mereka tidak diperlukan. Sel-sel baru ini dapat membelah tanpa henti dan dapat membentuk pertumbuhan yang biasanya disebut tumor. Kebanyakan dari kanker akan membentuk tumor padat yang merupakan massa dari jaringan (NCI, 2015).

American Cancer Society mengatakan bahwa pada tahun 2017 sekitar 600 orang di Amerika diperkirakan meninggal karena kanker, yang artinya per hari terdapat sekitar 2 orang yang meninggal. Kanker merupakan salah satu penyebab kematian yang paling umum kedua di Amerika Serikat setelah penyakit jantung, dan penyakit kanker sendiri menyumbang hampir 1 dari 4 kematian.

Hepatocelluler Carsinoma atau HCC merupakan suatu benjolan atau tumor ganas yang terjadi pada hati. Kanker hati yang timbul biasanya terjadi pada seseorang yang memiliki infeksi Hepatitis B atau C ataupun penyakit hati kronik lainnya seperti sirosis hati. Kanker hati primer yang terjadi berasal dari hepatosis (*Hepatocelluler Carsinoma*) ataupun berasal dari duktus empedu (kongaliokarsinoma). Sedangkan untuk kanker hati sekunder yang muncul terjadi akibat metastasis kanker yang berasal dari bagian tubuh lain yang mengalirkan darahnya ke hati melalui vena porta atau kanker lainnya (Corwin, 2008). Penyebab terjadinya HCC adalah hepatitis B , hepatitis C dan sirosis hati yang disebabkan oleh konsumsi alcohol, diet tinggi, aflatoksin, penderita diabetes, obesitas dan akibat dari seringnya terkena paparan bahan kimia (Maharani, 2015).

2.2. Hepatitis B

Hepatitis B adalah salah satu penyakit infeksi yang terjadi pada jaringan hati dan disebabkan oleh virus yang berasal dari family *hepadnavirus*. Ukuran dari virus tersebut tergolong sangat kecil, yaitu berkisar 42 nm dan dapat dilihat dengan mikroskop electron. Terdapat dua jenis penyakit hepatitis B, yaitu hepatitis B akut dan hepatitis B kronik. Sementara itu sekitar 30% dari hepatitis B kronik akan

berkembang menjadi kanker hati. Kasus hepatitis B ini dapat menyerang semua usia, ras maupun jenis kelamin (Masriadi, 2014).

Hepatitis B kronik didefinisikan sebagai peradangan hati yang berlanjut yang lebih dari enam bulan sejak munculnya gejala penyakit dan keluhan dari penderita. Bagian tubuh manusia yang rentan terhadap penularan dari Virus Hepatitis B (VHB) antara lain darah dan produk darah, air ludah, cairan *cerebrospinal*, *peritoneal*, *pleural*, cairan *pericardial* dan *synovial*, cairan *amniotic*, semen, cairan vagina, cairan bagian tubuh lainnya yang berisi darah, serta organ dan jaringan tubuh yang terlepas.

Penularan VHB yang sering terjadi adalah adanya kontak seksual atau kontak rumah tangga dengan seseorang yang terkena VHB, penularan perianal yang terjadi dari ibu kepada banyinya, penggunaan alat suntik oleh pecandu obat-obatan terlarang serta melalui pajanan nosocomial di rumah sakit (Masriadi, 2014).

2.3. Hubungan *Hepatocelluler Carsinoma* dengan Hepatitis B

Hubungan antara HCC dengan infeksi hepatitis B dapat dikatakan sangan kuat secara epidemiologis, klinis maupun eksperimental. Karsogenitas VHB terhadap hati dapat terjadi melalui inflamasi kronik, peningkatan poliferasi hepatosit, integrase VHB DNA ke dalam DNA sel pejamu serta aktivitas protein spesifik VHB yang berinteraksi dengan gen hati. Perubahan hepatosit dari kondisi yang inaktif menjadi sel aktif akan berimplikasi menentukan karsinogenesis hati. Siklus dari sel tersebut dapat diaktifkan secara tidak langsung oleh kompensasi poliferaktif yang merespon nekroinflamasi sel hati, atau karena akibat dipicu oleh

ekspresi yang berlebihan dari satu atau beberapa gen yang berubah akibat VHB (Ayuningtyas, 2015).

Proses terjadinya HCC pada VHB ada tiga tahapan, yaitu inisiasi, promosi dan progresi. Tahapan inisiasi ini akan terjadi integrase antara genom VHB ke genom hepatosit. Pada tahapan promosi akan mulai terjadi ekspansi klonal dari sel-sel yang terangsang dalam tahapan inisiasi. Selanjutnya pada tahapan progresi, sel-sel yang mengalami transformasi keganasan akan mengalami replikasi lebih lanjut. Beberapa kasus yang terjadi pada penderita hepatitis B yang berkembang menjadi HCC bisa langsung terjadi tanpa adanya proses sirosis.

2.4. Bioinformatika

Bioinformatika merupakan ilmu yang menggabungkan beberapa disiplin ilmu yang saling terhubung, yaitu biologi, informatika, matematika dan statistika. Menurut (Luscombe, Greenbaum, & Gerstein, 2001), bioinformatika menyertakan diri dengan memanfaatkan komputer untuk menyimpan, mencari keterangan, manipulasi, serta distribusi terkait data biologi macromolekul seperti DNA, RNA, serta Protein. Bioinformatika tidak hanya digunakan untuk riset dasar genom serta biologi molekuler, namun digunakan juga untuk bidang bioteknologi serta ilmu biomedis, misalnya aplikasi dalam desain obat berbasis pengetahuan, analisis DNA forensik, bioteknologi pertanian serta menciptakan biomarker untuk beberapa kasus penyakit penyakit (Xiong, 2006).

Menurut Fatchiyah (2009), bioinformatika dikemukakan pada tahun 1980-an yang mengacu pada penerapan komputasi pada biologi. Basis data dibuat dalam

bioinformatika telah dilakukan sejak tahun 1960-an. Ilmu dari bioinformatika lahir atas inisiatif para pakar komputer bersumber pada AI(Artificial Intelligence), para pakar berfikir jika seluruh indikasi yang terdapat dialam mampu dibuat secara artificial melalui adanya simulasi dari tanda- tanda tersebut. Data merupakan kunci utama untuk menentukan tindak lanjut dari tanda-tanda alam tersebut yang merupakan gen DNA serta RNA. Program yang didukung oleh internet merupakan fitur utama untuk bioinformatika. Salah satu analisis yang digunakan pada bioinformatika adalah analisis pada data ekspresi gen, yaitu beberapa gen dapat ditentukan dengan mengukur lever dari gen tersebut dengan menggunakan berbagai macam metode seperti *microarray* (Raza, 2012).

2.5. *Microarray*

Microarray merupakan salah satu dari kemajuan teknologi yang digunakan dalam penelitian dalam membantu pendekatan farmakologis untuk mengobati berbagai penyakit, serta dapat digunakan untuk pengukuran tingkat ekspresi dari gen yang terdapat pada sebuah jaringan atau sel tertentu. *Microarray* berbentuk seperti *chip* atau *slide* mikroskop yang isinya berupa rangkaian sampel jaringan, protein, RNA dan DNA, serta memiliki sampel yang berjumlah ratusan ribu yang terdapat di dalam *microarray* (Hovatta, et al., 20015).

Data *Microarray* dapat digunakan dalam mendeteksi dan mengklasifikasikan suatu jaringan penyakit pada manusia. *Microarray* menghasilkan ekspresi gen yang berisi informasi-informasi gen, kemudian dicocokkan dengan penyakit tertentu. Teknologi yang digunakan pada *Microarray* biasanya untuk *genotype* dengan skala

yang besar, profil pada ekspresi gen, hibridasi genomik serta penyeimbang dalam penggunaan aplikasi lainny. *Microarray* merupakan hasil dari kombinasi dari beberapa bidang teknolongi dan penelitian, seperti mekanik, pembuatan mikro, kimia, perilaku DNA, mikrofluida, enzim, optik serta bioinformatika (Dufva, 2009).

2.6. Ekspresi Gen

Ekspresi gen atau *Gene Expression* merupakan proses transkripsi pada DNA dalam sel menjadi RNA (Madigan, Martinko, V, & Clark, 2008). *Deoxyribonucleic* Gen merupakan urutan dari DNA yang memberikan kode protein, dimana protein merupakan pengendali dari sifat fisik sel seperti pada warna mata dan rambut. *Acid* atau DNA adalah sebuah asam nukleat yang didalamnya menyimpan informasi-informasi tentang genetika. Informasi yangterdapat pada DNA tersimpan dalam bentuk kode yang terdiri dari beberapa basis kimia, seperti adenine (A), guanine (G), sitosin (S) serta timin (T). Sedangkan untuk DNA berbeda dengan RNA, RNA merupakan sebuah untai tunggal yang memiliki panjang hanya 75 – 5000 nukleotida. Dalam suatu sel memiliki kandungan beberapa jenis RNA, yaitu : *messenger* RNA (mRNA), RNA transfer (tRNA) serta RNA ribosom (rRNA) (Bolstad, 2004).

2.7. *Preprocessing*

Preprocessing merupakan tahapan yang penting dalam sebuah analisis. Hal tersebut dikarenakan pada sebagian besar data yang diambil atau diperoleh

merupakan data mentah yang didalamnya terdapat data yang hilang, data noise (mengandung kesalahan), tidak konsisten (terdapat perbedaan kode atau nama) serta kualitas data. Salah satu tujuan pada tahap *preprocessing* adalah membersihkan data, normalisasi data, integrase data serta melakukan pengurangan data (Johan, 2017).

Terdapat 3 tahapan pada proses *preprocessing*, yaitu :

1. *Background Correction*, yaitu tahapan untuk menghilangkan *background noise*, penyesuaian *cross hibdrization cross hibdrization* yang menjadi pengikat dari DNA non spesifik yang melekat dengan *array*.
2. Normalisasi data, yaitu tahapan yang digunakan untuk menghilangkan variasi *non* biologis yang tidak diinginkan dan memungkinkan terdapat *array* sehingga data akan menjadi homogeny.
3. *Summarization*, yaitu proses yang berguna untuk mengukur gen yang ada pada chip sehingga menghasilkan nilai ekspresi gen.

Bolstad (2014) mengatakan bahwa bantuan yang digunakan pada tahapan *preprocessing* adalah dengan *package* dari AffyPLM dengan fungsi *threestep*. Perintah *threestep* ini merupakan alternative yang dapat digunakan untuk perhitungan pada ukuran ekspresi gen, dengan menggunakan RMA.2 dalam tahapan *background correction*, *quantile* pada tahap *normalization* serta *median* pada tahapan *summarization*.

2.8. *Filtering*

Filtering pada data ekspresi gen merupakan tahapan yang penting, karena tahapan tersebut digunakan untuk penyaringan data dan mengurangi jumlah gen yang tidak diikutsertakan pada analisis atau tidak valid, sehingga akan meningkatkan kualitas model pada analisis ekspresi gen serta membuat proses pemodelan menjadi lebih efisien. Proses tersebut akan mengurangi ruang fitur dalam analisis prediksi yang tentunya akan menyebabkan suatu model yang dilatih akan lebih cepat dan variansinmya sama (Dozmorov, 2016).

Perintah yang digunakan pada proses *filtering* adalah nsFilter atau filter non-spesifik, yang bertujuan untuk menghapus *probe* yang tidak dibutuhkan. Pada fungsi *nsFilter* terdapat perintah *var.cutoff* yang berguna untuk penyaringan gen dengan nilai tertentu serta menghapus data dengan melihat nilai *Inter Quartil Range* (IQR) dibawah quartil. Perintah *Require.entrez* yang memiliki fungsi untuk melakukan filter tanpa anotasi Entrez Gene ID, dimanapun sistem pengenal selain ID maka ID tersebut yang akan diperlukan. Perintah *Remove.DupEntrez* berguna untuk menghapus ID gen Entrez yang sama. Sedangkan perintah *Feature.exclude* untuk penyaringan *probe control* adalah AFFX agar tidak dimasukkan kedalam analisis (Gentlemen, et al. 2019).

2.9. Pemilihan Fitur

Pemilihan fitur atau *Feature Selection* merupakan proses yang dibuat agar pengklasifikasian lebih efektif dengan cara pengurangan data-data yang dianggap tidak relevan, sehingga dapat mempersingkat waktu pengklasifikasian dan dapat

pula meningkatkan akurasi (Karabulut, Ozel, & Ibrikci, 2011). Batuan yang digunakan adalah dengan *package multtest*, untuk melakukan *feature selection* dapat digunakan perintah *mt.teststat* yang mempunyai fungsi untuk perhitungan statistik uji, seperti contoh pada uji t-test, wicoxon, uji F untuk tiap baris dari kerangka data. Pengujian yang digunakan pada penelitian ini adalah uji t-test karena memiliki dua sampel. Uji t merupakan suatu pengujian dengan satu individu dan dua perlakuan yang berbeda. Hal yang dapat dilakukan adalah dengan menghapus *row* yang mempunyai nilai *p-value* yang telah ditentukan dengan menggunakan *package dplyr* (Pollard, et al., 2019).

2.10. *Machine Learning*

Machine Learning atau bisa disebut dengan pembelajaran mesin adalah gabungan dari ilmu komputer dan statistic dan bisa juga disebut dengan komputer sains. *Machine Learning* banyak digunakan pada peniruan perilaku manusia untuk menyelesaikan masalah atau melakukan otomatisasi serta digunakan untuk menirukan cara manusia untuk belajar dan menggeneralisasi. Salah satu ciri yang khas pada *Machine Learning* adalah adanya proses pelatihan dan pembelajaran, oleh sebab itu dibutuhkan data untuk dipelajari atau data *training* dan data untuk di uji atau data *testing* (Ahmad, 2017).

Proses *machine learning* pada dasarnya sama dengan *Data Mining* yang sistemnya membutuhkan data untuk menemukan pola agar dapat meningkatkan pemahaman program sendiri. Program *machine learning* mampu mendeteksi pola yang ada pada data serta mampu menyelesaikan tindakan dari program yang sesuai.

Sederhananya, *machine learning* merupakan pemrograman computer yang berguna untuk pencapaian kriteria/performa tertentu dengan sekumpulan data *training* atau pengalaman masa lalu (Primartha, 2018). Terdapat dua jenis pembelajaran pada *machine learning*, yaitu:

1. *Supervised Learning*, yaitu jenis pembelajaran yang memiliki variabel *input* dan *output* yang telah diketahui kelasnya. Variabel tersebut dapat dibuat menjadi suatu model hubungan matematis sehingga dapat melakukan prediksi dan klasifikasi berdasarkan data sebelumnya atau data yang telah di latih. Data latih tersebut yang dapat digunakan dalam melakukan prediksi ataupun klasifikasi.
2. *Unsupervised Learning*, yaitu jenis pembelajaran yang hanya memiliki data *input* dan tidak terdapat *output* variabel yang berhubungan. Pendekatan pada pembelajaran ini tidak menggunakan data latih atau training untuk melakukan prediksi ataupun klasifikasi. Berdasarkan model matematisnya, pembelajaran ini tidak mempunyai target variabel, tujuannya adalah untuk mengelompokkan objek yang hampir sama dalam suatu area tertentu.

Terdapat beberapa algoritma yang ada pada *machine learning*, yaitu regresi linier, regresi logistik, *decision tree, naive bayes, k-nearest neighbor* (KNN), *neural network, random forest* dan *Support Vector Machine* (SVM) (Attaran & Deb, 2018).

2.11. Klasifikasi

Teknik pada klasifikasi yang merupakan salah satu contoh dari algoritma *machine learning* adalah *supervised learning*. *Supervised Learning* adalah sebuah algoritma yang digunakan dalam *machine learning* dengan menggunakan data latih (*training*) yang sebelumnya telah diberi label dengan jawaban yang benar untuk memprediksi beberapa target dari beberapa kelas. Sedangkan algoritma merupakan suatu langkah-langkah atau urutan yang digunakan untuk perhitungan dalam menyelesaikan suatu masalah yang ditulis secara berurutan (Primartha, 2018). Terdapat dua hal utama yang perlu dilakukan pada klasifikasi, yaitu dengan melakukan *training* yang disimpan untuk prediksi serta melakukan *training* sebagai proses dalam klasifikasi untuk mengetahui di label mana objek data tersebut (Liu, Loh, & Sun, 2009).

Menurut (Gorunescu, 2011), klasifikasi adalah suatu proses dalam menempatkan suatu objek tertentu yang terdapat pada satu set kategori yang didasarkan pada empat komponen klasifikasi, yaitu :

a. Kelas, yaitu variabel dependen yang berupa data kategorik untuk mempresentasikan label yang terdapat pada suatu objek setelah klasifikasi. Misalnya, kelas hujan dan kelas panas.
b. Prediktor, merupakan variabel independen yang telah diwakili oleh karakteristik suatu data yang akan diklasifikasikan berdasarkan klasifikasi yang dibuat. Misalnya waktu, musim dan lokasi.

c. *Training* dataset, yaitu kumpulan dari sebuah data yang terdiri dari komponen kelas dan prediktor yang digunakan untuk melatih model dalam memperoleh kelas yang sesuai berdasarkan hasil prediksi dari model.

d. *Testing* dataset, yaitu suatu data baru yang digunakan untuk klasifikasi dari model yang diperoleh dari *training* dataset dan akurasi hasil klasifikasi dapat dievaluasi.

2.12. Jaringan Saraf Tiruan

Jaringan Syaraf Tiruan (JST) adalah suatu sistem komputasi yang di desain dengan menirukan cara kerja dari otak manusia untuk menyelesaikan dengan proses belajar melalui perubahan bobot sinapsisnya. Sekarwati (2005) mengatakan bahwa JST adalah sistem komputasi yang berdasarkan pemodelan pada sistem biologis (neuron) dengan pendekatan sifat-sifat biologis (*biological computation*). JST pertama kali ditemukan oleh Mc. Culloch dan Pitts yang kemudian berkembang pesat dan telah banyak digunakan di banyak aplikasi. Pemodelan pada JST ini didasari oleh kemampuan otak pada manusia dalam mengoorganisasi sel-sel penyusunan (*neuron*) sehingga mampu untuk melakukan tugas-tugas tertentu, terkhusus pengenalan pola dengan efektifitas jaringan tinggi (Suyanto, 2013).

JST terdiri dari kumpulan grup *neuron* yang tersusun dalam beberapa lapisan, antara lain:

1. *Input Layer*, berfungsi untuk menghubungkan sumber data ke jaringan pemrosesan. Setiap masukan akan mempresentasikan variabel bebas yang akan berpengaruh terhadap keluaran.

2. *Hidden Layer*, yaitu lapisan tersembunyi yang berfungsi untuk mendapatkan hasil output yang sesuai dengan keinginan. Pada JST *Multilayer* dapat memiliki satu atau lebih *hidden layer*.
3. *Output Layer*, merupakan lapisan yang berisi hasil dari pemrosesan JST. Output yang diperoleh dipengaruhi oleh bobot, jumlah *hidden layer* dan fungsi aktivasi yang diterapkan.

Model yang ada pada JST ditentukan berdasarkan arsitektur jaringan serta algoritma pelatihan yang digunakan. Arsitektur jaringan tersebut menjelaskan arah perjalanan dari data yang terdapat di dalam jaringan. Sedangkan yang menjelaskan bagaimana bobot koneksi diubah agar pasangan dari *input-output* yang diinginkan tercapai adalah algoritma pembelajaran.

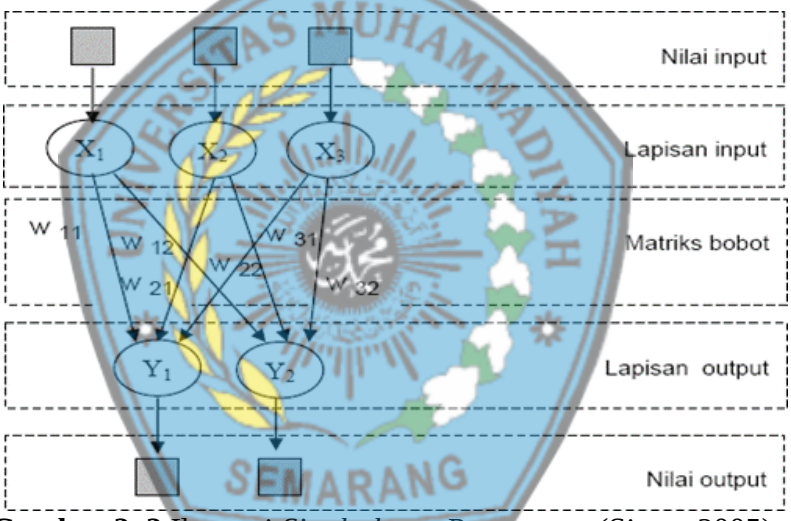
Gambar 2. 1 Sel Jaringan Syaraf Tiruan

Berdasarkan gambar diatas, diketahui bahwa sebuah sel syaraf tiruan sebagai elemen penghitung. Pada simpul Y menerima masukan dari 3 neuron x1, x2 dan x3 dengan bobot dari masing-masing masukan w1, w2 dan w3. Fungsi Aktivasi yang merupakan net (jejaring) masukan adalah kombinasi dari linier masukan dan bobotnya, sehingga *net=x1w1+x2w2+x3w3*. Besar sinyal yang akan diterima oleh

Y akan mengikuti nilai dari fungsi aktivasi *y=f(net)*.

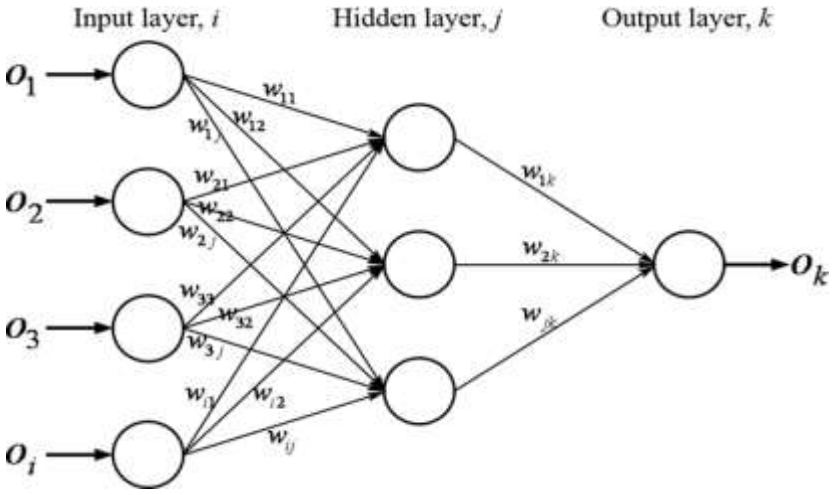
Pemodelan pada arsitektur JST yang sering digunakan terdapat 2 macam, antara lain:

1) *Single-Layer Perceptron*, yaitu pemodelan arsitektur jaringan berlapis yang paling sederhana. Cara kerjanya adalah *input layer* dengan *node* sumber yang diproyeksi ke *output layer* akan tetapi tidak berlaku untuk sebaliknya. Pemodelan jaringan ini merupakan pemodelan jaringan umpan maju atau *feedforward*. Ilustrasi untuk *Single-layer Perceptron* adalah sebagai berikut:



Gambar 2. 2 Ilustrasi *Single-layer Perceptron* (Siang, 2005)

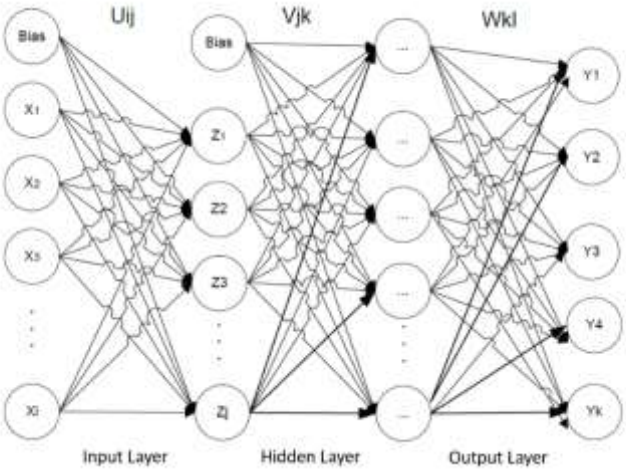
2) *Multi-Layer Perceptron*, yaitu pemodelan pada arsitektur jaringan yang mempunyai banyak lapisan. Pada *single-layer* jika ditambahkan satu atau lebih *hidden layer* maka akan mengganggu jaringan karena *input* dan *output* dari jaringan tidak dapat melihat *hidden layer* yang telah dimasukkan. Oleh karena itu diperlukan sebuah jaringan yang dapat menampung, yaitu *multilayer*. Berikut merupakan ilustrasi dari *multilayer perceptron*:



Gambar 2. 3 Ilustrasi *Multilayer Perceptron* (Popcescu et al, 2009)

2.12.1. *Multilayer Perceptron*

Multilayer Perceptron adalah sebuah arsitektur dari JST yang paling banyak digunakan dalam permasalahan klasifikasi yang berupa JST *feedforward* dengan satu atau lebih *hidden layer*. Biasanya jaringan jaringan pada MLP terdiri dari satu *layer neuron* komputasi keluaran. Sinyal masukan dipropagasikan dengan arah maju pada *layer* per *layer*. MLP memiliki tiga lapisan, yaitu *input layer, hidden layer* serta *output layer*. Berikut merupakan contoh dari arsitektur pada MLP:



Gambar 2. 4 Arsitektur *Multilayer Perceptron* (S.E Fahlaman, 1987)

Berdasarkan gambar di atas, tiap koneksi antar *layer* dihubungkan dengan bobot U_{ij} untuk koneksi antara *input layer* (x_i) menuju *hidden layer* (Z_j), kemudian untuk bobot V_{jk} untuk koneksi antara *hidden layer* (Z_j) menuju *output layer* (Y_k) berikutnya, serta bobot W_{kl} untuk koneksi antara *hidden layer* (Z_j) menuju *output layer* (Y_k). Proses pembelajaran yang terdapat pada MLP bertujuan untuk menemukan bobot sinaptik yang paling optimum untuk mengklasifikasian himpunan data latih dan data uji. Algoritma pembelajaran yang banyak digunakan untuk MLP yaitu *Backpropagation*.

2.12.2. Algoritma *Backpropagation*

Algoritma *Backpropagation* adalah algoritma pelatihan yang menggunakan banyak lapisan dalam mengubah bobot yang terhubung dengan neuron-neuron yang terdapat pada *hidden layer* (Haryati, Abdillah, & Hadiana, 2016). Algoritma *Backpropagation* melakukan pelatihan pada MLP dengan dua tahap, yaitu perhitungan maju (*Feedforward*) dan perhitungan mundur (Backpropagation). *Feedforward* digunakan untuk menghitung galat (*loss function*) antara prediksi target dan nilai aktual. Sedangkan *Backpropagation* digunakan untuk mempropagasikan balik galat untuk melakukan prediksi bobot sinaptik pada semua neuron yang ada. Dasar pada proses pembelajaran pada MLP yaitu menemukan bobot sinaptik yang menghasilkan *hyperplane* dengan kemiringan dan posisi yang tepat agar dapat mengklasifikasikan pola-pola yang ada pada himpunan data latih dengan kesalahan yang kecil (Suyanto, Machine Learning Tingkat Dasar dan Lanjut, 2018).

Tahapan yang terdapat pada *Feedforward* dan *Backpropagation* dalam algoritma *Backpropagation* adalah :

1. Tahap Feedforward

   a. Tiap lapisan *input* (x_i, i=1,2,3, …, n) menerima sinyal *input* (x_i) yang selanjutnya diteruskan ke unit-unit yang terdapat pada lapisan tersembunyi (*hidden layer*).

   b. Tiap unit lapisan *hidden* (z_j, j=1,2,3, …,p) menjumlahkan *input* yang telah diberi bobot dengan persamaan:

$$Z\_in_{jk} = V_{oj} + \sum_{i=0}^{n} X_i V_{ij} \qquad (2.2)$$

   Selanjutnya menghitung sinyal *hidden layer* menggunakan fungsi aktivasi dengan persamaan:

$$Z_j = f(Z\_in_j) \qquad (2.3)$$

   c. Mengkalikan tiap unit lapisan *output* (Y_k, k=1,2,3, …, m) dengan bobot serta dijumlahkan dengan biasnya menggunakan persamaan:

$$Y\_in_k = W_{oj} + \sum_{j=1}^{p} Z_j W_{jk} \qquad (2.4)$$

   Selanjutnya dihitung sinyal *output* menggunakan fungsi aktivasi persamaan berikut:

$$Y_k = f(Y\_in_k) \qquad (2.5)$$

2. Tahap *Backpropagation*

   a. Tiap unit lapisan *output* (Y_k, k=1,2,3, …, m) menerima pola target (t_k) sesuai pola *input* saat pelatihan yang kemudian informasi *error* pada lapisan *output* (δ_k) didefinisikan dengan persamaan:

$$\delta_k = (t_k - y_k) f'(y_k)(1 - y_k) \tag{2.6}$$

   Selanjutnya menghitung koreksi bobot yang akan digunakan untuk memperbaiki nilai W_{jk} dengan persamaan:

$$\Delta W_{jk} = \alpha \delta_k Z_j \tag{2.7}$$

   Kemudian hitung koreksi bias yang digunakan untuk memperbaiki nilai W_{0k} dengan persamaan:

$$\Delta W_{0k} = \alpha \delta_k \tag{2.8}$$

   b. Tiap unit lapisan *hidden* (j, j=1,2,3, …, p) dijumlahkan dengan delta *input* dari unit yang berada di lapisan sebelumnya menggunakan persamaan:

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k W_{jk} \tag{2.9}$$

   Kemudian hasilnya dikalikan dengan turunan fungsi aktivasi yang digunakan dalam jaringan untuk menghitung kesalahan *error* δ_j dengan persamaan:

$$\delta_j = \delta\_in_j f'(Z\_in_j) \tag{2.10}$$

   δ_j untuk menghitung koreksi *error* Δv_{ij} yang kemudian digunakan untuk memperbarui V_{ij} dengan persamaan:

$$\Delta v_{ij} = \alpha \delta_j x_i \quad (2.11)$$

Kemudian hitung koreksi bias Δv_{0j} untuk memperbarui V_{0j} dengan persamaan:

$$\Delta v_{0j} = \alpha \delta_j \quad (2.12)$$

c. Tiap unit *output* (k=1,2,3, …, m) untuk memperbaiki bias dan memperbarui bobot (j=0,1,2,3, …, p) sehingga akan menghasilkan bobot dan bias yang baru dengan persamaan:

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \quad (2.13)$$

Hal tersebut juga berlaku untuk bobot pada unit *input* ke unit *hidden*, diperbaiki menggunakan bobot yang sebelumnya telah dihitung dengan persamaan:

$$v_{jk}(baru) = v_{jk}(lama) + \Delta v_{jk} \quad (2.14)$$

d. Kemudian periksa kondisi berhenti. Apabila kondisi berhenti telah terpenuhi maka pelatihan jaringan dapan dihentikan. Cara untuk menentukan kondisi berhenti, salah satunya adalah dengan membatasi banyaknya *epoch* yang digunakan.

2.12.3. Fungsi Aktivasi

Fungsi aktivasi merupakan fungsi yang berfungsi untuk melakukan tranformasi suatu *input* menjadi *output*. Fungsi aktivasi dapat dikatakan operasi matematika yang digunakan untuk perhitungan *output*. Terdapat beberapa fungsi aktivasi yang dapat digunakan pada arsitektur MLP, salah satunya adalah fungsi

aktivasi *Rectified Linier Unit* (ReLu). Fungsi aktivasi tersebut termasuk fungsi yang sangat popular karena sangat mudah untuk dioptimalkan, serta tidak mudah jenuh karena tidak *asymptotic* (Jin, et al., 2015). Definisi fungsi ReLu adalah:

$$h(x_i) = \max(0, x_i) \tag{2.15}$$

Dimana, fungsi x_i adalah *input* dan $h(x_i)$ adalah *output.*

2.13. *Boosting*

Boosting adalah salah satu teknik *ensemble* yang biasanya digunakan untuk proses analisis klasifikasi maupun prediksi. Teknik *ensemble* adalah suatu metode pada algoritma pembelajaran yang dibangun oleh beberapa model klasifikasi maupun prediksi yang nantinya digunakan untuk melakukan klasifikasi pada data baru berdasarkan bobot prediksi dari hasil sebelumnya (Dietterich, 2000). Konsep *ensemble* pada *boosting* adalah dengan melatih model secara sekuensial, kemudian model tersebut digabung untuk melakukan prediksi. Model baru yang telah dihasilkan, belajar dari kesalahan model dari hasil sebelumnya (Zhou, 2012).

Langkah awal pada teknik *gradient boosting* adalah dengan memuat model pada data yang telah didefinisikan dengan persamaan:

$$F_1(x) = y \tag{2.16}$$

Kemudian model tersebut digunakan untuk perhitungan *residual* dari proses sebelumnya, seperti pada persamaan:

$$h_1(x) = y - F_1(x) \tag{2.17}$$

Kemudian membuat model baru dengan persamaan:

$$F_2(x) = F_1(x) + h_1(x) \tag{2.18}$$

Setelah semua proses dilakukan, akan diperoleh model final yang merupakan kumpulan dari model-model sebanyak n iterasi dan berakhir sampai menghasilkan nilai *error* terkecil dari *residual*. Model final tersebut dapat didefinisikan dengan persamaan:

$$\hat{F}(x) = F_1(x) \rightarrow F_2(x) = F_1(x) + h_1(x) \ldots \rightarrow F_M(x) = F_{M-1}(x) + h_{M-1}(x) \quad (2.19)$$

Model akhir dari *boosting* dapat didefinisikan dengan persamaan:

$$f(x) = \sum_{m-1}^{M} f_m(x) \quad (2.20)$$

Atau dapat pula didefinisikan dengan persamaan:

$$f(x) = \gamma_0 + \sum_{m=1}^{M} \gamma_m h_m(x) \quad (2.21)$$

Dengan $f_x = \gamma_0$ dan $f_m(x) = \gamma_m h_m(x)$ untuk $m = 1,2,3, \ldots,$ M dengan nilai $h_m(x) \in \{-1,1\}$. $\gamma_m(x)$ adalah pengklasifikasian yang lemah sedangkan γ_m merupakan bobot untuk tiap pengklasifikasi (Syahrani, 2019).

2.13.1. *Xtreme Gradient Boosting*

Xtreme Gradient Boosting atau XGBoost adalah suatu metode kombinasi antara *boosting* dengan *gradient boosting*. Pertama kali metode tersebut diperkenalkan oleh Friedman, yaitu dalam penelitiannya tentang hubungan antara *boosting* dengan optimasi untuk membuat *Gradient Boosting Machine* (GBM). Model yang dibangun dengan metode *boosting* adalah dengan membuat model baru untuk melakukan prediksi dari *error* pada model sebelumnya. Algoritma yang semacam itu dinamakan *gradient boosting*, karena menggunakan *gradient descent*

untuk memperkecil *error* pada saat pembentukan model baru. Tujun akhir dari proses tersebut adalah memperoleh fungsi yang paling mendekati $\hat{F}(x)$ terhadapt fungsi-fungsi yang membangun *f(x)* dengan cara meminimumkan nilai *loss function* L(y, *f(x)*) yang didefinisikan dengan persamaan:

$$\hat{F} = \arg\min_f E_{x,y}[L(y, f(x))] \tag{2.22}$$

Pada proses *training* tiap iterasi digunakan untuk meminimalkan nilai *loss function* berdasarkan fungsi awal $F_0(x)$. persamaan algoritma *gradient boosting* sebagai berikut:

$$\{\gamma_m, h_m\} = \arg\min \sum_{m=1}^{M} L(y_i, f^{(m-1)}(x_i) + \gamma_m h_m(x_i)) \tag{2.23}$$

XGBoost adalah salah satu varian dari *Gradient Boosting Method* (GBM) yang lebih efisien dan *scalable* karena dapat menyelesaikan beberapa fungsi seperti regresi, klasifikasi serta ranking. Pertama kalinya XGBoost dikenalkan pda *Higgs Boson Competition*, yang mana pada kompetisi tersebut XGBoost menjadi salah satu metode yang banyak digunakan oleh sebagian besar dari tim. Selain pada kompetisi tersebut, metode tersebut juga mejadi metode yang banyak digunakan di kompetisi *machine learning* yang diselenggarakan Kaggle tahun 2015. XGBoost adalah salah satu *tree ensemble algorithm* yang terdiri dari *classification and Regression trees* (CART). Algoritma XGBoost ini dapat melakukan optimasi 10 kali lebih cepat dibandingkan GBM lainnya (Chen & Guestrin, 2016).

Nilai akurasi dari hasil klasifikasi menggunakan metode XGBoost tergantung parameter-parameter yang akan digunakan. Berikut merupakan parameter yang dapat disesuaikan nilainya pada XGBoost supaya mendapatkan nilai akurasi yang baik:

Tabel 2. 1 Parameter pada *Xtreme Gradient Boosting*

Parameter	Keterangan
Eta	*Learning rute* proses pelatihan
Gamma	Parameter *penalty* dari *regularization*
Max_depth	Tingkat kedalaman suatu pohon, semakin dalam pohon maka akan semakin kompleks
Min_child_weight	Nilai minimal bobot yang dibutuhkan *child node*
Subsample	Jumlah sampel yang digunakan proses pelatihan. Misalnya 0,5, artinya menggunakan setengah dari data secara acak dalam pembuatan pohon baru
Colsample_bytree	Jumlah sampel pada kolom untuk membuat tree baru.

2.14. *Confussion Matrix*

Pengukuran kinerja pada sistem klasifikasi merupakan hal yang penting dalam menggambarkan seberapa baik sistem klasifikasi yang digunakan. Salah satu metode yang digunakan dalam mengukur kinerja sistem klasifikasi yaitu *confusion matrix*. *Confussion matrix* adalah salah satu alat ukur yang berbentuk *matrix* untuk memperoleh nilai ketepatan sistem klasifikasi terhadap kelas sesuai dengan algoritma yang digunakan. Tujuan digunakannya *confussion matrix* adalah untuk mempermudah dalam mencari ukuran performa pada hasil klasifikasi. Berikut merupakan tabel contoh dari *confussion matrix* dengan 2 klasifikasi benar dan salah (Sofi & Jajuli, 2015):

Tabel 2. 2 *Confussion Matrix*

Confusion Matrix		**Nilai Sebenarnya**	
		TRUE	***FALSE***
Nilai Prediksi	***TRUE***	*TP (True Positive)* *Correct result*	*FP (False Positive)* *Unexpected result*
	FALSE	*FN (False Negative)* *Missing result*	*TN (True Negative)* *Correct absence of result*

Confussion matrix mengandung nilai *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) serta *False Negative* (FN). Untuk nilai TP dan TN memberikan informasi *classifer* dalam melakukan klasifikasi data yang bernilai benar. Sedangkan untuk FP dan FN memberikan informasi *classifer* salah dalam melakukan klasifikasi data. Berdasarkan hasil dari *confussion matrix* yang diperoleh dapat digunakan untuk mengukur nilai *Accuracy*, *Precision* serta *Recall*. *Accuracy* yaitu suatu nilai untuk mengevaluasi hasil dari model klasifikasi dan merupakan nilai kedekatan antara nilai prediksi dan aktual. *Precision* yaitu ketelitian atau ketepatan dalam melakukan klasifikasi. Sedangkan *Recall* yaitu proporsi nilai positif aktual yang benar didefinisikan (Sasongko, 2016). Nilai *Specificity* dan *Sensitivity* digunakan untuk melihat ukuran statistic dari klasifikasi biner, mengukur model terbaik dan untuk memilih model yang efisien. Nilai *Sensitivity* merupakan nilai untuk mengukur proporsi nilai positif yang benar diidentifikasi, sedangkan nilai *Specificity* merupakan nilai untuk mengukur proporsi nilai negative yang benar diidentifikasi (Bramer, 2013).

Berikut merupakan rumus untuk mengitung nilai *Accuracy*, *Precision*, *Recall/Sensitivity*, *Specificity*, FPR dan AUC:

$$Precision = \frac{TP}{TP+FP}\ x\ 100\% \quad (2.24)$$

$$Recall/Sensitivity = \frac{TP}{TP+FN}\ x\ 100\% \quad (2.25)$$

$$Specificity = \frac{TN}{TN+FP}\ x\ 100\% \quad (2.26)$$

$$FPR = 1 - Specificity \quad (2.27)$$

$$AUC = \frac{1+sensitivity-FPR}{2} \quad (2.28)$$

Nilai *Area Under Curve* (AUC) berguna untuk pengukuran kinerja deskriminatif dengan memperkirakan propabilitas *output* sampel yang dipilih secara acak dari probabilitas positif dan negatif. AUC memiliki rentang nilai 0-1, semakin tinggi atau semakin besar nilai AUC makan klasifikasi dapat dikatakan baik. Tabel berikut merupakan penerapan klasifikasi AUC:

Tabel 2. 3 Nilai AUC

Nilai AUC	Keterangan
0,91 – 1	Klasifikasi sangat baik
0,81 – 0,90	Klasifikasi baik
0,71 – 0,80	Klasifikasi cukup
0,61 – 0,70	Klasifikasi buruk
≤ 0,60	Klasifikasi salah

Sumber : Gorunescu (2011)