IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Tanggal Publikasi 19/12/2021

Klasifikasi Diabetes Menggunakan *Supervised Learning* Dengan Algoritma *Artificial Neural Network*

Abdurrachman Farras - 119140052[1], Dhifaf Athiyah Zhabiyan - 119140047[2], and Eliza Maharani Sutowo - 119140002[3]
Program Studi Teknik Informatika, Jurusan Teknologi Produksi dan Informasi, Institut Teknologi Sumatera
abdurrachman.119140052@student.itera.ac.id, dhifaf.119140047@student.itera.ac.id, eliza.119140002@student.itera.ac.id.

ABSTRACT Diabetes Melitus (DM) adalah sebuah penyakit yang muncul dan berkembang secara tahunan yang ditandai oleh kadar glukosa darah (gula darah) melebihi normal. Sebanyak 14,3 juta atau 73.7% orang di Indonesia yang mengidap penyakit diabetes, tidak menyadari bahwa dirinya terkena diabetes. Dengan memanfaatkan teknologi komputer yang dapat bekerja dengan meniru cara kerja otak manusia dan mengadaptasi kemampuan seorang pakar sehingga dapat mengambil keputusan berdasarkan informasi maupun pola yang diberikan yaitu dengan menggunakan jaringan syaraf tiruan atau Artificial Neural Network (ANN). Dalam penelitian kami menggunakan metode pemodelan data diabetes yaitu dengan memanfaatkan algoritma *Artificial Neural Networks* yang menggunakan *backpropagation* sebagai algoritma untuk analisa data dan meminimalisir tingkat error pada output yang diberikan. Dalam dataset *Early Stage Diabetes Risk Prediction* yang digunakan terdapat informasi sebanyak 520 data dari subjek perempuan dan laki-laki dengan rentang usia dari 16 sampai 90 tahun. Pada pembuatan struktur ANN atau *Artificial Neural Network* kami membuat jaringan dengan susunan lapisan yang terdiri dari 1 *input layer,* 1 *hidden layer* dan 1 *output layer*. Dari hasil keseluruhan pengujian dan evaluasi didapatkan hasil bahwa model dengan *node* dari *hidden layer* berjumlah 25 dengan *learning rate* 0.1 serta *epoch* 25000 mampu memberikan hasil terbaik dengan nilai akurasi 0.99706 pada data *training* dan 0.99441 pada data *test* dengan kesalahan prediksi yang terbilang minim.

KATA KUNCI Artificial intelligence, Artificial neural networks, Multi-layer neural network, dan Supervised learning.

I. PENDAHULUAN

Diabetes atau bisa disebut juga Diabetes Melitus (DM) adalah sebuah penyakit yang muncul dan berkembang secara tahunan yang ditandai oleh kadar glukosa darah (gula darah) melebihi normal yaitu kadar gula darah sewaktu sama atau lebih dari 200 mg/dl, dan kadar gula darah puasa di atas atau sama dengan 126 mg/dl [1].

Dalam dunia kontemporer saat ini, diabetes sudah menjadi penyakit yang sangat lazim terjadi di dunia. Menurut data dari IDF (*International Diabetes Federation)* kematian yang diakibatkan oleh diabetes pada tahun 2021 di dunia sendiri sudah mencapai angka 6,7 juta kematian pada rentang usia 20-79 tahun. Jumlah angka kematian tersebut menjadi salah satu penyumbang terbesar dari total angka kematian dunia pada tahun 2021 dengan rentang usia yang sama yaitu sebesar 12,2%. Di Asia Tenggara sendiri angka kematian akibat diabetes mencapai 747.000 kematian pada tahun 2021 [2].

Berdasarkan data IDF pada laporan tahunan edisi ke-10 yang tersedia di https://diabetesatlas.org/ , 3 dari 4 orang dewasa yang mengidap diabetes tinggal di negara dengan tingkat pendapatan menengah ataupun rendah. Indonesia sendiri yang merupakan negara berkembang dengan tingkat pendapatan menengah menjadi negara dengan jumlah pengidap diabetes terbesar pada urutan ke-5 di tahun 2021. Sebanyak 19,5 juta orang di Indonesia terkena penyakit diabetes ini [2].

Tak cukup sampai disitu saja diabetes yang merupakan *silent killer* karena sering tidak disadari oleh penyandangnya dan dapat menyerang hampir seluruh sistem tubuh manusia, mulai dari kulit sampai jantung yang menimbulkan komplikasi [1]. Sebanyak 14,3 juta atau 73.7% orang di Indonesia yang mengidap penyakit diabetes, tidak menyadari bahwa dirinya terkena diabetes.

Hal tersebut memang disebabkan oleh berbagai faktor baik genetik, ekonomi, dan layanan kesehatan di Indonesia [2].

Untuk menyelamatkan nyawa manusia, diperlukan deteksi dan diagnosis dini dari penyakit diabetes ini. Dengan dibantu oleh kemajuan teknologi di zaman sekarang yang memberikan banyak manfaat baik bagi kehidupan manusia. Salah satunya adalah teknologi komputer dengan sistem yang dapat bekerja dengan meniru cara kerja otak manusia dan mengadaptasi kemampuan seorang pakar sehingga dapat mengambil keputusan berdasarkan informasi maupun pola yang diberikan yaitu dengan menggunakan jaringan syaraf tiruan atau *Artificial Neural Network* (ANN).

Algoritma jaringan syaraf tiruan (ANN) adalah algoritma yang menggunakan model terlatih dan memiliki kemampuan yang sangat baik untuk menyelesaikan masalah non-linier melalui pengenalan pola, yang dapat dimodelkan. Selain kelebihannya, neural network memiliki satu kelemahan yaitu masalah penentuan parameter, sehingga perlu dilakukan eksperimen untuk menentukan setiap parameter [3]. Arsitektur ANN akan dibangun dan dilatih dengan menggunakan data-data yang diperoleh dari rekam medis orang orang yang baru mengidap diabetes dan juga orang orang yang kemungkinan akan mengidap diabetes. Data ini diambil dari situs kaggle yakni, *Early Stage Diabetes Risk Prediction Dataset*. Dan penelitian ini bertujuan untuk mengklasifikasikan apakah seseorang mengidap diabetes berdasar 16 fitur medis yang diberikan dengan menggunakan arsitektur ANN yang terdiri dari 1 *input layer*, 1 *hidden layer,* dan 1 *output layer*.

Jurnal ini disusun sebagai berikut : Bagian I berisi pendahuluan jurnal, Bagian II berisi penjelasan singkat tentang dasar teori yang berkaitan dengan topik jurnal, Bagian III berisi tentang metodologi penelitian yang digunakan, lalu pada Bagian IV berisi tentang hasil penelitian disertai pembahasannya, Bagian V berisi kesimpulan dari penelitian yang telah dilakukan, terdapat bagian daftar pustaka yang berisi daftar referensi yang digunakan dalam penelitian ini.

II. DASAR TEORI

A. Diabetes Melitus

Diabetes Melitus merupakan sebuah penyakit yang terjadinya ditandai dengan gangguan metabolisme karbohidrat, lemak dan protein yang dihubungkan dengan kurangnya kinerja dan atau sekresi insulin. Secara singkatnya diabetes melitus merupakan penyakit yang disebabkan dengan adanya kekurangan insulin secara relatif maupun absolut. Adapun beberapa faktor resiko yang berkaitan dengan kemungkinan terjadinya diabetes melitus yaitu sebagai berikut :

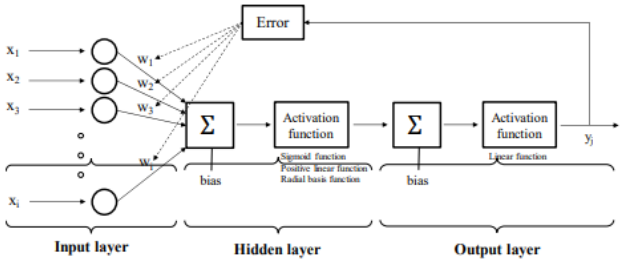
1. Memiliki kadar gula dalam darah lebih dari 200 mg%
2. Memiliki hipertensi
3. Memiliki riwayat keluarga Diabetes Melitus
4. Memiliki kenaikan kadar lemak (*Trigliserida*) > 250 mg/dl
5. Memiliki umur diatas 45 Tahun
6. Faktor genetik
7. Mengkonsumsi alkohol dan rokok

Biasanya penderita Diabetes Melitus memiliki gejala akut seperti *Polyphagia* (banyak makan), *Polydipsia* (banyak minum), *Polyuria* (sering buang air kecil), nafsu makan bertambah namun berat badan turun dengan cepat dan mudah lelah. Selain gejala akut terdapat juga gejala kronik seperti kesemutan, kebas, pandangan mulai kabur, gigi yang mudah lepas, dan terjadinya keguguran janin pada ibu hamil.

Diagnosis Diabetes Melitus dinyatakan saat terjadinya keluhan, gejala, serta hasil gula dalam darah > 200 mg/dl ataupun gula dalam darah saat puasa > 126 mg/dl [4].

B. Artificial Neural Network

Artificial Neural Network (ANN) merupakan jaringan yang didasari oleh kumpulan Neuron yang terhubung dan dapat dimodelkan dalam pemodelan kompleks [5]. ANN memiliki struktur dasar yang dimana terdiri dari *input, hidden*, dan *output*. setiap neuron yang menerima *input* maka maka nilai tersebut akan dikirimkan ke pada neuron yang terhubung pada jaringan-jaringan berikut nya.



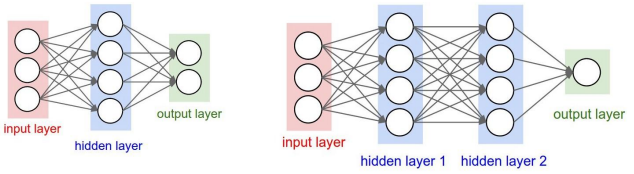
Gambar 1. Sinematik diagram ANN [5]

Data inputan berasal dari dataset yang di masukan, dataset di analisis. informasi dianalisis dan dilewatkan melalui lapisan tersembunyi yang mengandung neuron tersembunyi, melalui fungsi transfer. Dalam lapisan tersembunyi, data dienkripsi, dianalisis, dan dihitung melalui bobot. *Offset* juga diperkirakan untuk memastikan throughput data yang seimbang. Akhirnya, hasilnya dihitung pada lapisan keluaran [6].

1. WEIGHTS DAN BIAS

Setiap koneksi memiliki *weight* yang mana nilai tiap *weight* akan berbeda beda. Neuron pada setiap *Multi Layer*

Perceptron (MLP) saling terhubung dengan ciri khas tanda panah seperti gambar berikut [7].



Gambar 2. Sinematik diagram *Multi Layer Perceptron* [7]

Hidden layer dan *output layer* memiliki masukan yang disebut dengan *bias*.

2. ACTIVATION FUNCTION

Activation function berfungsi untuk menentukan keaktifan dari sebuah neuron berdasarkan jumlah *weight input*. *Activation function* memiliki *neuron* pada *hidden layer* dan *output layer* yang bergantung dari setiap data atau masalah yang ingin diselesaikan. *Activation function* berdasarkan sebuah neuron merupakan *linear*. Apabila sebuah *neuron* memakai *linear function*, maka keluaran berdasarkan neuron tadi merupakan jumlah *weighted* berdasarkan *input* dengan bias [8].

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

Gambar 3. Rumus *Activation function*

Sigmoid function memiliki rentang antara 0 sampai 1.

3. BACKWARD PROPAGATION

Proses *backpropagation* dibuat untuk menyesuaikan tiap bias dan *wight* berdasarkan *error*. Tahapan dari *backprop* diawali dengan menghitung *gradient* dari *loss function* terhadap semua parameter kemudian memperbaharui seluruh parameter menggunakan *Stochastic Gradient Descent (SGD)* menggunakan mengurangi atau menambahkan nilai weight usang [7]. Penerapan dengan menggunakan momentum akan menerima *local error minimum* [9].

$$\Delta w_{i,j}(n) = \eta \delta_j x_{i,j} + \alpha \Delta w_{i,j}(n-1)$$

Gambar 4. Rumus momentum local *error minimum*

Terdapat juga *backward prop output unit*,

$$\delta_j = o_j(1-o_j)\sum_k w_{jk} \cdot \delta_k$$

Gambar 5. Rumus *backward prop output* [9]

dan juga *hidden unit*.

$$\delta_k = (y_{target,k} - y_k) y_k (1-y_k)$$

Gambar 6. Rumus *Hidden Unit*

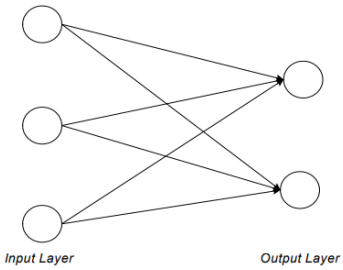
serta menggunakan *Backward Prop Update Weigh*t untuk untuk pembaruan.

$$w_{i,j}(n+1) = w_{i,j}(n) + \Delta w_{i,j}(n)$$

Gambar 7. Rumus *Backward Prop Update Weigh*t

4. FORWARD PROPAGATION

Forward Propagation memiliki fungsi untuk memilih *output* dari *node*. *Output* yang dipaparkan adalah *output layer*. Setiap *Input layer* memiliki tujuan *output* yang bisa lebih dari satu. *Output layer* menampung hasil dari proses pengolahan fitur pada *neural network* menggunakan *Forward Propagation* untuk mencari error pada *output layer* [10].



Gambar 8. Sinematik *input output layer* [10]

Untuk melakukan *forward propagation* diperlukan transformasi input data yang masuk dengan menggunakan persamaan berikut,

$$z = W_{i,j} p + b$$

Gambar 9. Rumus *forward propagation*

Kemudian nilai z yang diperoleh tersebut akan dimasukkan kedalam fungsi aktivasi yang digunakan.

C. Supervised Learning

Untuk mengolah data diperlukan metode pengolahan data yang dapat mengekstraksi informasi. Salah satu contoh dari metode ini adalah metode berbasis grafik, dimana ukuran-ukuran kesamaan tiap lokal membangun garis di setiap titik data. Oleh karena itu data yang berdimensi tinggi dapat mendapatkan jarak fitur *Euclidean* yang baik pada setiap indikator antar titik data [11]. Untuk itu digunakan *supervised learning* untuk mengolah dan

pemetaan data di antara data *input-output*. Dalam hal ini, kumpulan data diberi label, yang berarti bahwa algoritma secara jelas mengidentifikasi fitur dan membuat prediksi atau klasifikasi yang sesuai [12].

III. METODOLOGI PENELITIAN

Metode yang digunakan dalam pemodelan data diabetes yaitu dengan algoritma *Artificial Neural Networks* yang menggunakan *backpropagation* sebagai algoritma untuk analisa data dan meminimalisir tingkat error pada output yang diberikan. Adapun kriteria dalam penelitian ini yaitu berfokus pada akurasi pengklasifikasian dan pemodelan data.

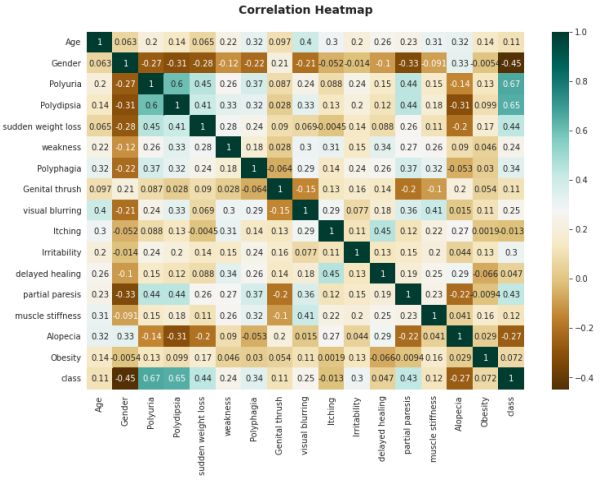
A. Data

Dalam penelitian jurnal ini kami menggunakan dataset publik yaitu Pima Indian Database yang didapatkan dari Ishan Dutta Dataset pada platform kaggle [13]. Dalam dataset tersebut diberikan informasi sebanyak 520 data dari subjek perempuan dan laki-laki dengan rentang usia dari 16 sampai 90 tahun. Pada data tersebut terdiri dari 16 atribut dan 1 label atau *class*. 16 Atribut tersebut terdiri dari *Age, Gender, Polyuria, Polydipsia, Sudden weight loss, Weakness, Polyphagia, Genital thrush, Visual blurring, Itching, Irritability, Delayed healing, Partial paresis, Muscle stiffness, Alopecia,* dan *Obesity.* Berikut merupakan tabel deskripsi terkait atribut yang ada di dalam dataset tersebut :

TABEL I
DESKRIPSI ATRIBUT DATASET

Atribut	Deskripsi Atribut	Persebaran
Age	Usia subjek dalam tahun	16 - 90 Tahun
Gender	Jenis kelamin subjek	Perempuan : 192 Laki - laki : 327
Polyuria	Subjek sering buang air kecil	Iya : 262 Tidak : 258
Polydipsia	Subjek sering atau banyak minum	Iya : 287 Tidak : 233
Sudden Weight Loss	Subjek tiba-tiba kehilangan berat badan	Iya : 303 Tidak : 217
Weakness	Subjek merasakan lemas atau lelah	Iya : 215 Tidak : 305
Polyphagia	Subjek sering atau banyak makan	Iya : 283 Tidak : 237
Genital Thrush	Subjek mengalami sariawan kelamin	Iya : 404 Tidak : 116
Visual blurring	Subjek mengalami pengaburan	Iya : 287 Tidak : 233
Itching	Subjek mengalami gatal - gatal	Iya : 267 Tidak : 253
Irritability	Subjek mengalami iritasi	Iya : 394 Tidak : 126
Delayed healing	Subjek mengalami penyembuhan yang lama	Iya : 281 Tidak : 239
Partial paresis	Subjek mengalami lumpuh sebagian	Iya : 296 Tidak : 224
Muscle stiffness	Subjek mengalami kekakuan otot	Iya : 325 Tidak : 195
Alopecia	Subjek mengalami kebotakan atau kerontokan	Iya : 341 Tidak : 179
Obesity	Subjek mengalami obesitas	Iya : 432 Tidak : 88

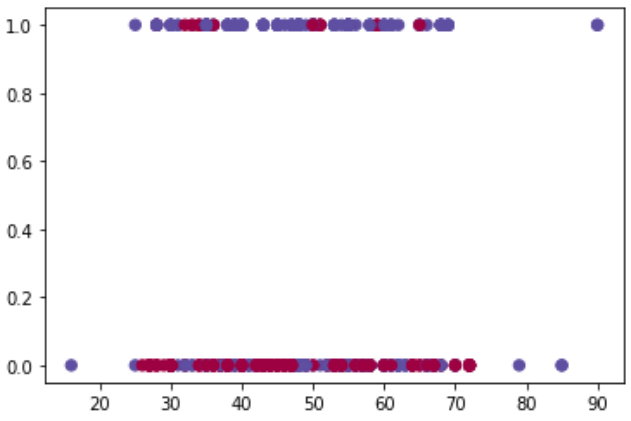
Dari data yang telah dikumpulkan terdapat korelasi antara setiap atribut yang dapat dilihat dari matriks korelasi sebagai berikut :



Gambar 10. Matriks korelasi antar setiap atribut

Selain korelasi antar setiap atribut, terdapat juga hubungan antara atribut dengan *class* yang bernilai "negative" atau "0" dan "positive" atau "1" sebagai berikut

:



Gambar 11. Matriks korelasi antara atribut dengan *class*

B. *Preprocessing Data*

Tahap *preprocessing* data dilakukan sebelum adanya kegiatan pemodelan data. Berikut merupakan hal yang dilakukan dalam tahap *preprocessing* data yaitu :

1. STANDARISASI USIA

Pada atribut usia, terdapat banyak ragam data sehingga dibutuhkan standarisasi data usia yang dapat mempercepat pelatihan data dan meningkatkan nilai akurasi dari model dalam melakukan klasifikasi. Berikut merupakan rumus *z-score* yang digunakan dalam standarisasi usia yaitu sebagai berikut :

$$Z = (Yi - \overline{Y}) / SD \text{ [14]}$$

Keterangan :
Z : Z-score
Yi : Skor mentah
$\overline{Y}$: Mean awal
SD : Standar deviasi

	Age
340	-0.743025
341	-0.413847
342	-1.072204
343	-0.084668
344	1.067456

Gambar 12. Hasil standarisasi usia

2. MENGUBAH DATA KE DATA BINER

Selain atribut usia, atribut lainnya memiliki nilai data "yes" dan "no" oleh sebab itu dibutuhkan penyeragaman bentuk data agar mudah diolah oleh komputer yaitu dalam bentuk binary. Pada penelitian ini data yang bernilai "yes" bernilai binary "1" dan "no" bernilai "0". Selain itu terdapat atribut *gender* dimana data yang bernilai *male* bernilai 1 dan data yang bernilai *female* bernilai 0. Terdapat juga *class* yang nilai datanya perlu diubah ke binary dimana *class* yang bernilai *positive* diberi nilai 1 dan *negative* diberi nilai 0.

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itching	Irritability	delayed healing	partial paresis	muscle stiffness	Alopecia	Obesity	class
340	-0.743025	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
341	-0.413847	0.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0
342	-1.072204	0.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
343	-0.084668	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
344	1.067456	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
515	-0.743025	0.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0
516	-0.002374	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
517	0.820572	0.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0
518	-1.319067	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0
519	-0.496141	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Gambar 13. Pengubahan data ke binary

3. MEMBAGI DATA MENJADI *TRAIN* DAN *TEST* DATA

Untuk melakukan proses validasi data dan juga pengujian dari model yang sudah dibuat maka dari itu diperlukan pemisahan data. Yaitu data-data yang akan digunakan untuk melakukan pelatihan model atau biasa disebut *train dataset,* dan juga data-data yang belum pernah disentuh oleh model serta akan menjadi bahan ujian oleh model yang hasil validasi atau prediksinya akan menjadi bahan evaluasi untuk menentukan apakah model *overfit* atau tidak yang biasa disebut dengan *test dataset.* Untuk pembagiannya sendiri kami menggunakan rasio 6,5:3,5 untuk *train* dan *test dataset* secara berurutan.

C. Pembuatan Struktur ANN

Pada pembuatan struktur ANN atau *Artificial Neural Network* kami membuat jaringan dengan beberapa lapisan seperti 1 *input layer,* 1 *hidden layer* dan 1 *output layer*. Berikut merupakan langkah - langkah dalam membangun struktur ANN yaitu sebagai berikut :

1. ATRIBUT UTAMA YANG DIGUNAKAN

Di dalam *class* ANN kita dapat mendefinisikan beberapa *function* yang akan dibutuhkan. Seperti pada bagian bawah ini yang merupakan pendefinisian nilai random untuk *weight* dan *bias layer* 1 dan juga *layer* 2.

```
def neuralAttribute(self):
    np.random.seed(1)
    self.parameters['W1'] = np.random.randn(self.dimentions[1], self.dimentions[0]) / np.sqrt(self.dimentions[0])
    self.parameters['b1'] = np.zeros((self.dimentions[1], 1))
    self.parameters['W2'] = np.random.randn(self.dimentions[2], self.dimentions[1]) / np.sqrt(self.dimentions[1])
    self.parameters['b2'] = np.zeros((self.dimentions[2], 1))
    return
```

Gambar 14. *Function weight and bias*

Pada function tersebut didefinisikan bahwa :
W1 : *dimentions[1]* menunjukkan jumlah baris yang tersembunyi pada *layer* tersebut dan *dimensions[0]* menunjukkan jumlah kolom pada *layer* sebelumnya.

b1 : menunjukkan jumlah baris sama dengan W1 dan satu kolom

W2 : *dimensions[2]* menunjukkan jumlah baris yang tersembunyi pada *layer* tersebut dan *dimentions[1]* menunjukkan jumlah kolom pada dari input ke *layer* tersebut.

b2 : menunjukkan jumlah baris yang sama dengan W2 dan satu kolom [15]

Selain itu terdapat *function forward propagation* yang digunakan sebagai berikut. Dalam penerapannya menggunakan rumus *forward propagation* untuk mengkalkulasi inputan yang akan dikirim kepada layer selanjutnya. Kemudian untuk menghitung setiap luaran yang dihasilkan oleh setiap *nodes* pada *layer* yang ada digunakan rumus sigmoid.

```
def forwardPropagation(self):
    Z1 = self.parameters['W1'].dot(self.inputX) + self.parameters['b1']
    A1 = Sigmoid(Z1)
    self.cache['Z1'],self.cache['A1']=Z1,A1

    Z2 = self.parameters['W2'].dot(A1) + self.parameters['b2']
    A2 = Sigmoid(Z2)
    self.cache['Z2'],self.cache['A2']=Z2,A2

    self.networkOutputY=A2
    loss=self.calculateLoss(A2)
    return self.networkOutputY, loss
```

Gambar 15. *Function forward propagation*

Terdapat juga *function backward propagation* yang diimplementasikan dengan melakukan perhitungan menggunakan rumus *backpropagation* untuk menghitung *local gradient* bagi *output*, dan *hidden layer.* Kemudian melakukan update beban serta bias yang disimpan dalam *dictionary* python.

```
def backwardPropagation(self):
    deltaLoss_networkOutputY = - (np.divide(self.outputY, self.networkOutputY ) - np.divide(1 - self.outputY, 1 - self.networkOutputY))

    deltaLoss_Z2 = deltaLoss_networkOutputY * derivativeSigmoid(self.cache['Z2'])
    deltaLoss_A1 = np.dot(self.parameters["W2"].T,deltaLoss_Z2)
    deltaLoss_W2 = 1./self.cache['A1'].shape[1] * np.dot(deltaLoss_Z2,self.cache['A1'].T)
    deltaLoss_b2 = 1./self.cache['A1'].shape[1] * np.dot(deltaLoss_Z2, np.ones([deltaLoss_Z2.shape[1],1]))

    deltaLoss_Z1 = deltaLoss_A1 * derivativeSigmoid(self.cache['Z1'])
    deltaLoss_A0 = np.dot(self.parameters["W1"].T,deltaLoss_Z1)
    deltaLoss_W1 = 1./self.inputX.shape[1] * np.dot(deltaLoss_Z1,self.inputX.T)
    deltaLoss_b1 = 1./self.inputX.shape[1] * np.dot(deltaLoss_Z1, np.ones([deltaLoss_Z1.shape[1],1]))

    self.parameters["W1"] = self.parameters["W1"] - self.learningRate * deltaLoss_W1
    self.parameters["b1"] = self.parameters["b1"] - self.learningRate * deltaLoss_b1
    self.parameters["W2"] = self.parameters["W2"] - self.learningRate * deltaLoss_W2
    self.parameters["b2"] = self.parameters["b2"] - self.learningRate * deltaLoss_b2

    return
```

Gambar 16. *Function backward propagation*

2. FUNGSI SIGMOID DAN DERIVATIVE SIGMOID

Diperlukan fungsi sigmoid dan derivative sigmoid untuk menghitung perhitungan nilai aktivasi, baik ketika melakukan *forward propagation* ataupun *backward propagation*. Fungsi tersebut diimplementasikan sebagai berikut :

```
def Sigmoid(Z):
    return 1/(1+np.exp(-Z))

def derivativeSigmoid(Z):
    sigma= 1/(1+np.exp(-Z))
    derivativeZ = sigma * (1-sigma)
    return derivativeZ
```

Gambar 17. *Function sigmoid* dan *derivative sigmoid*

3. MENGHITUNG *LOSS*

Selain menghitung luaran yang dihasilkan dari masukan yang diberikan pada setiap layer, pada proses *forward propagation* juga dilakukan perhitungan *loss*. Perhitungan *loss* sendiri akan memberitahukan seberapa jauh hasil yang diperoleh dari target yang seharusnya. Dalam kesempatan kali ini karena permasalahan yang dihadapi berupa klasifikasi data biner, yang mana luaran atau hasil klasifikasi akan bernilai 1 dan 0 (1 untuk positif diabetes dan 0 negatif diabetes). Maka akan digunakan fungsi perhitungan *loss* yang lain yaitu, *Cross-Entropy Loss Function* dan berikut merupakan implementasinya.

```
def calculateLoss(self,networkOutputY):
    loss = (1./self.sample) * (-np.dot(self.outputY,np.log(networkOutputY).T) - np.dot(1-self.outputY, np.log(1-networkOutputY).T))
    return loss
```

Gambar 18. Fungsi perhitungan *loss* dengan *Cross-Entropy Loss Function*

IV. HASIL DAN PEMBAHASAN

Percobaan penelitian kali ini dilakukan menggunakan Google Colab (https://colab.research.google.com), dengan kapasitas ram sebesar 12 GB penyimpanan 100 GB, serta tidak mengaktifkan fitur GPU-nya. Adapun percobaan yang dilakukan terdiri dari beberapa tahapan yang bisa dilihat dari tabel berikut.

TABEL II
TAHAPAN PERCOBAAN

Eksperimen	Yang dilakukan	Output yang didapatkan
1	Menyusun model dengan konfigurasi seperti yang direncanakan serta melakukan perubahan-perubahan fungsi yang dipakai.	Mengetahui cara pengimplementasian dalam membangun model ANN yang memberikan performa baik.
2	Melakukan uji variasi nilai learning rate, dan epoch untuk mencari performa model terbaik, dengan menggunakan jumlah dataset yang sudah ditentukan.	Mengetahui nilai dari masing-masing hyperparameter yang memberikan performa model terbaik.

3	Melakukan uji variasi banyak *node* dalam *hidden layer* untuk mencari model dengan performa terbaik.	Mengetahui jumlah *node* dari *hidden layer* yang memberikan performa model terbaik.
4	Eksperimen terakhir yang dilakukan dengan menggunakan konfigurasi arsitektur model terbaik berdasarkan keseluruhan eksperimen yang telah dilakukan.	Mendapatkan model dengan performa terbaik dari keseluruhan eksperimen yang telah dilakukan.

A. Hasil Percobaan

Setelah melakukan eksperimen-eksperimen yang telah direncanakan maka didapatkan beberapa hasil yang dirangkum dalam beberapa tabel dan graf hasil setiap training yang sukses dilakukan. Setiap rangkuman tabel merupakan gabungan dari beberapa pengaturan *learning rate, epoch, train accuracy, validation accuracy* dan juga waktu dalam satuan sekon yang diperlukan untuk melakukan pelatihan pada model. Dan terdapat tiga tabel yang menunjukkan percobaan dilakukan dengan menggunakan jumlah *node* pada *hidden layer* yang berbeda yaitu 10, 25, dan 25 secara berurutan.

Dengan menggunakan konfigurasi jumlah *node* pada *hidden layer* sebanyak 10 buah, diperoleh data pada tabel dibawah ini. Akurasi dari data *train* tertinggi sebesar 0.99706 dihasilkan dari model ANN dengan konfigurasi *learning rate* = 0.1 dan *epoch* = 25,000 serta *epoch* = 50,000. Kemudian untuk akurasi pada data *test* nilai tertinggi yaitu sebesar 0.99441, diperoleh dengan konfigurasi *learning rate* = 0.1 dan *epoch* = 25,000. Rentang waktunya sendiri adalah 2.785 - 16.833 sekon untuk melakukan pelatihan secara sempurna.

TABEL III
PERCOBAAN NODE PADA HIDDEN LAYER 10 BUAH

Learning Rate	Epoch	Train Accuracy	Validation Accuracy	Time Needed
0.001	10000	0.69412	0.46369	2.785
	25000	0.79706	0.67598	7.6
	50000	0.91176	0.89385	15.096
0.01	10000	0.91176	0.91061	3.387
	25000	0.92941	0.95531	7.911
	50000	0.93529	0.94413	16.833
0.1	10000	0.97059	0.96648	3.305
	25000	0.99706	0.99441	7.343
	50000	0.99706	0.98883	14.366

Selanjutnya adalah dengan menggunakan konfigurasi jumlah *node* pada *hidden layer* sebanyak 25 buah, diperoleh data pada tabel dibawah ini. Akurasi dari data *train* tertinggi sebesar 0.99706 dihasilkan dari model ANN dengan konfigurasi *learning rate* = 0.1 dan *epoch* = 25,000 serta *epoch* = 50,000. Kemudian untuk akurasi pada data *test* nilai tertinggi yaitu sebesar 0.99441, diperoleh dengan konfigurasi *learning rate* = 0.1 dan *epoch* = 25,000 serta *epoch* = 50,000. Rentang waktunya sendiri adalah 4.866 - 27.382 sekon untuk melakukan pelatihan secara sempurna.

TABEL IV
PERCOBAAN NODE PADA HIDDEN LAYER 25 BUAH

Learning Rate	Epoch	Train Accuracy	Validation Accuracy	Time Needed
0.001	10000	0.70588	0.48045	4.866
	25000	0.87059	0.77095	13.18
	50000	0.90882	0.90503	26.604
0.01	10000	0.91765	0.92737	5.658
	25000	0.93529	0.94972	12.856
	50000	0.94118	0.93855	25.427
0.1	10000	0.98824	0.97765	6.19
	25000	0.99706	0.99441	15.337
	50000	0.99706	0.99441	27.382

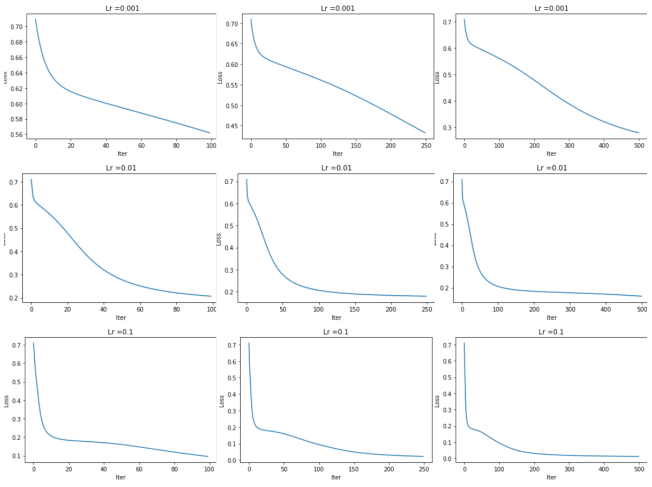
Terakhir dengan menggunakan konfigurasi jumlah *node* pada *hidden layer* sebanyak 50 buah, diperoleh data pada tabel dibawah ini. Akurasi dari data *train* tertinggi sebesar 0.99706 dihasilkan dari model ANN dengan konfigurasi *learning rate* = 0.1 dan *epoch* = 25,000 serta *epoch* = 50,000. Kemudian untuk akurasi pada data *test* nilai tertinggi yaitu sebesar 0.99441, diperoleh dengan konfigurasi *learning rate* = 0.1 dan *epoch* = 25,000 serta *epoch* = 50,000. Rentang waktunya sendiri adalah 17.347 - 97.496 sekon untuk melakukan pelatihan secara sempurna.

TABEL V
PERCOBAAN NODE PADA HIDDEN LAYER 50 BUAH

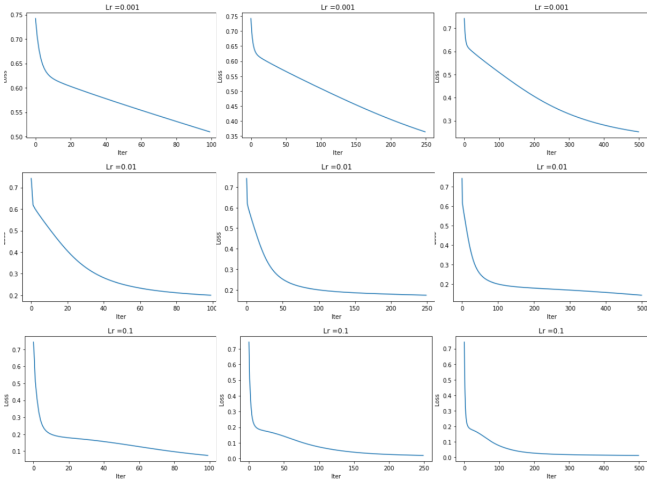
Learning Rate	Epoch	Train Accuracy	Validation Accuracy	Time Needed
0.001	10000	0.69412	0.46369	17.347
	25000	0.87647	0.81006	36.903
	50000	0.90588	0.91061	71.987
0.01	10000	0.91765	0.92737	20.672
	25000	0.93529	0.94972	53.554

	50000	0.95294	0.94413	97.714
0.1	10000	0.97941	0.97207	20.628
	25000	0.99706	0.99441	51.049
	50000	0.99706	0.99441	97.496

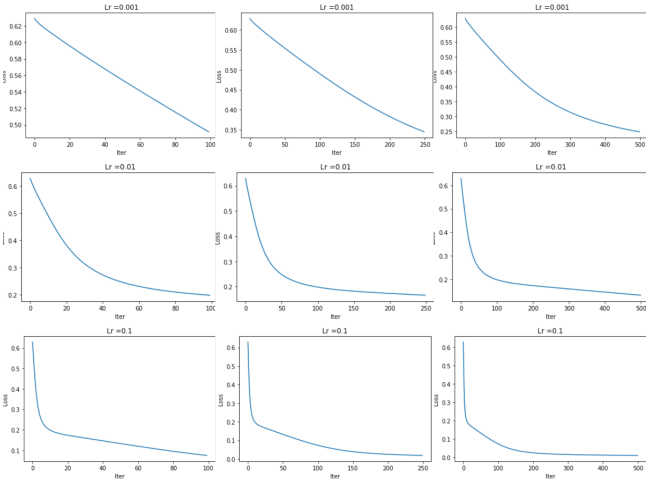
Adapun untuk melihat bagaimana perubahan atau penurunan nilai loss yang terjadi pada setiap *epoch* atau iterasi yang dilakukan dengan masing-masing konfigurasi dari setiap percobaan yang dilakukan kami tampilkan dengan grafik garis yang berhasil diperoleh di bawah ini. Terlihat penurunan *loss* yang cukup signifikan serta sangat mendekati 0 terjadi ketika *learning rate* = 0,1 dan juga jumlah *epoch* = 25,000 maupun *epoch* = 50,000. Grafik-grafik dibawah juga ditampilkan dan dikelompokkan berdasarkan banyaknya jumlah *node* pada *hidden layer,* 10, 25, dan 50 secara berurutan.



Gambar 19. Grafik percobaan node ketika hidden layer sama dengan 10.



Gambar 20. Grafik percobaan node ketika hidden layer sama dengan 25.



Gambar 21. Grafik percobaan node ketika hidden layer sama dengan 50.

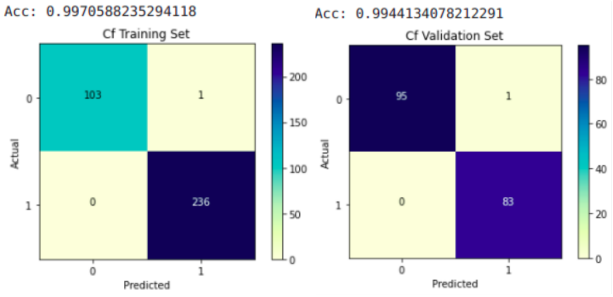
Dengan pertimbangan-pertimbangan dari hasil percobaan diatas maka akan diambil model dengan konfigurasi *node* pada *hidden layer* sebanyak 25 buah, dengan pertimbangan waktu yang diperlukan tidak terlalu lama yaitu 15.337 sekon dan *node* pada *hidden layer* yang dinilai tidak terlalu banyak ataupun terlalu sedikit untuk memproses setiap fitur yang masuk dari *input layer*. Kemudian *epoch* yang dipilih adalah sebanyak 25,000 *epoch* karena tingkat akurasi yang dihasilkan baik itu akurasi dengan data latih maupun data tes antara 25,000 dan 50,000 *epoch,* tidaklah jauh berbeda sehingga lebih baik memilih yang memiliki waktu pelatihan yang lebih singkat. Selanjutnya model dengan konfigurasi tersebut akan dilatih dan dilanjutkan untuk memasuki proses evaluasi.

B. Evaluasi

Pada proses evaluasi ini digunakan metode *confusion matrix* karena jenis permasalahan yang dipecahkan dengan menggunakan arsitektur ANN ini adalah permasalahan klasifikasi. Pada proses evaluasi ini model akan diuji untuk melakukan prediksi terhadap data tes yang sama sekali belum pernah diberikan pada model pada saat proses pelatihan. Kemudian pada proses evaluasi ini juga akan dilakukan perubahan pada nilai *threshold* pada saat model akan berusaha memprediksi data yang diberikan. *Threshold* ini adalah seberapa dekat nilai *output network* dengan angka 1 sehingga data fitur yang masuk bisa dikategorikan positif diabetes. Nilai *output network* yang dimaksud adalah nilai hasil prediksi (nilai yang dihasilkan ketika nilai-nilai dari fitur yang ada dimasukkan ke dalam arsitektur ANN yang sudah dibentuk dengan *weights* serta *bias* yang sudah dilatih). Nilai *threshold* ini juga dapat menunjukkan

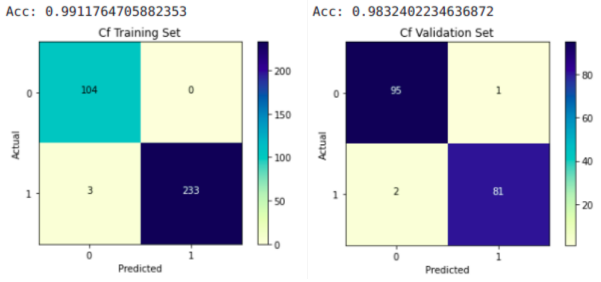
seberapa besar kepercayaan dari model yang berhasil dibuat.

Adapun dari model yang dilatih dengan konfigurasi yang sudah dipilih sebelumnya ketika nilai *threshold* = 0.5 (merupakan pengaturan bawaan), diperoleh akurasi diatas 0.99 baik pada data *training* maupun data *test.* Dan apabila dilihat pada *confusion matrix* yang ada hanya terjadi 1 buah kesalahan prediksi atau klasifikasi.



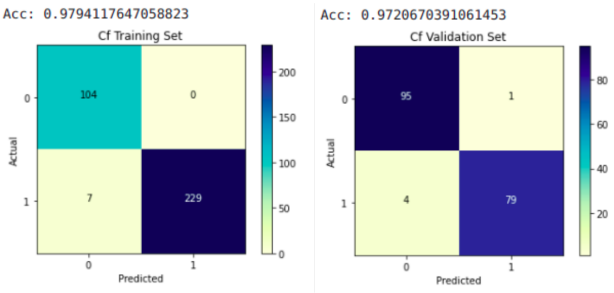
Gambar 22. *Confusion matrix* ketika nilai *threshold* sama dengan 0.5.

Kemudian selanjutnya adalah ketika nilai *threshold* diubah menjadi 0.7, diperoleh hasil yang masih baik dengan total kesalahan sebanyak 3 buah baik pada data *training* maupun data *test.*



Gambar 23. *Confusion matrix* ketika nilai *threshold* sama dengan 0.7.

Kemudian selanjutnya adalah ketika nilai *threshold* diubah menjadi 0.9, diperoleh hasil yang masih cukup baik dengan total kesalahan sebanyak 7 buah baik pada data *training* dan sebanyak 5 buah pada data *test.*



Gambar 24. *Confusion matrix* ketika nilai *threshold* sama dengan 0.9.

Dari hasil evaluasi, model yang dihasilkan mampu memberikan performa yang baik dengan akurasi yang masih berada diatas 95% dengan kesalahan prediksi yang terbilang minim pada saat diuji dengan data tes yang belum pernah diberikan sama sekali kepada model saat pelatihan. Walaupun hanya menggunakan arsitektur ANN sederhana, data yang terbatas, dan juga fitur yang cukup banyak secara keseluruhan model masih mampu memberikan hasil yang sangat baik.

V. KESIMPULAN

Penelitian kali ini menggunakan *Early Stage Diabetes Risk Prediction Dataset* yang terdiri dari 520 data disertai dengan 16 atribut dan 1 label *class*. Dari data tersebut dilakukan pengkalisifikasian diabetes menggunakan algoritma *Artificial Neural Network*. Sebelum melakukan pembuat struktur ANN, dilakukan tahap persiapan dataset sebelum diproses dalam struktur ANN. Pada penelitian kali ini kami menggunakan rasio 6,5 : 3,5 untuk pembagian antara *train* dan *dataset*. Setelah melakukan beberapa percobaan untuk mencari *hyperparameter* yang sekiranya memberikan hasil *optimum* kami menggunakan konfigurasi akhir model yaitu 25 node pada *hidden layer*, *learning rate* 0.1 dan *epoch* 25000. Pada pengujian dengan nilai *threshold* 0.5 diperoleh nilai akurasi 0.99706 pada data *training* dan 0.9944 pada data *test* serta hanya mengalami 1 kesalahan prediksi atau klasifikasi. Lalu pada pengujian dengan nilai *threshold* 0.7 diperoleh akurasi menjadi 0.99 pada data *training* dana 0.98 pada data *test* dengan 3 kesalahan klasifikasi pada data *training* dan data *test*. Selanjutnya pada pengujian dengan nilai *threshold* 0.9 dihasilkan nilai akurasi diatas 0.97 untuk kedua data dengan 7 kesalahan prediksi pada data *training* dan 5 buah kesalahan pada data *test.* Dari hasil keseluruhan pengujian dan evaluasi didapatkan hasil bahwa model dengan *node* berjumlah 25 dengan *learning rate* 0.1 dan *epoch* 25000 mampu memberikan nilai akurasi 0.99706 pada data *training* dan 0.99441 pada data *test* dengan kesalahan prediksi yang terbilang minim.

DAFTAR PUSTAKA

[1] Hestiana, D. W. (2017). FAKTOR-FAKTOR YANG BERHUBUNGAN DENGAN KEPATUHAN DALAM PENGELOLAAN DIET PADA PASIEN RAWAT JALAN DIABETES MELLITUS TIPE 2 DI KOTA SEMARANG. In *JHE* (Vol. 2, Issue 2). http://journal.unnes.ac.id/sju/index.php/jhealthedu/

[2] International Diabetes Federation, Diabetes around the world in 2021, 2021, Tersedia di : IDF Diabetes Atlas | Tenth Edition

[3] Somantri, O., & Supriyanto, C. (2016). Algoritme Genetika untuk Peningkatan Prediksi Kebutuhan Permintaan Energi Listrik. *Jurnal Nasional Teknik Elektro Dan Teknologi Informasi (JNTETI)*, *5*(2). https://doi.org/10.22146/jnteti.v5i2.233

[4] Fatimah, R. N. (2015). DIABETES MELITUS TIPE 2. In *J MAJORITY* | (Vol. 4). https://juke.kedokteran.unila.ac.id/index.php/majority/article/view/615/619

[5] J. Y. Yoon, H. Kim, Y. J. Lee, and S. H. Sim, "Prediction model for mechanical properties of lightweight aggregate concrete using artificial neural network," Materials, vol. 12, no. 7, 2019, doi: 10.3390/ma12172678.

[6] L. T. Le, H. Nguyen, J. Dou, and J. Zhou, "A comparative study of PSO-ANN, GA-ANN, ICA-ANN, and ABC-ANN in estimating the heating load of buildings' energy efficiency for smart city planning," Applied Sciences (Switzerland), vol. 9, no. 13, Jul. 2019, doi: 10.3390/app9132630.

[7] S. Sena, "Pengenalan Deep Learning Part 1 : Neural network," *Medium*, 19-Mar-2018. [Online]. Available: https://medium.com/@samuelsena/pengenalan-deep-learning-8fbb7d8028ac . [Accessed: 19-Dec-2021].

[8] I. Buatan -If3111, "Artificial Neural Network (Part 2)." [Online]. Available: http://hagan.okstate.edu/NNDesign.pdf

[9] K. Swingler, "Lecture 4: Multi-Layer Perceptrons."

[10] "BAB 2 LANDASAN TEORI 2.1 Teori-teori Dasar/Umum 2.1.1 Neural Network."

[11] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," Machine Learning, vol. 109, no. 2, pp. 373–440, Feb. 2020, doi: 10.1007/s10994-019-05855-6.

[12] D. W. Apley and J. Zhu, "Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models," Dec. 2016, [Online]. Available: http://arxiv.org/abs/1612.08468

[13] Early Stage Diabetes Risk Prediction Dataset, Tersedia di:https://www.kaggle.com/ishandutta/early-stage-diabetes-risk-prediction-dataset

[14] Mengubah Skor ke Bentuk Skor Standar (*Z-score*) dan Skor Terstandar T (*T-score*) di SPSS, Tersedia di : https://www.semestapsikometrika.com/2017/09/mengubah-skor-ke-bentuk-skor-standar-di.html

[15] The Keys of Deep Learning in 100 Lines of Code, Tersedia di : https://towardsdatascience.com/the-keys-of-deep-learning-in-100-lines-of-code-907398c76504