Klasifikasi Daerah Tertinggal di Indonesia dengan Metode Multilayer Perceptron Backpropagation

Untuk Memenuhi Tugas Mata Kuliah Jaringan Syaraf Tiruan

Dosen Pengampu : Dr. Winita Sulandari, S.Si., M.Si.

Disusun oleh :

Fadia Mulyarti	(M0718018)
Fida Mardliyah `	(M0718023)
Fitri Azizah	(M0718024)
Rida Afifatama Hidayat	(M0718046)

PROGRAM STUDI STATISTIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS SEBELAS MARET

SURAKARTA

2021

DAFTAR ISI

BAB I
PENDAHULUAN

1.1. Latar Belakang

Kesenjangan pembangunan di Indonesia masih cukup tinggi. Perbedaan wilayah ini terkonsentrasi di daerah pedaerahan dan perkotaan, antara Indonesia bagian Timur dan Indonesia bagian Barat, serta pulau Jawa dan pulau luar Jawa. Dengan adanya kesenjangan pembangunan daerah di negara Indonesia ini, ada beberapa wilayah yang termasuk daerah tertinggal. Berdasarkan Peraturan Presiden No. 63 Tahun 2020, daerah tertinggal adalah daerah kabupaten/kota yang wilayah serta masyarakatnya kurang berkembang dibandingkan dengan daerah lain dalam skala nasional. Terdapat enam kriteria atau indikator yang menentukan penetapan daerah tertinggal, yaitu perekonomi masyarakat, sumber daya manusia, sarana dan prasarana, kemampuan keuangan daerah, aksesibilitas, dan karakteristik daerah. Oleh karena itu, diperlukan klasifikasi dan pemodelan untuk menentukan sebaran daerah tertinggal sehingga pencegahan dan pengembangan dapat dilakukan secara efisien. Metode yang dapat digunakan untuk mengklasifikasi dan memodelkan daerah tertinggal adalah dengan menggunakan metode jaringan syaraf tiruan dengan algoritma multilayer perceptron backpropagation.

1.2. Rumusan Masalah

Berdasarkan uraian pada latar belakang diatas, diperoleh rumusan masalah sebagai berikut:

a. Bagaimana arsitektur jaringan untuk klasifikasi daerah tertinggal di Indonesia dengan metode MLP Backpropagation?
b. Bagaimana hasil klasifikasi daerah tertinggal di Indonesia dengan metode MLP Algoritma Backpropagation?

1.3. Tujuan Penelitian

Berdasarkan perumusan masalah diatas, diperoleh tujuan penelitian sebagai berikut:

a. Mengetahui arsitektur jaringan untuk klasifikasi daerah tertinggal di Indonesia dengan metode MLP Backpropagation.
b. Mengetahui hasil klasifikasi daerah tertinggal di Indonesia dengan metode MLP Backpropagation.

BAB II
LANDASAN TEORI

2.1. Daerah Tertinggal

Daerah tertinggal adalah suatu daerah kabupaten/kota yang masyarakat dan wilayahnya relatif kurang berkembang dibandingkan daerah lain. Daerah tertinggal berarti daerah yang tidak memiliki industri modern, terdapat hanya sedikit layanan modern, ekonomi lemah dan umumnya memiliki standari hidupi yang rendah. Indonesia terdiri dari 34 provinsi yang terbagi menjadi 514 kabupaten dan kota.

2.2. Jaringan Syaraf Tiruan

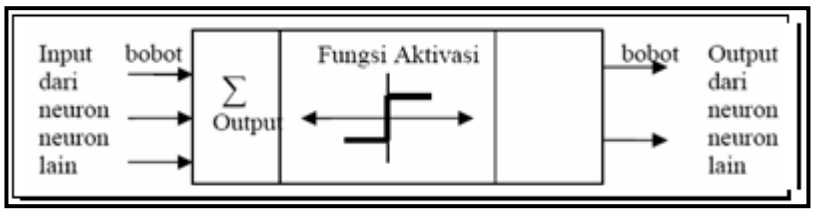
Jaringan Syaraf Tiruan (JST) merupakan sebuah model pada bidang machine learning yang merupakan jaringan sekelompok unit pemroses kecil yang dimodelkan berdasarkan sistem syaraf pada manusia (Fausett, 1994). Model matematis digunakan sebagai fungsi linear atau nonlinear dari jumlahan antara input dan bobot. Perbedaan pembelajaran mesin dengan jaringan syaraf adalah kalau pembelajaran mesin mengacu pada pengembangan algoritma yang dapat menganalisis dan belajar dari data untuk kemudian membuat keputusan sementara kalau jaringan saraf berarti sekelompok algoritma dalam pembelajaran mesin yang melakukan perhitungan yang mirip dengan neutroni di otak manusia.

JST ditentukan oleh tiga hal sebagai berikut, yaitu :

a. Pola hubungan antar neuron disebut arsitektur jaringan
b. Metode untuk menentukan bobot penghubung disebut metode pembelajaran
c. Fungsi aktivasi.

Ada beberapa tipe jaringan syaraf, namun demikian hampir semuanya memiliki komponen-komponen yang sama. Seperti halnya otak manusia, JST juga terdiri dari beberapa neuron dan terdapat penghubung antara neuron-neuron

tersebut. Gambar 2.1 di bawah ini menunjukkan struktur neuron jaringan syaraf tiruan.



Gambar 2.1 Struktur Neuron Jaringan Syaraf Tiruan

Gambar 2.1 menunjukkan bahwa neuron buatan sebenarnya mirip dengan sel neuron biologis. Neuron-neuron bekerja dengan cara yang sama pula dengan sel neuron biologis. Jaringan Syaraf Tiruan yang telah dan sedang dikembangkan merupakan pemodelan matematika dari jaringan syaraf biologis, berdasarkan asumsi :

1. Pemrosesan informasi terjadi pada banyak elemen pemroses sederhana yang disebut neuron.
2. Sinyal dilewatkan antar neuron yang membentuk jaringan neuron.
3. Setiap elemen pada jaringan neuron memiliki 1 (satu) pembobot. Sinyal yang dikirimkan ke lapisan neuron berikutnya adalah informasi dikalikan dengan pembobot yang bersesuaian.
4. Tiap-tiap neuron mengerjakan fungsi aktivasi untuk mendapatkan hasil output masing-masing.

2.3. Multilayer Perceptron

Multilayer perceptron (MLP) merupakan salah satu pemodelan dalam teknologi JST yang mempunyai ciri khusus atau karakteristik memiliki nilai bobot yang lebih baik dari pada pemodelan yang lain, sehingga menghasilkan klasifikasi yang lebih akurat pula. Seperti namanya, jenis jaringan multilayer perceptront ini adalah hasil generalisasi dari arsitektur perceptron dengan satu layer, sehingga memiliki beberapa lapisan ataupun hidden layer, yang letaknya diantara ruang input dan output layer. Secara umum, jaringan seperti ini terdiri dari sejumlah unit neuron sebagai lapisan masukan, satu atau lebih lapisan simpul-simpul neuron

komputasi lapisan tersembunyi, dan sebuah lapisan simpul-simpul neuron komputasi keluaran. Pada MLP digunakan fungsi standar Sigmoid dimana jumlah pembobotan dari sejumlah input dan bias dimasukkan ke activation level melalui fungsi transfer untuk menghasilkan output, dan unit-unit diatur dalam lapisan topologi *feed-forward* yang disebut *Feed Forward Neural Network* (Venkatesan & Anitha, 2006).

2.4. Metode Backpropagation

Backpropagation merupakan salah satu dari JST yakni metode pelatihan yang terawasi (*Supervised Learning*) dengan jaringan multilayer dan memiliki ciri khusus atau karakteristik meminimalkan *error* pada output yang dihasilkan oleh jaringan (Fausett, 1994).

Biasanya pada proses klasifikasi Backpropagation Neural Network. Classifier ini berkerja dengan cara melakukan dua tahap perhitungan yaitu perhitungan maju yang akan menghitung nilai kesalahan (error) antara nilai output sistem dengan nilai yang seharusnya dan perhitungan mundur untuk memperbaiki bobot berdasarkan nilai error tersebut.

2.5. Keras dan Tensorflow

Keras adalah API dengan jaringan saraf tingkat tinggi, membantu jalannya deep learning dan kecerdasan buatan. Keras ditulis dengan bahasa pemrograman Python dan mampu dijalankan pada TensorFlow, CNTK, atau Theano. Keras merupakan *neural network library* yang mudah digunakan. Fitur yang menonjol dari Keras yaitu:

1. Keras merupakan antarmuka tingkat tinggi yang menggunakan TensorFlow dan Theano sebagai backendnya.
2. Keras dapat berjalan lancar di kedua CPU dan GPU.
3. Keras mendukung hampir semua model jaringan saraf - sepenuhnya terhubung, konvolusional, pooling, recurrent, embedding, dan lain lain.

Selanjutnya, model ini dapat dikombinasikan untuk membangun model yang lebih kompleks.

4. Keras adalah kerangka kerja berbasis Python, yang membuatnya mudah untuk dideteksi dan dijelajahi atau dipelajari.

TensorFlow adalah salah satu koleksi software open source untuk komputasi numerik yang menggunakan grafik aliran data. Arsitekturnya yang fleksibel dapat dimanfaatkan untuk menerapkan komputasi ke satu atau beberapa CPU atau GPU pada desktop, server, atau perangkat seluler dengan cukup menggunakan satu API.

2.6. SMOTE (*Synthetic Minority Oversampling Technique*)

Dalam tahap pre-processing data terdapat pengecekan keseimbangan kelas data pada variabel target untuk kasus supervised learning. Jumlah kelas yang tidak seimbang akan membuat model menjadi kesulitan saat melakukan prediksi pada kelas yang jumlahnya sedikit, sehingga membuat hasil prediksi dari model akan kurang optimal. Metode SMOTE Metode SMOTE diperkenalkan oleh Chawla, et al., (2002) digunakan untuk menangani data tidak seimbang dengan memperbanyak pengamatan melalui pembangkitan data buatan. Data buatan tersebut dibuat berdasarkan *k-neighbour* terdekat. Jumlah *k-neighbour* terdekat ditentukan dengan mempertimbangkan kemudahan dalam melaksanakannya. Pembangkitan data buatan berskala numerik berbeda dengan kategorik. Data numerik diukur jarak kedekatannya dengan jarak Euclidean sedangkan data kategorik lebih sederhana yaitu dengan nilai modus. Berikut ini prosedur pembangkitan data buatan.

a. Data Numerik
    1. Menghitung selisih antar vektor utama dengan k-tetangga terdekatnya.
    2. Mengalikan hasil selisih dengan angka yang diacak diantara 0 dan 1.
    3. Menambahkan perbedaan tersebut ke dalam nilai utama pada vektor utama asal sehingga diperoleh vektor utama baru.

b. Data Kategorik

1. Memilih mayoritas antara vektor utama yang dipertimbangkan dengan ktetangga terdekatnya untuk nilai nominal. Jika terjadi nilai sama maka pilih secara acak.
2. Menjadikan nilai tersebut data contoh kelas buatan baru.

2.7. Fungsi Aktivasi

Fungsi aktivasi terdapat pada setiap layer jaringan syaraf tiruan. Fungsi ini adalah fungsi umum yang akan digunakan untuk membawa input menuju output yang diinginkan. Fungsi aktivasi inilah yang akan menentukan besarnya bobot. Penggunaan fungsi aktivasi tergantung pada kebutuhan dan desired output (Fausett, 1994).

Ada beberapa fungsi aktivasi yang sering digunakan dalam jaringan syaraf tiruan, antara lain yang digunakan dalam penelitian ini :

a. Rectified Linear Unit (ReLU)

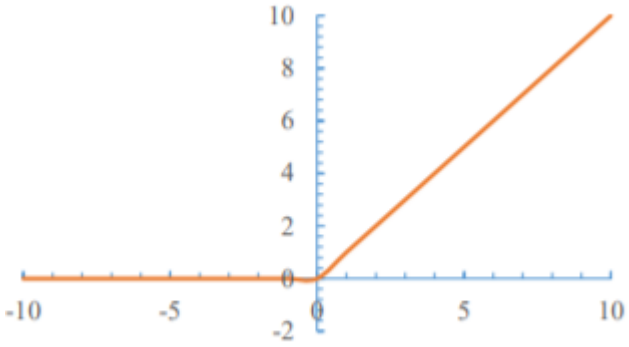
Fungsi ReLU diperkenalkan oleh G.Hinton dan V.Nair pada tahun 2010. Fungsi ini digunakan pada kontex convolutional neural networks. Fungsi ReLU dinyatakan dengan persamaan

$$y(u) = \max(0, u)$$

atau

$$y(u) = \begin{cases} u, & \text{if } u \geq 0 \\ 0, & \text{if } u < 0 \end{cases}$$

Output yang dihasilkan oleh fungsi ReLU akan membentuk grafik yang ditunjukan oleh Gambar 2.2.



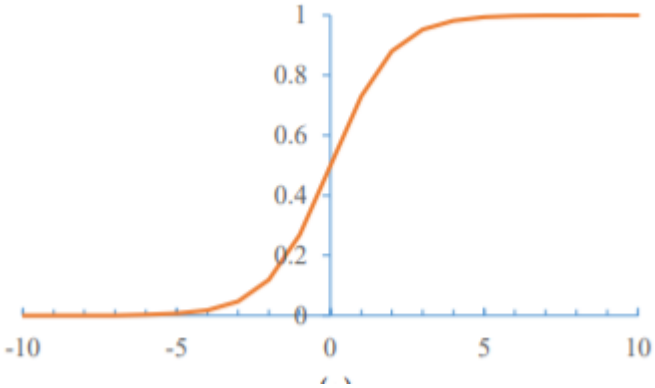
Gambar 2.2 Grafik Fungsi Aktivasi ReLU

b. Fungsi Sigmoid

Fungsi sigmoid digunakan untuk memperoleh output yang bersifat nonlinear, dirumuskan sebagai,

$$y(u) = \frac{1}{1 + e^{(-u)}}$$

Output yang dihasilkan oleh fungsi sigmoid akan membentuk grafik yang ditunjukan oleh Gambar 2.3.



Gambar 2.3 Grafik Fungsi Aktivasi Sigmoid

2.8. Epoch

Epoch merupakan jumlah iterasi atau yang menandakan satu siklus algoritma *machine learning* ‘belajar’ dari seluruh set data training. Satu epoch

berarti sebuah algoritma *machine learning* telah ‘belajar’ dari data training secara keseluruhan. Dalan jaringan saraf tiruan, proses pembelajaran yang berulang-ulang bertujuan untuk mencapai konvergensi nilai bobot.

2.9. *Loss Function*

Loss function yang harus ditentukan yaitu perhitungan nilai error atau seberapa jauh y model dari y real. Contoh loss function adalah SSE, MSE, dan lain-lain. Agar mudah, di sini kita gunakan logaritmic loss. Perintah logaritmic loss yang digunakan namanya binary_crossentropy yang berhubungan erat dengan metode Adam.

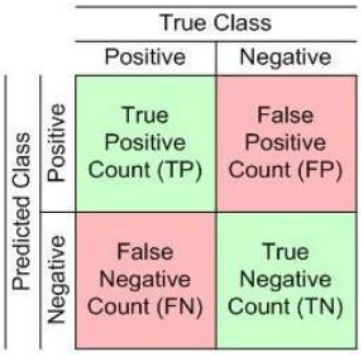
2.10. Adaptive Moment Estimation optimization (Adam)

Algoritma optimasi Adaptive Moment Estimation (Adam) merupakan perluasan dari Stochastic Gradient Descent (SGD) yang baru-baru ini telah digunakan sebagai pembelajaran yang mendalam dalam computer vision dan natural language processing. Algoritma Adam pertama kali diperkenalkan oleh Kingma & Ba (2014). Adam merupakan algoritma optimasi yang mengembangkan dengan memanfaatkan kelebihan dari algoritma *Adaptive Gradient* (AdaGrad) dan *Root Mean Square Propagation* (RMSProp). Selain mengadaptasi tingkat pembelajaran parameter berdasarkan rata-rata pertama (mean) seperti dalam RMSProp, Adam juga menggunakan rata-rata kedua dari gradien (varians *uncentered*). Algoritma menghitung rata-rata pergerakan eksponensial dari gradien dan gradien kuadratnya, dan parameter beta 1 dan beta 2 mengontrol tingkat peluruhan rata-rata pergerakan.

2.11. Evaluasi Model

Menurut Rosandy (2016) pada analisis klasifikasi terdapat berbagai cara untuk mengukur hasil kinerja dari model yang diperoleh, salah satunya adalah dengan *confusion matrix. Confusion matrix* merupakan metode untuk menghitung akurasi pada hasil klasifikasi. Matriks ini memberikan rincian klasifikasi, kelas

yang diprediksi terletak pada bagian atas matriks dan kelas yang diobservasi terletak pada bagian kiri (Sulaehani, 2016).



Gambar 2.3 *Confusion Matrix* Klasifikasi Biner

True Positive (TP) menunjukkan jumlah yang prediksi positif dan keadaan sebenarnya positif, *True Negative* menunjukkan jumlah yang prediksi salah dan keadaan sebenarnya adalah salah, *False Positive* (FP) menunjukkan jumlah prediksi salah yang sebenarnya benar, dan *False Negative* (FN) menunjukkan jumlah prediksi benar yang sebenarnya salah. Perhitung nilai akurasi, presisi, dan recall dapat menggunakan rumus berikut.

$$Akurasi = \frac{TP + TN}{TP + FP + TN + FN}$$

$$Presisi = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

BAB III
METODOLOGI PENELITIAN

3.1. Sumber Data

Penelitian ini menggunakan tujuh variabel yang digunakan untuk mengklasifikasikan Kota/Kabupaten di Indonesia menjadi daerah tertinggal atau tidak tertinggal pada tahun 2020. Variabel yang digunakan adalah Indeks Kedalaman Kemiskinan (X1), Indeks Keparahan Kemiskinan (X2), Indeks Pembangunan Manusia (X3), Pengeluaran Per Kapita Disesuaikan (X4), Umur Harapan Hidup Saat Lahir (X5), Angka Harapan Hidup (X6), dan Harapan Lama Sekolah (X7). Variabel targetnya klasifikasi Kabupaten dan Kota Indonesia yakni tertinggal atau tidak tertinggal. Data diperoleh dari laman resmi Badan Pusat Statistik Indonesia (bps.go.id).

3.2. Tahapan Analisis

Dalam menjalankan algoritma analisis, digunakan software pendukung yaitu Google Colaboratory dan Jupyter Notebook dengan bahasa pemrograman Python. Langkah-langkah yang dilakukan dalam penelitian ini adalah sebagai berikut :

1. Mengentukan variabel dan mengumpulkan data penelitian yang digunakan.
2. Melakukan analisis eksplorasi data.
3. Melakukan *preprocessing* data.
    a. Melakukan pengecekan pada data yaitu cek *missing value,* duplikasi data, dan *imbalance* data.
    b. Melakukan *scaling* data pada ke tujuh variabel independen.
    c. Melakukan pembagian data menjadi 80% data latih dan 20% data uji.
    d. Melakukan *Synthetic Minority Over-Sampling Technique* (SMOTE) pada data latih untuk menangani kelas tidak seimbang.
4. Membangun dan melatih model dengan metode MLP Backpropagation.
5. Menguji model yang dibangun pada data uji.

BAB IV
HASIL DAN PEMBAHASAN

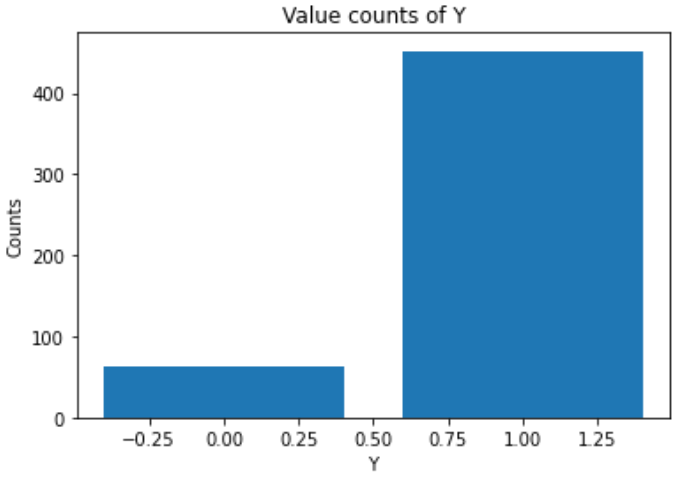
4.1. Statistika Deskriptif

Kabupaten /kota di Indonesia pada tahun 2020 yang memiliki nilai Indeks Kedalaman Kemiskinan (X1) tertinggi sebesar 13,87 adalah kabupaten Lanny Jaya sedangkan kota Sawah Lunto memiliki Indeks Kedalaman Kemiskinan terendah dengan nilai 0,07. Indeks Keparahan Kemiskinan (X2) tertinggi dipegang oleh kabupaten Puncak Jaya dengan nilai 6,99 dan terendah sebesar 0,04 yaitu pada kabupaten Mamuju Utara. Kota Yogyakarta memiliki nilai Indeks Pembangunan Manusia (X3) tertinggi sebesar 86,61 dan terendah pada kabupaten Nduga dengan nilai 31,55. Daerah dengan Pengeluaran Per Kapita Disesuaikan (X4) tertinggi adalah kota Jakarta Selatan sebesar 23575 dan yang terendah sebesar 3975 yaitu kabupaten Nduga. Nilai tertinggi untuk Umur Harapan Hidup Saat Lahir (X5) sebesar 77,65 yaitu di kabupaten Sukoharjo dan nilai terendah sebesar 55,27 ada pada kabupaten Nduga. Kabupaten Sukoharjo memegang nilai tertinggi Angka Harapan Hidup (X6) yaitu sebesar 155,51 dan nilai terendah 110,45 pada kabupaten Nduga. Kota Banda Aceh memiliki nilai Harapan Lama Sekolah (X7) terbesar yaitu 17,79 dan nilai terendah sebesar 3,61 di kabupaten Nduga. Tabel 4.1 berikut memuat statistika deksriptif dari variabel independen pada data.

Tabel 4.1. Statistika Deskriptif Variabel Independen

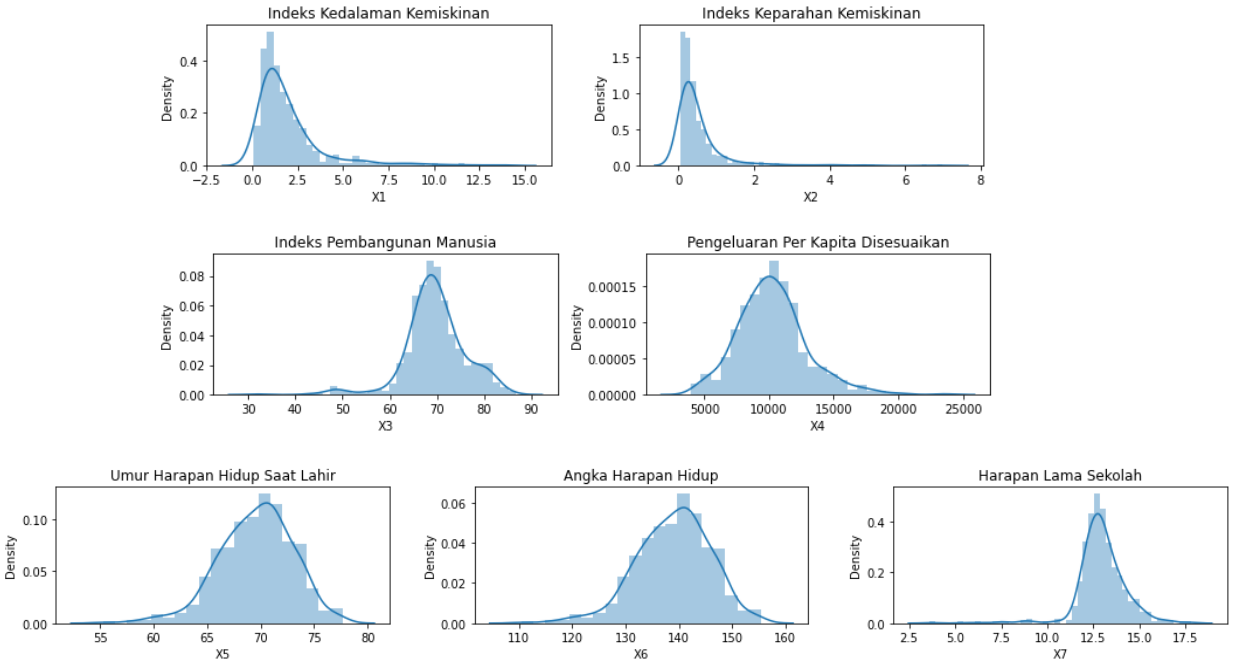
Variabel	Min	Maks	Std	Mean
Indeks Kedalaman Kemiskinan (X1)	0,07	13,87	1,988	2,013
Indeks Keparahan Kemiskinan (X2)	0,04	6,99	0,779	0,543
Indeks Pembangunan Manusia (X3)	31,55	86,61	6,516	69,631
Pengeluaran Per Kapita Disesuaikan (X4)	3975,00	23575	2694,126	10259,716
Umur Harapan Hidup Saat Lahir (X5)	55,27	77,65	3,461	69,548
Angka Harapan Hidup (X6)	110,45	155,51	6,938	139,009
Harapan Lama Sekolah (X7)	3,61	17,79	1,331	12,956

Proporsi kelas untuk daerah tertinggal yang ada di Indonesia dapat dilihat pada Gambar 4.1. Nilai 0 berarti “Ya” dan 1 berarti “Tidak” untuk pengkategorian kelas daerah tertinggal. Dapat dilihat bahwa jumlah daerah tertinggal sebanyak 62 dan yang bukan merupakan daerah tertinggal ada sebanyak 452.



Gambar 4.1. Proporsi Kelas Daerah Tertinggal

Gambar 4.2 berikut menunjukkan plot distribusi data untuk seluruh variabel independen.



Gambar 4.2. Plot Distribusi Data

4.2. Preprocessing Data

Tahap awal *preprocessing* adalah pengecekan *missing value* pada dataset awal. Setelah dicek dapat diketahui bahwa terdapat satu *missing value* pada variabel Indeks Keparahan Kemiskinan (X2). Untuk mengatasi missing value caranya dengan imputasi sesuai dengan bentuk distribusi data, jika data berdistribusi normal menggunakan mean dan jika tidak berdistribusi normal akan digunakan median. Sesuai dengan Gambar 4.2, variabel X2 tidak berdistribusi normal sehingga akan diimputasi menggunakan median. Setelah itu dilakukan pengecekan duplikasi data dan didapatkan tidak terdapat duplikasi pada data.

Langkah selanjutnya dalam preprocessing data yaitu melakukan *scaling* atau transfromasi data menggunakan MinMax *scaler* dengan rentang 0-1 untuk seluruh variabel independen. Statistika deskriptif untuk data yang sudah diimputasi dan di *scaling* disajikan pada Tabel 4.2.

Tabel 4.2. Statistika Deskriptif Setelah Preprocessing

Variabel	Min	Maks	Std	Mean
Indeks Kedalaman Kemiskinan (X1)	0	1	0,144	0,141
Indeks Keparahan Kemiskinan (X2)	0	1	0,112	0,072
Indeks Pembangunan Manusia (X3)	0	1	0,118	0,692
Pengeluaran Per Kapita Disesuaikan (X4)	0	1	0,138	0,321
Umur Harapan Hidup Saat Lahir (X5)	0	1	0,155	0,638
Angka Harapan Hidup (X6)	0	1	0,154	0,634
Harapan Lama Sekolah (X7)	0	1	0,094	0,659

4.3. Pembagian data

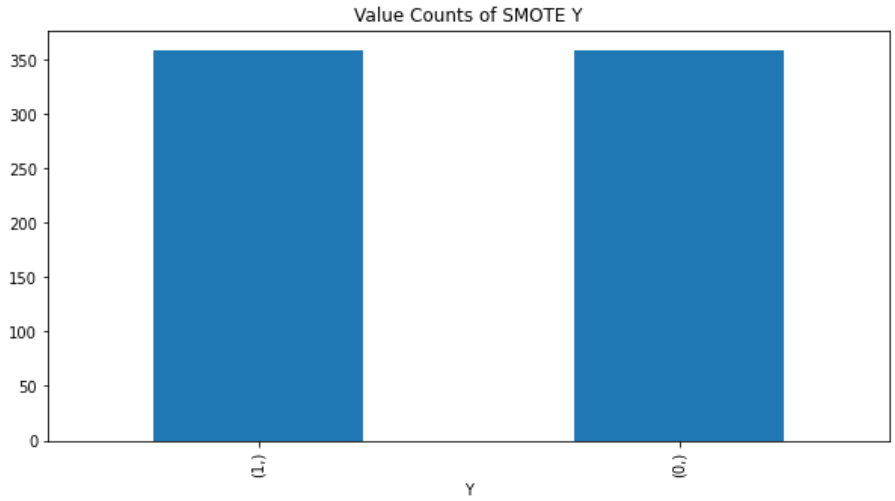
Setelah dipreprocessing, data dibagi menjadi data latih dan data uji.

Tabel 4.3. Pembagian Data

Data Latih (80%)	Data Uji (20%)
Jumlah : 411	Jumlah : 103
0 : 52	0 : 10
1 : 359	1 : 93

4.4. SMOTE

Gambar 4.1 menunjukkan bahwa data tidak seimbang (*imbalance data*) sehingga untuk menanganinya akan digunakan teknik SMOTE. Proporsi kelas untuk daerah tertinggal setelah dikenai SMOTE dapat dilihat pada Gambar 4.3. Dapat dilihat bahwa jumlah daerah tertinggal dan bukan tertinggal adalah sebanyak 359 sehingga total data latih menjadi 718.



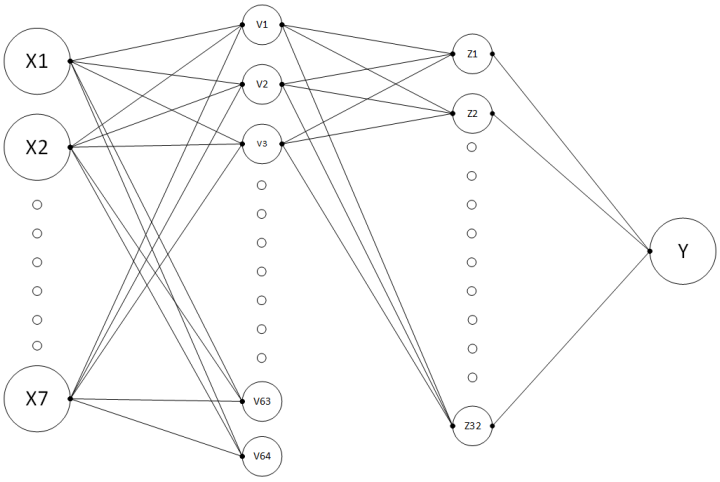
Gambar 4.3. Proporsi Kelas Daerah Tertinggal Setelah SMOTE

4.5. MLP Backpropagation

Pemodelan menggunakan metode MLP Backpropagation dapat dilakukan dengan Tensorflow. Pemodelan menggunakan Sequential method dengan 4 buah layer yang terdiri dari 1 *input layer*, 2 *hidden layer*, dan 1 *output layer*. *Input layer* memiliki jumlah unit sebanyak 7, *hidden layer* pertama sebanyak 64 unit, dan hidden *layer* kedua memiliki 32 unit dengan fungsi aktivasi menggunakan *rectified linear unit* (ReLU). Layer terakhir yaitu *output layer* memiliki jumlah unit sesuai dengan jumlah output dan menggunakan fungsi aktivasi Sigmoid. Pemodelan data latih menggunakan *validation split* sebesar 0,2 sehingga data latih terbagi lagi menjadi data *train* dan *test* dengan rasio 8:2.

Tahap selanjutnya dalam membangun model yaitu tahapan kompilasi. Pada tahap ini ditentukan model loss, metrics dan optimizer. Model ini menggunakan

loss function berupa binary_crossentopy yang umumnya digunakan untu klasifikasi biner, jenis metrics untuk mengukur performa model berupa akurasi dan optimizer menggunakan Adam. Untuk lebih jelasnya, struktur model yang digunakan dapat dilihat pada Gambar 4.4 dan keterangan pada Tabel 4.4.



Gambar 4.4. Arsitektur Model

Tabel 4.4. Struktur Model MLP

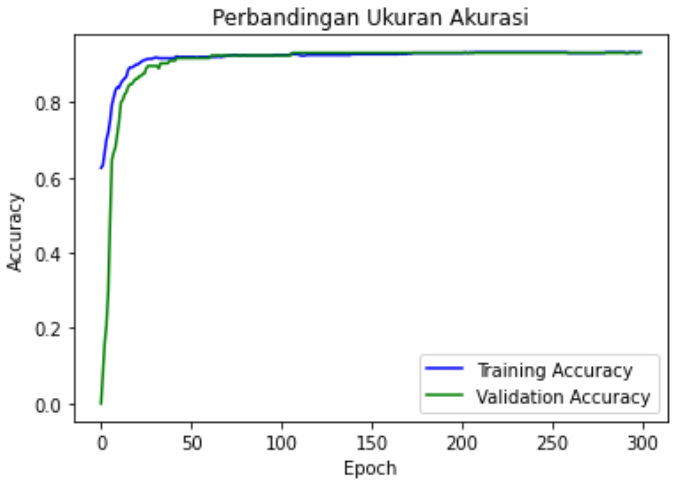
Model MLP	Keterangan
Jumlah layer	4 (1 input, 2 hidden, dan 1 output)
Jumlah unit pada input layer	7 (sesuai dengan jumlah variabel independent)
Jumlah unit pada hidden layer	64 dan 32
Jumlah unit pada output layer	1 (sesuai dengan variabel target)
Fungsi aktivasi	reLU dan Sigmoid
Learning rate	0,001
Optimizer	Adam
Metrics	Akurasi
Loss	Binary crossentropy
Epoch	25

Proses pelatihan model dengan backpropagation memiliki dua tahapan yaitu *forward* (perhitungan maju) dan *backward* (perhitungan mundur). Proses perhitungan maju akan menghitung nilai kesalahan (*error*) menggunakan *loss function*. Pelatihan model pada data latih yang telah dibuat menghasilkan nilai

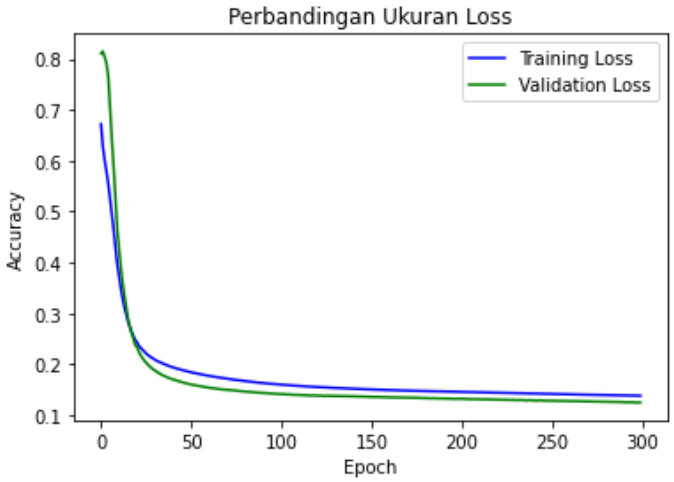
akurasi dan loss antara data *train* dan *test* untuk epoch dapat dilihat pada Tabel 4.5. Plot perbandingan akurasi dan loss antara data *train* dan *test* ditunjukkan pada Gambar 4.5 dan 4.6.

Tabel 4.5. Akurasi dan Loss Data Latih dan Uji

Epoch ke-	Akurasi		Loss	
	Train	Test	Train	Test
10	0,8397	0,7153	0,3960	0,4640
20	0,8990	0,8611	0,2476	0,2369
30	0,9164	0,8958	0,2107	0,1898
40	0,9164	0,9097	0,1951	0,1714
50	0,9199	0,9167	0,1851	0,1608
100	0,9251	0,9236	0,1604	0,1418
150	0,9268	0,9306	0,1508	0,1361
200	0,9303	0,9306	0,1455	0,1320
250	0,9321	0,9306	0,1417	0,1282
300	0,9321	0,9306	0,1348	0,1250



Gambar 4.5. Perbandingan Ukuran Akurasi



Gambar 4.6. Perbandingan Ukuran Loss

Model yang optimal didapatkan dengan melihat nilai *error* terkecil dan akurasi terbesar dari proses pelatihan, yaitu didapatkan pada epoch ke-300. Selanjutnya proses perhitungan mundur akan memperbaharui bobot dan bias berdasarkan nilai error tersebut. Bias dan bobot tiap layer dengan model yang optimal dapat dilihat pada Lampiran 1-6.

4.6. Evaluasi Model

Hasil confusion matrix untuk data latih dapat dilihat pada Tabel 4.6. Evaluasi performa model pada data latih menghasilkan akurasi sebesar 94,71% dan nilai presisi serta recall dapat dilihat di tabel.

Tabel 4.6. *Confusion Matrix* Data Latih

Pred	0	1	Presisi	Recall
0	334	25	0,96	0,93
1	13	346	0,93	0,96

Setelah didapatkan model terbaik dan telah diukur performanya pada data latih, langkah selanjutnya yaitu mengevaluasi model tersebut untuk memprediksi data uji yang berjumlah 103 data dan dikemudian diukur performanya melalui confusion matrix.. Hasil confusion matrix data uji disajikan dalam Tabel 4.7

Evaluasi model pada data uji menghasilkan akurasi sebesar 95,15% dan nilai presisi serta recall dapat dilihat di tabel.

Tabel 4.7. Confusion Matrix Data Uji

Pred	0	1	Presisi	Recall
0	10	0	0,67	1,00
1	5	88	1,00	0,95

BAB V
KESIMPULAN

5.1. Kesimpulan

Berdasarkan hasil analisis yang telah dilakukan, maka dapat disimpulkan bahwa:

1. Arsitektur jaringan untuk klasifikasi daerah tertinggal di Indonesia dengan metode MLP Backpropagation menggunakan 4 buah layer yang terdiri dari 1 input layer dengan 7 unit, 2 hidden layer dengan 64 dan 32 unit, dan 1 output layer dengan 1 unit. Fungsi aktivasi yang digunakan *rectified linear unit* (ReLU) dan Sigmoid. Pemodelan data latih menggunakan validation split sebesar 0,2. Model ini menggunakan *loss function* berupa *binary_crossentopy*, jenis metrics berupa akurasi dan optimizer menggunakan Adam.
2. Hasil akurasi klasifikasi daerah tertinggal di Indonesia dengan metode MLP Backpropagation pada data latih menghasilkan akurasi sebesar 94,71%. Nilai presisi dan recall untuk kelas 0 sebesar 96% dan 93%, untuk kelas 1 sebesar 93% dan 96%. Hasil akurasi pada data uji sebanyak 103 data menghasilkan akurasi sebesar 95,15%. Nilai presisi dan recall untuk kelas 0 sebesar 67% dan 100%, untuk kelas 1 sebesar 100% dan 67%.

5.2. Saran

Penelitian selanjutnya mengenai klasifikasi daerah tertinggal di Indonesia dapat mengambil lebih banyak data untuk diproses. Pengklasifikasian daerah tertinggal dapat menggunakan MLP Backpropagation. Saran untuk pemodelan dapat lebih bervariasi dalam penentuan learning rate dan jumlah unit pada hidden layer, serta maksimal epoch pada saat pelatihan model atau dapat menggunakan *Grid Search* untuk mendapatkan parameter model terbaik.

DAFTAR PUSTAKA

Chawla, N. V., Bowyer , K. W., Hall, L. O. and Kegelmeyer , W. P., 2002. SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, Volume 16, pp. 321-357.

Fausett, L., 1994. Fundamentals of Neural Networks: Architectures, Algorithms, and Application. 1 penyunt. Prentice Hall, New Jersey.

Handayani, A., Jamal, A., dan Septiandri, A., A. 2017. Evaluasi Tiga Jenis Algoritme Berbasis Pembelajaran Mesin untuk Klasifikasi Jenis Tumor Payudara. *Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI)*, 6(4): 394-403.

Jadhav, A., D'Cruz, J., Chavan, V., Dighe, A. & Chaudhari, J. 2016. Detection of Lung Cancer Using Backpropagation Neural Networks and Genetic Algorithm. International Journal of Advanced Research in Computer and Communication Engineering, 5(4): 963-967.

Kingma D, Ba J. Adam: A method for stochastic optimization. ArXiv Prepr ArXiv14126980. 2014

Lukito, Y. 2017. Model Multi Layer Perceptron untuk Indoor Positioning System Berbasis Wi-Fi. Jurnal Tkenologi dan Sistem Komputer, 5(3): 123-128.

Mutmainah, S. 2021. Penanganan Imbalance Data Pada Klasifikasi Kemungkinan Penyakit Stroke. Jurnal SNATi, 1(1): 10-16.

P. Venkatesan, & S. Anitha. (2006, November 10). Application of a Radial Basis Function Neural Network for Diagnosis of Diabetes Mellitus. Current Science, Vol. 91, No. 9. pp. 1195-1199.

Republik Indonesia. 2015. Peraturan Presiden Republik Indonesia No 131 Tahun 2015 Tentang Penetapan Daerah Tertinggal Tahun 2015-2019. Sekretariat Negara. Jakarta.

Rosandy, T., 2016. Perbandingan Metode Naive Bayes Classifier Dengan Metode Decision Tree (C4.5) Untuk Menganalisa Kelancaran Pembiayaan. TIM Darmajaya, Volume 02, pp. 52-62.

Siringoringo, R. 2018. Klasifikasi Data Tidak Seimbang Menggunakan Algoritma SMOTE dan K-Nearest Neighbor. Jurnal ISD, 3(1): 44-49.

Sulaehani, R. 2016. Prediksi Keputusan Klien Telemarketing Untuk Deposito Pada Bank Mnggunakan Algoritma Naïve Bayes Berbasis *Backward Eleimination*. *Jurnal Ilmiah ILKOM*, 182:189.

Tika, G., Adiwijaya. 2019. Klasifikasi Topik Berita Bahasa Indonesia Menggunakan Multilayer Perceptron. E-Proceesing of Engineering, 6(1): 2137-2143.

LAMPIRAN

Lampiran 1. Bobot Input Layer ke Hidden Layer 1

X1	X2	X3	X4	X5	X6	X7
-0,35695	0,221985	0,181962	0,378211	0,052378	0,072089	0,226669
-0,10738	0,112326	0,116173	0,501103	-0,16489	0,306539	0,273006
-0,11668	-0,08043	-0,0062	0,224167	0,010278	-0,21656	-0,03987
-0,12249	0,356884	0,094669	0,310437	0,224204	0,340337	-0,09906
0,30819	0,112362	-0,14569	-0,38241	0,205825	-0,16588	-0,05404
0,319515	-0,09329	-0,1598	-0,43924	-0,11906	0,211143	-0,07111
-0,11181	0,000552	-0,2093	-0,22227	-0,12253	0,016236	0,140381
0,088549	-0,10448	-0,05871	-0,01303	0,335057	-0,17542	0,284631
0,282209	-0,21415	-0,19505	-0,25638	0,048608	-0,07829	0,031343
-0,22036	-0,28877	-0,28285	0,176232	-0,07015	-0,19438	0,066609
-0,0797	-0,19433	-0,23906	-0,08053	0,201815	-0,26951	0,217467
-0,28074	0,159333	-0,01836	0,482029	0,26987	0,270729	0,097771
-0,15875	-0,09759	0,096862	0,24876	-0,22293	-0,23929	-0,186
0,011686	-0,022	0,16704	-0,23132	-0,16943	-0,16146	-0,04966
0,388724	0,108509	-0,15808	-0,32735	0,243998	-0,2534	0,014672
-0,12431	0,139699	-0,00962	-0,27883	-0,15132	-0,18408	0,044567
0,010036	-0,12859	-0,01367	-0,56359	0,158914	0,102852	0,005474
0,234805	0,085309	-0,11031	-0,37706	-0,0443	0,196386	-0,08374
-0,18662	-0,09567	-0,09811	0,489424	-0,13082	0,231381	0,344504
-0,16753	0,136287	0,09906	0,638213	0,282809	0,298709	0,245633
0,36007	0,042564	-0,24195	-0,31125	0,059132	-0,07679	0,127663
0,289708	-0,07472	0,167149	-0,41602	0,054141	0,044446	0,103941
0,27223	0,11896	-0,24771	-0,14811	0,184417	-0,15379	0,027871
0,201647	0,010568	0,209025	-0,55886	-0,08934	-0,11874	-0,03987
-0,17197	0,207478	0,106801	0,608637	0,204433	0,159703	0,324614
0,01737	-0,14861	0,284475	0,656821	0,348956	0,331942	0,335302
0,367893	0,085834	-0,01261	-0,25711	0,123448	0,00426	-0,22154
-0,14887	0,231165	0,196523	0,276461	0,119468	0,138965	0,000866
-0,2079	-0,13324	0,003297	0,687074	0,122167	0,205171	0,076627
0,429709	-0,10763	0,138119	-0,58959	0,153481	-0,03273	-0,24885
-0,04506	0,259961	0,043302	-0,0682	-0,15155	0,178139	-0,1747
0,306145	-0,26481	-0,16649	-0,24295	0,367199	0,039749	0,283966
0,011573	0,447906	0,042908	0,630347	0,034433	0,305973	0,012232
-0,23639	-0,1415	0,106746	0,550009	0,326549	-0,10372	0,345604
0,049666	0,368598	0,191744	0,603558	0,251393	0,048262	-0,21744
0,326337	0,022567	-0,12642	0,123403	0,187248	-0,24115	-0,21149

0,390869	-0,14622	0,017824	-0,13037	0,176492	-0,19852	-0,22558
-0,08193	0,197139	0,051029	-0,01848	0,033439	-0,236	-0,19925
-0,0821	0,098379	0,115623	-0,28056	0,098274	-0,12782	-0,04916
0,15242	0,110348	0,200908	-0,34228	-0,06487	-0,21852	0,032237
-0,24275	0,036315	-0,21571	0,125722	-0,17576	-0,18758	-0,02716
0,167331	0,084861	0,098777	-0,22326	-0,03921	-0,0619	-0,24126
-0,28885	-0,26886	-0,25436	-0,09958	-0,28678	0,13937	0,269086
0,463306	-0,30921	-0,09798	-0,49682	0,217141	-0,13145	0,029164
-0,10553	0,327235	-0,15572	0,368906	0,262077	0,125042	0,285288
-0,37146	0,084883	0,353102	0,590187	0,217593	-0,22832	0,129257
0,050471	-0,26811	-0,28149	-0,02565	-0,19373	0,120966	-0,08312
0,133945	0,162027	0,140026	0,428657	0,056606	0,224104	-0,06032
-0,29026	-0,21142	-0,03609	0,122125	0,004963	-0,15762	-0,17306
0,170781	0,039654	0,190031	-0,39709	-0,06665	-0,22964	0,064207
-0,17331	0,244043	-0,09324	-0,0853	0,056766	-0,03113	-0,19223
-0,03514	-0,1278	-0,03268	-0,49652	-0,18753	-0,29385	0,103237
0,119676	0,175848	-0,15717	0,074186	-0,19136	-0,17089	0,178767
0,047357	-0,13101	0,170543	-0,10178	0,024003	-0,20688	-0,20429
-0,26308	-0,26556	-0,24658	0,192317	-0,18973	0,097616	-0,27761
0,009156	0,20754	-0,22658	0,012993	-0,03381	-0,31672	0,072873
-0,19286	-0,07905	0,236276	0,222748	0,070512	0,309363	-0,09969
0,115079	-0,21182	0,13857	-0,15021	0,056612	-0,29176	0,012183
-0,00464	-0,14729	0,129545	0,009772	-0,04716	-0,23482	-0,23113
-0,09863	0,119648	0,236935	-0,16521	0,073237	-0,14646	-0,26355
0,400499	0,2167	0,172739	-0,13759	-0,26732	-0,12133	0,03328
-0,42694	0,126558	0,090926	0,599422	0,245006	0,230167	0,267552
0,073983	-0,19256	-0,205	-0,0864	0,161323	-0,18606	0,177661
0,138339	-0,24356	0,132388	-0,06844	-0,27789	-0,24935	-0,02699

Lampiran 2. Bias Input Layer ke Hidden Layer 1

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
-0,06273	-0,12139	0	-0,09439	0,225631	0,243437	0	0,164102

9	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0,212866	0	0	-0,08372	0	0	0,188077	0

17	**18**	**19**	**20**	**21**	**22**	**23**	**24**

0,081209	0,165502	-0,11671	-0,07304	0,177361	0,114018	0,165709	0,200296

25	26	27	28	29	30	31	32
-0,12274	-0,07156	0,146759	-0,21569	-0,11017	0,181936	-0,02577	0,193801

33	34	35	36	37	38	39	40
-0,08248	-0,11057	-0,14192	0,087509	0,176498	0,001609	-0,01864	0,132493

41	42	43	44	45	46	47	48
0	0,221884	0	0,153051	-0,07728	-0,13112	0	-0,13002

49	50	51	52	53	54	55	56
0	0,160472	-0,0091	0,243792	-0,01052	0	0	-0,02844

57	58	59	60	61	62	63	64
-0,10577	-0,03154	0	-0,006	0,158578	-0,20076	-0,01693	-0,00516

Lampiran 3. Bobot Hidden Layer 1 ke Hidden Layer 2

1	2	3	4	5	6	7	8
0,385395	0,411514	-0,17635	0,162154	-0,20386	-0,53841	-0,13782	0,269229
-0,00375	0,383158	-0,06164	-0,08096	-0,2067	-0,26824	-0,13979	0,310057
0,077331	-0,30008	0,102474	0,24559	0,233987	0,577514	0,006921	-0,05005
0,278539	0,060458	0,102423	-0,09295	-0,1905	-0,1932	-0,11031	-0,00735
0,255415	0,251616	-0,14799	0,331455	-0,36634	-0,32022	0,212572	0,268573
-0,25723	-0,01037	0,12068	0,057916	0,578023	0,277442	-0,04488	0,199743
0,310368	0,417635	-0,14844	0,273441	-0,19989	-0,40481	0,162699	0,142234
-0,03091	-0,22366	0,161128	0,048118	0,599446	0,664133	0,113793	-0,14889

-0,05516	-0,03361	0,24571	0,072848	-0,2303	-0,48773	0,187266	0,157944
0,228827	0,100969	-0,20994	0,15835	-0,40216	-0,16379	-0,03613	-0,1206
0,349731	0,343127	-0,14187	0,31601	-0,63061	-0,27076	0,000228	-0,10309
0,178557	0,420429	-0,03949	0,344505	-0,54345	-0,57517	-0,20903	-0,09498
0,228305	-0,09134	0,050909	0,121799	-0,11768	0,203174	-0,04605	-0,04069
0,129429	0,166102	0,215505	-0,13654	0,191524	-0,18052	-0,06036	0,006252
0,198217	-0,22444	0,081171	-0,10281	-0,01474	-0,08062	0,2126	-0,22109
-0,10875	-0,29533	-0,1203	-0,25258	0,425416	0,718583	0,050215	0,056301
-0,11056	-0,09601	0,114016	0,064522	0,434214	0,399606	-0,05601	-0,00178
-0,10748	-0,13097	-0,06875	-0,06014	0,421332	0,4871	-0,18899	0,195448
0,15669	-0,27593	-0,13304	0,075313	0,417907	0,482826	0,111452	0,037523
0,138655	-0,21654	0,123739	-0,18021	-0,09772	-0,10178	-0,05764	-0,05702
0,197316	-0,17041	0,006932	-0,16292	0,045787	-0,02761	-0,19149	-0,10319
-0,03237	-0,15079	0,24951	-0,18581	0,249149	0,230127	-0,01045	0,100675
-0,0879	0,125602	-0,23672	0,11236	0,282837	0,367247	0,239866	0,141455
0,046972	0,207822	0,056793	-0,06924	0,01435	-0,15596	0,056309	-0,18556
0,109313	0,230207	0,029703	-0,15277	0,087736	0,210258	0,215037	-0,22519
0,39253	0,197145	0,207813	0,032547	-0,46773	-0,24645	-0,24213	0,243237
0,170035	-0,15981	0,060591	-0,03338	0,178959	0,367521	-0,00988	0,110088
-0,07186	0,216324	-0,09656	0,16743	-0,09324	-0,09736	0,103832	-0,03458
-0,19517	0,216016	0,162237	0,106956	0,071681	0,009232	0,159091	0,02847
0,01917	-0,07151	-0,07825	-0,18555	0,333316	0,500346	0,154001	0,010116

-0,19703	-0,31664	0,179932	-0,02818	0,409693	0,630041	-0,04881	0,181204
-0,01863	0,049085	-0,12712	-0,13315	0,269098	0,359582	-0,12224	0,117365

9	**10**	**11**	**12**	**13**	**14**	**15**	**16**
-0,36515	-0,19889	-0,20702	0,339938	0,134403	-0,19023	-0,38763	-0,14159
-0,60603	0,139327	0,165821	0,091987	-0,03767	0,042518	-0,16054	0,204338
0,347379	-0,15963	-0,24053	-0,2242	-0,17741	0,137397	0,148745	0,116232
-0,23293	0,172113	-0,1687	0,28855	0,21667	0,235596	-0,35465	0,176712
-0,57626	-0,07634	-0,02283	0,338995	0,180068	-0,15175	-0,45432	-0,16836
0,379826	0,087339	-0,17425	0,116249	0,173014	-0,15131	0,46333	0,033914
-0,28253	0,047347	-0,08475	0,307366	-0,14395	-0,16726	-0,4499	-0,21612
0,360533	-0,03874	-0,08798	0,148194	0,016366	-0,17466	0,375994	-0,23253
-0,65805	-0,02071	0,194278	0,35158	0,249291	0,102131	-0,19787	0,031606
-0,3991	-0,21837	0,246503	0,283115	-0,06302	0,180949	-0,32098	0,15238
-0,18645	-0,19097	0,249094	0,364372	-0,04886	-0,04168	-0,28861	-0,104
-0,18958	0,079083	0,11902	0,380674	-0,08948	0,091809	-0,33242	-0,042
0,224218	-0,20763	0,205084	-0,08307	-0,18628	0,069928	-0,18512	0,155348
0,209732	-0,1462	0,203729	-0,213	-0,08129	-0,0198	-0,17827	0,003562
0,070254	-0,06768	0,202675	0,168561	0,005856	-0,18853	-0,03762	-0,24832
0,493495	-0,2275	0,141531	0,034	-0,14749	-0,13643	0,405302	0,175911
0,354526	0,00196	-0,15631	-0,07465	-0,0314	0,159075	0,410104	-0,04485
0,184363	-0,18754	-0,00354	0,197457	0,172942	0,028972	0,077818	-0,19123

0,63663	0,179225	-0,08594	0,184447	-0,22877	0,070996	0,329205	0,006033
0,168869	-0,04258	0,14207	0,063478	-0,06423	0,234014	-0,14433	0,160069
0,010309	0,191454	-0,13735	-0,08646	0,238073	-0,12738	-0,01865	0,230138
0,560336	0,09909 2	0,220656	0,202995	-0,0682	0,156115	0,153373	-0,10909
0,366633	-0,22665	-0,19106	-0,26543	-0,12545	0,056606	0,040975	-0,09657
0,058548	0,170084	0,240508	0,029097	-0,00702	0,056099	0,033616	-0,047
0,22812	0,013892	-0,22204	-0,15952	0,189525	0,0559	-0,17578	0,194314
-0,47105	0,153392	-0,06971	-0,05398	0,156037	0,211963	-0,05986	-0,0454
0,535069	0,000578	-0,22712	-0,27237	0,178775	0,044958	0,282087	0,228166
0,002297	0,06351	-0,13754	-0,01084	-0,18904	0,052413	-0,02921	0,195294
-0,03022	0,069297	0,115456	-0,10082	-0,06172	0,020268	-0,07664	-0,2356
0,181523	0,19232	-0,12549	0,072261	-0,05727	-0,14621	0,372101	0,231012
0,264197	-0,20144	-0,18632	-0,13684	-0,05189	0,218424	0,12716	-0,13955
0,384018	-0,23961	-0,22876	-0,07988	-0,0396	-0,17414	0,456475	0,218112

17	**18**	**19**	**20**	**21**	**22**	**23**	**24**
-0,11449	-0,32004	0,093659	0,353874	-0,35154	0,104757	-0,04844	-0,29203
-0,11346	-0,01439	0,373777	0,337454	-0,45214	0,216973	-0,08264	-0,13358
0,322782	0,444397	0,098764	-0,09169	0,492327	0,057467	0,324565	0,506546
-0,07371	-0,20483	0,18047	0,188431	-0,34552	0,097214	0,02645	-0,12541
0,154312	-0,13684	0,26972	-0,00364	-0,29719	0,039208	-0,15724	-0,51871
-0,08142	0,37295	-0,24179	0,120035	0,188828	-0,1539	0,338852	0,09708

0,113586	-0,00199	0,332405	0,322858	-0,32525	-0,1546	-0,00062	-0,4526
0,246779	0,35034	-0,03119	0,151005	0,077059	0,020688	0,159534	0,487154
0,179534	-0,06031	0,350675	-0,08242	-0,49223	0,248078	-0,05297	-0,52962
-0,04268	-0,10838	0,038506	0,228664	-0,07645	0,049766	-0,36534	-0,07846
-0,22996	-0,02755	0,351303	0,234525	-0,20678	0,02537	-0,38381	-0,14805
-0,10895	-0,17802	0,196562	0,183845	-0,14713	-0,11625	-0,37571	-0,3529
-0,19679	-0,18996	-0,09951	0,087612	-0,09226	-0,21057	-0,12573	0,162227
-0,01504	-0,07548	0,24247	0,001271	-0,16539	-0,02914	-0,00665	-0,07572
-0,0617	0,189864	-0,15177	-0,19331	0,243871	-0,00507	0,091905	-0,14392
-0,00352	0,161762	-0,23244	0,136848	0,470887	0,154314	0,220646	0,475102
0,145043	0,029017	-0,29154	-0,06036	0,194111	0,222932	0,185709	0,068732
0,064202	0,006368	-0,09363	0,061711	0,148669	0,292115	0,294071	0,40277
0,114031	0,37672	-0,20852	-0,00962	0,011059	0,189839	0,249202	0,269387
0,103411	0,233436	-0,19837	0,18378	-0,18465	-0,10209	0,236345	-0,20926
0,102434	-8,1E-05	-0,12772	-0,14428	-0,07135	0,092108	0,055024	0,137371
0,106035	0,34337	-0,27779	0,074007	0,077202	0,25116	0,219588	0,303363
0,284048	-0,0597	-0,2694	-0,13072	0,468858	0,166008	0,313041	0,183136
0,112521	0,112659	0,079528	-0,09009	0,202946	-0,18971	0,04287	-0,18244
0,229237	-0,1569	-0,12573	0,073139	-0,20888	-0,1954	0,207299	0,17614
-0,19906	0,073857	0,480267	0,194611	-0,12807	0,177626	-0,16109	-0,17363
-0,03579	-0,01125	-0,31213	0,050688	0,358582	-0,04644	0,44526	0,301668
0,072485	-0,02733	-0,06347	-0,11644	-0,06919	-0,17336	-0,05067	-0,09823

0,234923	-0,18462	-0,04498	-0,1968	0,065633	-0,10471	0,242874	-0,12774
0,104667	0,055759	0,065329	0,083502	-0,05843	-0,18948	0,31931	0,249188
-0,11068	0,074292	0,063556	-0,1279	0,345268	0,195741	0,434377	0,283983
0,021609	0,296352	-0,09936	0,008548	0,420233	0,097408	0,295776	0,50967

25	**26**	**27**	**28**	**29**	**30**	**31**	**32**
-0,01895	0,212794	-0,33085	0,155371	0,415348	-0,28421	-0,08036	-0,0995
0,254109	0,337252	-0,04394	-0,00989	0,505592	-0,61541	0,247698	-0,17554
-0,19199	0,175923	0,043271	-0,10439	-0,22398	0,41536	0,072444	0,14089
0,053648	0,373304	-0,24365	0,211457	0,285378	-0,53842	0,001943	-0,092
0,384267	0,195248	-0,38559	0,000881	0,264728	-0,23479	-0,07233	0,252853
-0,14732	-0,10941	0,489707	0,165437	-0,20476	0,261485	-0,11036	0,153146
0,312598	0,144686	-0,34399	-0,01502	0,267874	-0,37695	0,193546	0,193863
0,136061	-0,15697	0,028309	-0,26255	-0,06575	0,584774	-0,09246	0,222322
0,416456	0,130033	-0,30301	0,369544	0,552716	-0,32492	-0,14868	0,239018
-0,05484	0,371352	-0,03819	0,271295	0,414978	-0,51338	-0,05664	0,238301
0,245553	0,188408	-0,27032	0,375172	0,146829	-0,13874	-0,18234	0,230206
0,194197	0,20222	-0,19927	0,035008	0,305662	-0,4247	-0,10445	-0,08686
-0,04604	0,087932	-0,20809	-0,12723	0,075866	0,19728	-0,20872	-0,09398
0,013069	-0,20246	0,108011	0,071889	-0,19409	0,201544	-0,23771	-0,07381
0,084381	-0,04597	0,100031	0,113223	-0,23073	-0,16419	0,181751	-0,1133
-0,25118	0,18169	0,512873	-0,04084	-0,21459	0,404232	-0,0904	0,275527

0,192445	-0,16401	0,289339	0,20269	-0,20584	0,369952	0,245314	0,333384
0,075318	-0,22688	0,30957	-0,03054	-0,14493	0,39641	0,029002	0,216619
-0,26606	0,161276	0,194352	0,13241	-0,13786	0,518704	0,25068	0,303395
0,073112	0,129741	-0,13274	-0,24656	-0,18861	0,020452	-0,15848	0,060958
-0,11201	-0,11478	0,198183	0,15885	-0,23422	-0,18604	0,091178	0,098498
-0,10939	0,04267	0,276472	-0,27542	0,055101	0,373298	0,128351	0,213754
0,16596	-0,25353	0,009971	-0,1116	-0,216	0,235566	-0,02653	0,216689
-0,18739	-0,02313	0,183563	-0,03123	-0,1595	-0,04249	0,01615	0,031109
0,137049	-0,21256	-0,02839	-0,06557	-0,19573	-0,2268	0,018204	-0,02581
0,143377	0,36688	-0,1783	0,091882	0,116396	-0,34333	0,219447	0,113277
0,136607	0,094895	0,242298	-0,24286	-0,23032	0,540252	-0,02543	-0,12706
0,018245	-0,15424	-0,08819	-0,03192	0,034664	0,112675	-0,08011	-0,03529
-0,07847	-0,22153	-0,18425	-0,04907	0,071084	0,100379	0,012534	0,229668
0,182943	-0,03433	0,073734	-0,18498	0,100705	0,451635	0,390895	-0,0165
-0,27398	0,217013	0,068253	-0,02191	-0,26716	0,195218	0,105449	0,288776
-0,1721	0,009862	0,480664	-0,08889	-0,07162	0,530489	-0,02754	0,061231

33	**34**	**35**	**36**	**37**	**38**	**39**	**40**
0,21312	0,143379	0,441484	-0,1154	-0,18521	-0,16235	-0,09214	-0,31857
0,159272	0,399343	0,422047	-0,36209	-0,51209	0,147492	0,017034	0,061413
-0,071	0,138298	-0,36896	0,21765	0,247509	0,116924	0,053487	0,270403
0,403331	0,187398	0,331494	-0,05885	-0,45088	0,031345	-0,2292	0,026818

0,099319	0,334325	0,105735	0,15535	-0,36946	0,202758	-0,00906	0,014662
-0,17307	0,17892	-0,15478	0,453774	0,315022	0,246603	0,023676	0,110986
0,337705	0,239137	-0,01263	-0,35066	-0,09251	-0,17184	0,140873	-0,00264
-0,04543	-0,2971	-0,21326	0,078233	0,384493	-0,11454	0,073771	0,279769
0,285019	0,445587	0,321242	-0,27001	-0,34461	0,163731	0,015105	-0,10668
0,059195	0,25264	0,361912	-0,11351	-0,07067	-0,02494	-0,223	0,164144
0,222149	0,22866	0,031568	0,062305	-0,02969	-0,16203	-0,25373	-0,19041
0,438257	0,317858	0,217556	-0,00757	-0,29828	0,203137	-0,10265	0,215077
-0,07731	-0,1144	0,238397	-0,02704	-0,2356	-0,09707	-0,06049	0,092514
0,178627	0,156467	-0,04089	0,159055	-0,02501	0,226408	0,116078	-0,21525
-0,19281	-0,08546	0,176084	-0,02871	-0,13443	0,00353	0,07102	-0,11198
-0,14937	-0,05134	-0,15471	0,834401	0,108442	-0,02819	-0,16307	0,120098
-0,06186	-0,01142	-0,29134	0,40949	0,342916	-0,14928	0,029356	0,275621
0,195741	-0,17044	-0,09516	0,251633	0,31592	-0,19103	0,019661	0,237573
-0,20063	-0,28298	-0,10325	0,294413	0,568706	0,141532	-0,15186	0,122606
-0,16735	-0,05812	-0,14634	0,103409	0,021735	-0,24619	0,239702	0,205623
0,088786	0,042112	-0,04827	-0,109	-0,23158	0,202537	0,159555	-0,01855
0,102742	-0,16095	-0,13847	0,58221	0,187245	0,093922	0,127209	0,198426
-0,12583	0,052323	-0,09338	0,412964	0,401018	-0,14556	0,067966	0,129141
-0,2385	-0,06533	0,043152	-0,20026	-0,14512	-0,1789	-0,19169	0,037905
-0,2259	0,073529	0,194526	-0,08612	0,098012	-0,1284	0,005928	-0,0565
-0,09569	0,095922	0,462167	-0,00202	-0,22695	-0,18485	0,065493	-0,16709

0,193108	-0,17151	0,035115	0,173812	0,306784	0,067707	-0,11711	-0,05295
0,063232	-0,13526	-0,03137	0,213255	-0,21534	0,022861	0,22439	-0,06146
-0,20182	0,069456	0,047638	0,197592	0,00116	-0,02945	-0,05552	0,075431
-0,26924	0,160908	-0,14089	1,01717	0,385529	-0,16637	0,171897	-0,22642
0,130523	-0,04342	-0,21426	0,310087	0,405856	-0,0656	-0,06249	0,015004
0,010305	-0,29377	0,107423	0,470195	0,116133	0,16395	-0,10377	0,010468

41	**42**	**43**	**44**	**45**	**46**	**47**	**48**
-0,04377	-0,15201	-0,15036	-0,18249	0,333382	0,291674	-0,10226	0,018902
0,228152	-0,41674	-0,12842	-0,0073	-0,14448	0,301087	-0,21813	0,337895
0,07843	0,183131	-0,20712	0,28987	-0,21253	-0,30934	0,218512	0,077554
-0,19205	-0,29551	0,093404	-0,34746	0,33304	0,244531	0,162394	0,261297
-0,07298	-0,19283	0,107903	-0,3254	0,329622	0,28657	0,085929	0,062177
-0,02717	0,309864	0,160091	0,341376	0,197877	-0,16542	-0,09077	-0,20586
-0,11397	-0,24024	-0,11378	-0,47698	0,272954	0,484029	-0,0606	-0,05431
-0,06091	0,446655	-0,05948	0,135677	0,129459	-0,10651	-0,22902	-0,14754
-0,10591	-0,34357	0,227691	-0,09719	0,077965	0,172382	-0,09049	0,276243
-0,06928	-0,43639	0,110049	-0,4191	-0,11172	0,388399	0,176071	0,07228
0,225886	-0,45694	-0,23566	-0,19972	0,286429	0,210484	0,202671	0,273224
0,127342	-0,07006	-0,12308	0,002878	0,13917	0,281145	-0,08256	0,307674
0,247088	0,150204	-0,07927	0,214691	-0,05596	0,206596	-0,09493	-0,08341
0,241504	-0,18756	-0,01428	0,190865	-0,14949	-0,1225	0,030484	-0,01341

-0,24539	0,078565	-0,24743	0,075714	-0,04026	-0,19159	-0,05011	0,028867
0,035934	0,288782	0,22525	0,522865	0,216193	-0,36694	-0,2139	-0,21172
0,001234	0,272151	0,143849	0,473566	0,149616	0,01822	0,102052	-0,22366
-0,10888	0,478273	0,16612	0,272505	-0,12247	-0,18548	-0,17578	0,015381
-0,13597	0,375545	0,166496	0,360617	-0,15152	-0,33235	-0,22127	-0,01242
0,16391	0,24979	0,018349	-0,20487	-0,01669	-0,10596	-0,00144	0,211908
0,233604	-0,0422	0,008335	-0,05273	0,008052	-0,17346	-0,01877	-0,17959
-0,04732	0,495109	-0,09252	0,367704	-0,18077	-0,28119	-0,18078	-0,06279
-0,06586	0,521004	0,224935	0,281073	0,193656	-0,15315	0,074325	0,076059
-0,17362	-0,21459	0,039946	-0,14434	-0,1202	-0,13742	-0,022	-0,24481
0,051976	-0,15105	0,021623	0,10997	-0,18758	-0,23447	0,003912	-0,21307
0,11256	-0,4909	-0,16631	-0,29592	0,046763	0,235509	0,134682	0,204074
-0,22104	0,476339	0,070914	0,506994	0,265465	-0,11793	-0,08885	-0,10314
0,119693	0,065439	0,150902	-0,24145	-0,22369	-0,01736	0,185727	-0,06029
0,004431	-0,22269	0,051183	-0,17817	-0,01179	-0,13473	-0,21939	0,01212
-0,00127	0,39389	-0,2155	0,448765	-0,05491	-0,31819	0,054047	0,035981
-0,22178	0,355486	0,115139	0,237255	-0,00559	-0,032	-0,21561	0,089143
-0,04322	0,415896	0,108061	0,18668	0,087601	-0,24304	0,015676	0,105927

49	**50**	**51**	**52**	**53**	**54**	**55**	**56**
-0,0659	-0,10431	-0,06139	-1,32341	-0,01842	0,18724	0,106467	-0,04368
0,013904	-0,32815	-0,25802	-1,41565	-0,11608	-0,0333	0,021682	0,027625

0,134218	0,318114	0,073087	1,04945	-0,07521	-0,00106	-0,1767	0,094168
0,224138	-0,05772	0,228039	-1,46601	-0,10687	-0,17809	-0,14422	-0,24323
-0,00635	0,071603	0,199275	-1,22556	0,060219	0,087964	-0,21187	0,219871
0,076481	0,358665	0,139673	1,516359	-0,08313	0,18539	0,007792	-0,08768
0,079545	-0,15919	0,213929	-1,01351	0,017785	-0,24362	0,236252	-0,19772
0,15616	0,374272	-0,01486	1,347979	0,202982	-0,15166	-0,23521	-0,05262
-0,17008	-0,30548	0,05112	-1,23484	0,101147	0,10626	-0,19105	0,162268
-0,19779	-0,04056	0,15491	-1,12074	-0,21743	0,141462	-0,03919	0,184362
-0,14877	-0,1211	0,113315	-1,47206	-0,03882	0,145248	-0,23377	0,171674
-0,23316	-0,29625	0,241268	-1,37872	0,139194	0,107248	-0,16072	-0,25968
0,034855	-0,03081	-0,08575	0,031874	0,054054	0,198357	-0,01542	-0,24909
0,223981	0,073364	0,150719	-0,0524	0,104432	-0,1643	-0,10298	-0,23155
0,151573	-0,21026	0,215429	-0,19718	-0,09696	0,114777	-0,20193	-0,1458
-0,05717	0,40775	-0,16889	1,314189	0,191475	-0,21295	0,019189	0,108851
0,181582	-0,00579	-0,15174	1,116471	0,287582	-0,14797	0,209735	-0,00858
0,108815	0,040803	0,019016	1,452492	0,055145	0,212364	-0,15988	-0,02324
0,01039	0,129578	0,15423	1,365816	0,062041	0,031726	-0,06027	-0,06957
-0,16129	-0,06444	0,185046	-0,20207	-0,1454	0,182351	0,19213	0,20347
0,056713	0,184012	0,220485	0,087492	-0,1513	-0,15585	0,199603	-0,21921
0,013621	0,288398	-0,12011	1,209143	0,309595	-0,05281	-0,04155	-0,11456
-0,13695	0,279301	0,094865	1,13221	0,028516	-0,07969	0,074638	0,018423
0,110313	0,133239	0,140535	0,084023	-0,11456	-0,12006	0,090468	-0,12937

0,185193	-0,02367	0,173958	0,198612	-0,14085	0,24746	-0,09192	0,130624
0,198995	-0,22805	-0,09296	-1,173	-0,0931	-0,04023	0,033844	0,117201
0,058058	-0,00658	0,088351	1,154425	0,103452	0,062321	0,125709	0,160661
-0,16224	-0,18109	0,190449	-0,17748	0,219962	-0,00831	-0,0017	0,098182
0,221472	-0,1074	0,020222	0,124285	-0,23929	0,187578	0,179644	-0,23488
0,188652	0,172633	0,18916	1,030152	0,540354	-0,19821	-0,08506	0,340794
0,063869	0,203636	-0,24978	1,338336	0,301164	-0,01049	0,099872	0,05999
-0,20357	0,066251	-0,17513	1,477546	-0,0526	-0,17577	0,136193	0,132178

57	**58**	**59**	**60**	**61**	**62**	**63**	**64**
0,016619	-0,08042	-0,13633	0,184549	-0,20504	0,28076	0,114546	0,059023
0,024227	0,013488	0,182732	-0,05372	0,09867	0,423962	-0,03195	0,207417
0,013126	0,103294	0,207441	-0,15116	0,126851	0,023057	0,204906	-0,13685
0,316379	0,008568	0,137031	0,035789	0,107726	0,01724	-0,17352	-0,06616
0,147285	0,191701	-0,03576	0,237782	-0,12014	0,070068	-0,08895	-0,07902
-0,06254	0,26156	0,085236	-0,16553	0,273764	-0,25186	-0,17011	0,112709
0,089648	0,103745	-0,11134	-0,04536	-0,05211	0,101045	0,159471	-0,09508
0,090411	0,105278	-0,20195	0,104243	0,297253	-0,15094	-0,20478	0,00398
0,367768	-0,04933	0,23736	-0,13855	0,109715	0,230168	0,091554	-0,21763
0,202282	-0,24719	0,146443	0,222848	-0,14183	0,468352	0,174352	-0,07417
0,060176	-0,12361	-0,03033	-0,22653	0,177535	0,256817	-0,05021	0,037724
0,437453	-0,12601	0,102741	0,072016	-0,21817	0,347835	-0,05829	-0,12132

-0,24713	-0,19821	-0,22646	0,214757	-0,11971	-0,16469	0,194671	-0,0352
-0,11656	-0,15792	-0,07589	-0,09984	0,07042	-0,10523	0,054586	-0,14077
0,08367	-0,114	0,099736	0,142209	-0,12296	-0,0171	-0,21343	0,199458
0,044925	0,114795	0,073206	-0,23128	0,172541	-0,1634	-0,22476	-0,07468
-0,17049	-0,10169	-0,09496	-0,18926	0,259158	-0,26042	-0,0076	0,227834
-0,10911	-0,21536	0,122537	-0,20673	-0,10418	-0,27997	-0,00144	-0,10083
0,095189	-0,14316	0,195344	-0,06787	0,141121	-0,24401	0,029251	0,121961
0,194102	-0,11167	-0,07864	-0,02041	-0,15321	-0,17197	0,226368	0,230343
0,082579	0,129456	-0,21108	0,196197	-0,08141	-0,08842	-0,0891	-0,0377
0,09436	0,159162	-0,16526	-0,17874	-0,15446	-0,31647	-0,11112	-0,10257
0,005693	-0,17758	0,058435	-0,00986	0,332976	-0,0306	0,218798	0,000367
0,219369	-0,04265	-0,1637	-0,02536	0,148156	-0,17768	-0,22277	-0,06208
-0,22974	-0,04221	-0,24764	0,194464	-0,12427	0,016386	0,167445	0,172943
0,403047	0,045358	-0,14545	0,074873	-0,21864	0,066275	-0,14412	0,164842
-0,19231	-0,16881	0,054158	-0,16009	0,178772	-0,18364	-0,18249	-0,07675
-0,00869	0,196951	-0,05444	0,068611	-0,14663	0,239533	-0,18615	-0,01321
-0,02897	0,234396	0,067185	0,095136	-0,0581	0,130457	0,083487	-0,23174
-0,16981	-0,12	-0,23352	0,054581	-0,0132	-0,16643	0,046131	0,165394
0,038088	-0,06121	-0,10747	0,234312	0,055965	-0,24593	-0,04017	0,040012
0,083428	0,151028	-0,13072	-0,09662	0,000578	-0,12566	0,218117	-0,18346

Lampiran 4. Bias Hidden Layer 1 ke Hidden Layer 2

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**

-0,08219	-0,06317	0,145093	-0,05557	-0,04539	0,168267	-0,05345	0,113948

9	10	11	12	13	14	15	16
-0,08249	-0,05543	-0,05499	-0,03225	0	-0,0052	0	0,156139

17	18	19	20	21	22	23	24
0,185809	0,165005	0,174347	0	-0,01401	0,165019	0,131673	0

25	26	27	28	29	30	31	32
0	-0,05958	0,141852	0	0	0,165747	0,146265	0,150471

Lampiran ke 5. Bobot Hidden Layer 2 ke Output

1	2	3	4	5	6	7	8
0,301422	0,285227	-0,40517	0,395593	0,582426	-0,44063	0,408409	-0,51826

9	10	11	12	13	14	15	16
0,201538	0,282222	0,382328	0,398375	0,342958	0,030584	-0,25889	-0,31158

17	18	19	20	21	22	23	24
-0,23035	-0,56982	-0,7449	0,385246	0,307862	-0,47292	-0,38535	0,401978

25	26	27	28	29	30	31	32
-0,32827	0,298272	-0,67243	0,076297	0,153481	-0,38297	-0,6867	-0,40841

Lampiran ke 6. Bias Hidden Layer 2 ke Output

1
-0,04392

Lampiran 7. Syntax Python

```
# -*- coding: utf-8 -*-
"""JST-Klasifikasi Kel 1.ipynb
```

```
Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1jUeJ0MnhEGv713ilrsDVDOzmK6qGgm-2

# Import Data
"""

import os
import pandas as pd
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing
data = pd.read_excel("/content/desa.xlsx")

data

"""#Data Understanding

"""

data.head()

data.shape

data.info()

data.describe().T

"""# EDA

## Categorical
"""

## Util functions
def plot_count(df, col):
    count = df[col].value_counts()
    plt.title(f'Value counts of {col}')
    plt.xlabel(f'{col}')
    plt.ylabel('Counts')
    plt.bar(count.index, count.values);
    return count

plot_count(data, 'Y')

"""## Numerik"""

fig, axes = plt.subplots(nrows=2, ncols=4 , figsize=(20,5))
axes[0,0].set_title("Indeks Kedalaman Kemiskinan")
axes[0,1].set_title("Indeks Keparahan Kemiskinan")
```

```
axes[0,2].set_title("Indeks Pembangunan Manusia")
axes[0,3].set_title("Pengeluaran Per Kapita Disesuaikan")
axes[1,0].set_title("Umur Harapan Hidup Saat Lahir")
axes[1,1].set_title("Angka Harapan Hidup")
axes[1,2].set_title("Harapan Lama Sekolah")

sns.distplot(data["X1"], ax=axes[0,0])
sns.distplot(data["X2"], ax=axes[0,1])
sns.distplot(data["X3"], ax=axes[0,2])
sns.distplot(data["X4"], ax=axes[0,3])
sns.distplot(data["X5"], ax=axes[1,0])
sns.distplot(data["X6"], ax=axes[1,1])
sns.distplot(data["X7"], ax=axes[1,2])

fig.tight_layout()

"""## Proportion Stage of Class

"""

# PLT sountplot target
plot_count(data, 'Y')

# SNS countplot target
plt.figure(figsize=(10,5))
sns.countplot(y=data['Y'])
sns.despine(top=True, right=True, bottom=True, left=True)
plt.tick_params(axis='both', which='both', bottom=False, top=False,
left=False)
plt.xlabel('')
plt.title('Number Per Stage')

"""## Correlation"""

plt.figure(figsize=(12,8))
plt.title('Feature corr')
sns.heatmap(data.corr());

"""# Prepocessing

## Missing Value
"""

data.isnull().sum()

data.info()

#Melihat kemiringan dari data numerik untuk imputasi
data.skew(axis=0, skipna=True)

data['X2'].fillna(data['X2'].median(),inplace=True)

data.isnull().sum()
```

```
"""## Duplication """

data.duplicated().sum()

"""## Encode"""

from sklearn.preprocessing import LabelEncoder

#Separating categorical and numerical columns
Id_col     = ['ID']
num_cols   = ['X1','X2','X3','X4','X5','X6','X7']

#Biner category columns
bin_cols   = ['Y']

#Label encoding Biner category columns
le = LabelEncoder()
for i in bin_cols :
    data[i] = le.fit_transform(data[i])

data

data.info()

"""## Scaling Min Max"""

fs=['X1','X2','X3','X4','X5','X6','X7']
df_baru=data[fs]
df_baru.head()

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
df_baru[fs] = scaler.fit_transform(df_baru[fs])
df_baru.head()

df_baru.to_excel('transform.xlsx', index=False)

datanew=pd.concat([data['Y'], df_baru], axis=1)

datanew.head()

datanew.describe().T

"""# Split data"""

feats = datanew.drop(['Y'], axis=1)
target = datanew[['Y']]

from sklearn.model_selection import train_test_split

##partition data into data training and data testing
X_train, X_val, y_train, y_val = train_test_split(feats, target,
test_size = 0.20, random_state=123)
print(X_train.shape, X_val.shape)
```

```
y_train.value_counts()

y_val.value_counts()

"""# SMOTE"""

from imblearn.over_sampling import SMOTE

#handle imbalance class using oversampling minority class with smote
method
os = SMOTE(sampling_strategy='minority',random_state =
123,k_neighbors=5)
train_smote_X,train_smote_Y = os.fit_resample(X_train,y_train)
train_smote_X = pd.DataFrame(data =
train_smote_X,columns=X_train.columns)
train_smote_Y = pd.DataFrame(data = train_smote_Y)

#Proportion before smote
y_train.value_counts()

#Proportion after smote
train_smote_Y.value_counts()

# PLT sountplot target
plt.figure(figsize=(10,5))
train_smote_Y.value_counts().plot.bar()
plt.title("Value Counts of SMOTE Y")
plt.show()

"""## Modelling"""

import tensorflow as tf

from tensorflow import random

np.random.seed(123)
random.set_seed(123)

from keras.models import Sequential
from keras.layers import Dense

jum_hidden_layer1 = 64
jum_hidden_layer2 = 32
jum_input_unit = X_train.shape[1]
jum_ouput_layer = 1
model = Sequential()
model.add(Dense(jum_hidden_layer1, input_dim=jum_input_unit,
activation='relu'))
model.add(Dense(jum_hidden_layer2, activation='relu'))
model.add(Dense(jum_ouput_layer,activation = 'sigmoid'))

model.summary()
```

```
model.compile(optimizer='Adam', loss = 'binary_crossentropy',
metrics=['accuracy'])

class mycb(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs=None):
    if(logs.get('accuracy') > 0.95 and logs.get('val_accuracy') >
0.95):
      print('\nFor Epoch', epoch, '\nAccuracy has reach = %2.2f%%'
%(logs['accuracy']*100), 'terpenuhi.'),
      self.model.stop_training =False

history = model.fit(train_smote_X,
                    train_smote_Y,
                    epochs=300,
                    validation_split=0.2,
                    shuffle=False,
                    validation_steps=10,  # berapa batch yang akan
dieksekusi pada setiap epoch
                    verbose=2,
                    callbacks=mycb()
)

"""## Menyimpan Bobot & Bias"""

first_layer_weights = model.layers[0].get_weights()[0]
first_layer_biases  = model.layers[0].get_weights()[1]
second_layer_weights = model.layers[1].get_weights()[0]
second_layer_biases  = model.layers[1].get_weights()[1]
third_layer_weights = model.layers[2].get_weights()[0]
third_layer_biases  = model.layers[2].get_weights()[1]

first_layer_weights_pd = pd.DataFrame(first_layer_weights)
first_layer_weights_pd.T.to_excel('bobotlayer1.xlsx', index=False)

first_layer_biases_pd = pd.DataFrame(first_layer_biases)
first_layer_biases_pd.T.to_excel('biaslayer1.xlsx', index=False)

second_layer_weights_pd = pd.DataFrame(second_layer_weights)
second_layer_weights_pd.T.to_excel('bobotlayer2.xlsx', index=False)

second_layer_biases_pd = pd.DataFrame(second_layer_biases)
second_layer_biases_pd.T.to_excel('biaslayer2.xlsx', index=False)

third_layer_weights_pd = pd.DataFrame(third_layer_weights)
third_layer_weights_pd.T.to_excel('bobotlayer3.xlsx', index=False)

third_layer_biases_pd = pd.DataFrame(third_layer_biases)
third_layer_biases_pd.T.to_excel('biaslayer3.xlsx', index=False)

"""# Perbandingan Plot

## Akurasi
"""
```

```
epochs = range(len(history.history['accuracy']))

plt.plot(epochs, history.history['accuracy'], label='Training
Accuracy', color='blue')
plt.plot(history.history['val_accuracy'], label='Validation
Accuracy', color='green')
plt.title('Perbandingan Ukuran Akurasi')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(loc=0)
plt.figure()

plt.show()

"""## Loss"""

plt.plot(epochs, history.history['loss'], label='Training Loss',
color='blue')
plt.plot(epochs, history.history['val_loss'], label='Validation
Loss', color = 'green')
plt.title('Perbandingan Ukuran Loss')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(loc=0)
plt.figure()

plt.show()

"""# Evaluasi Data Latih"""

pred1 = model.predict(train_smote_X)
print(pred1)

# make probability predictions with the model
prediction1 = model.predict(train_smote_X)
# round predictions
rounded1 = [round(x[0]) for x in prediction1]
rounded1

data_prediction1 = pd.DataFrame(rounded1, columns=['prediction'])
data_prediction1

train_smote_Y_new =  train_smote_Y.reset_index()

train_smote_Y_new = pd.DataFrame(train_smote_Y_new, columns=['Y'])
train_smote_Y_new

hasiltrain = pd.concat([train_smote_Y_new, data_prediction1],
axis=1)

hasiltrain

hasiltrain.to_excel('hasiltrain.xlsx', index=False)
```

```
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

y_test1 = hasiltrain['Y']
y_preds1 = hasiltrain['prediction']

print('accuracy score: ',accuracy_score(y_test1, y_preds1))
print('\n')
print('confusion matrix: \n',confusion_matrix(y_test1,y_preds1))
print('\n')
print(classification_report(y_test1, y_preds1))

"""# Evaluasi Data Uji"""

pred2 = model.predict(X_val)
print(pred2)

# make probability predictions with the model
prediction2 = model.predict(X_val)
# round predictions
rounded2 = [round(x[0]) for x in prediction2]
rounded2

data_prediction2 = pd.DataFrame(rounded2, columns=['prediction'])
data_prediction2

y_val_new =  y_val.reset_index()

y_val_new = pd.DataFrame(y_val_new, columns=['Y'])
y_val_new

hasiltest = pd.concat([y_val_new, data_prediction2], axis=1)

hasiltest

hasiltest.to_excel('hasiltest.xlsx', index=False)

from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

y_test2 = hasiltest['Y']
y_preds2 = hasiltest['prediction']

print('accuracy score: ',accuracy_score(y_test2, y_preds2))
print('\n')
print('confusion matrix: \n',confusion_matrix(y_test2,y_preds2))
print('\n')
print(classification_report(y_test2, y_preds2))
```