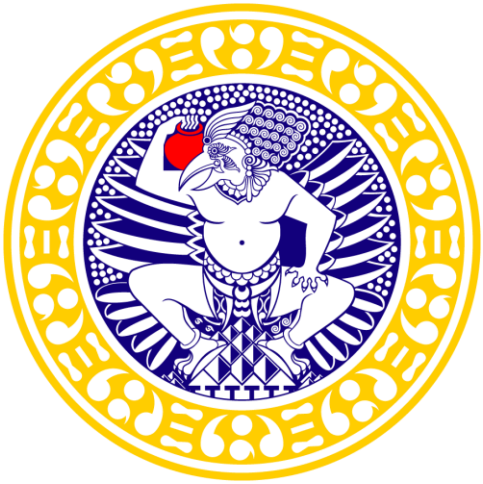
TUGAS AKHIR JARINGAN SYARAF TIRUAN

IMPLEMENTASI ALGORITMA BACKPROPAGATION UNTUK MEMPREDIKSI HARGA SAHAM PT. TELEKOMUNIKASI INDONESIA, TBK



Kelompok 2:

Ilham Farhan Zarkasyi	082011233070
Akrom Fuadi	082011233079

Dosen Pengampu:

Auli Damayanti, S.Si, M.Si.

PROGRAM STUDI S-1 MATEMATIKA

DEPARTEMEN MATEMATIKA

UNIVERSITAS AIRLANGGA

2023

KATA PENGANTAR

Segala puji syukur penulis panjatkan kehadirat Allah SWT yang telah melimpahkan taufiq, rahmat, hidayah, dan karunia-Nya, sehingga penulis dapat menyelesaikan tugas kelompok dengan judul "**Implementasi Algoritma Backpropagation untuk Memprediksi Harga Saham PT. Telekomunikasi Indonesia, Tbk**".

Keberhasilan penulis dalam menyusun tugas kelompok ini tidak lepas dari bantuan berbagai pihak yang telah memberikan doa, dukungan, bimbingan, dan masukan demi kesempurnaan penulisan ini. Oleh karena itu, pada kesempatan ini penulis mengucapkan terimakasih yang sebesar-besarnya kepada :

1. Auli Damayanti, S.Si. M.Si. selaku dosen kelas Jaringan Syaraf Tiruan yang telah memberikan ilmu, saran, dan motivasi kepada penulis.
2. Teman-teman kelas Jaringan Syaraf Tiruan, yang telah saling membantu dan memberikan dukungan selama pengerjaan tugas kelompok.

Penulis menyadari bahwa dalam penulisan tugas kelompok ini masih terdapat kekurangan dan tidak luput dari kesalahan.

Surabaya, 11 November 2023

Kelompok 2

DAFTAR ISI

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Pada saat ini peran pasar modal dalam perekonomian Indonesia mulai melembaga. Pembelian saham menjadi salah satu pilihan modal yang sah, selain bentuk modal lainnya seperti uang, tanah, dan emas. Implikasi dari adanya pilihan saham sebagai salah satu pilihan investasi, baik itu jangka panjang maupun jangka pendek, mempunyai arti yang luas karena harga saham berbeda dengan uang. Harga saham merupakan suatu masalah yang sangat penting bagi perusahaan karena mencerminkan citra perusahaan di masyarakat. Jika harga saham di perusahaan baik maka dapat dikatakan bahwa citra perusahaan baik. Harga saham suatu perusahaan menunjukkan nilai penyertaan dalam perusahaan.

Pada pasar modal yang sempurna dan efisien, harga saham mencerminkan semua informasi yang tersedia secara umum di bursa maupun informasi yang hanya dapat diperoleh dari golongan-golongan tertentu. Tinggi rendahnya harga saham dapat dipengaruhi oleh banyak faktor seperti kondisi dan kinerja perusahaan, resiko dividen, tingkat suku bunga, kondisi perekonomian, kebijaksanaan pemerintah, laju inflasi, penawaran dan permintaan serta masih banyak faktor lainnya. Karena dimungkinkan adanya perubahan faktor-faktor di atas, harga saham dapat naik atau turun. Prediksi harga akan sangat bermanfaat bagi investor untuk dapat melihat bagaimana prospek investasi saham sebuah perusahaan di masa datang. Prediksi harga saham dapat mengantisipasi naik turunnya harga saham. Dengan adanya prediksi, sangat membantu para investor dalam pengambilan keputusan. Untuk memaksimalkan keuntungan dari pasar saham, semakin banyak teknik-teknik peramalan yang digunakan oleh para investor. Pada saat ini, para investor tidak lagi bergantung pada satu teknik saja untuk mendapatkan informasi mengenai masa depan pasar saham. Teknik-teknik tersebut diantaranya adalah dengan Arch, Garch, Jaringan Syaraf Tiruan, Algoritma Genetika, dan lain-lain.

Prediksi harga saham dapat dilakukan dengan pendekatan analisis fundamental dan analisis teknikal. Prediksi yang akan dibahas disini adalah melalui pendekatan analisis teknikal dengan menggunakan Jaringan Syaraf Tiruan (JST). Jaringan Syaraf Tiruan mempelajari nilai harga saham yang lalu untuk memperoleh nilai bobot koneksi yang optimum dan menggunakan nilai bobot tersebut sebagai pengetahuan untuk menentukan

harga saham mendatang. Untuk dapat menyelesaikan suatu permasalahan, Jaringan Syaraf Tiruan memerlukan algoritma pembelajaran. Metode pembelajaran yang akan digunakan untuk memprediksi harga saham adalah dengan algoritma Backpropagation. Dengan algoritma ini, jaringan-jaringan dapat dilatih dengan menggunakan data harga saham dari situasi sebelumnya, menggolongkannya dan menyesuaikan bobot penghubung dalam jaringan sebagai input baru dan meramalkan harga saham berikutnya.

1.2 Rumusan Masalah

Rumusan masalah yang didapatkan dari latar belakang tersebut yaitu:

1. Bagaimana hasil akurasi perkiraan harga saham PT. Telekomunikasi Indonesia Tbk. pada tahap pengujian berdasarkan data penutupan saham dengan menggunakan Algoritma Backpropagation?
2. Bagaimana implementasi perkiraan harga saham PT. Telekomunikasi Indonesia Tbk. menggunakan program?

1.3 Tujuan

Tujuan yang didapatkan dari rumusan masalah tersebut yaitu:

1. Mengetahui hasil akurasi perkiraan harga saham PT. Telekomunikasi Indonesia Tbk. berdasarkan harga menggunakan Algoritma Backpropagation.
2. Mengetahui implementasi perkiraan harga saham PT. Telekomunikasi Indonesia Tbk. menggunakan program.

1.4 Manfaat

Manfaat dari makalah tugas akhir ini yaitu:

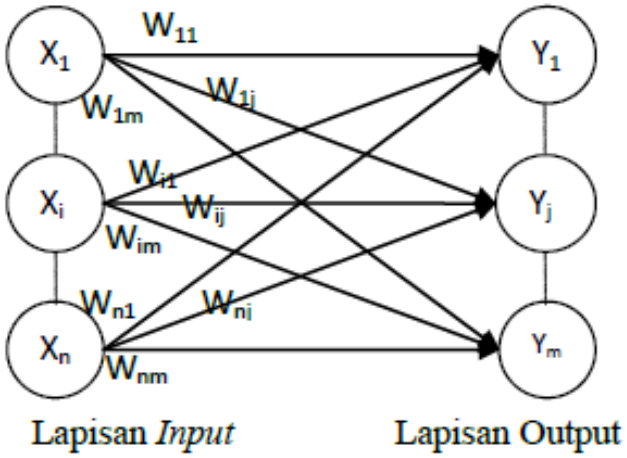
1. Dapat dijadikan masukan untuk para investor dalam mengambil keputusan dalam berinvestasi saham.
2. Dapat digunakan untuk referensi pada penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

2.1 Jaringan Syaraf Tiruan

Jaringan syaraf tiruan, atau biasa disebut sebagai neural network dalam bahasa Inggris, adalah model matematika dan komputasi yang terinspirasi oleh struktur dan fungsi jaringan saraf biologis manusia. Model ini digunakan dalam bidang kecerdasan buatan (artificial intelligence) untuk meniru kemampuan pemrosesan informasi dan pembelajaran mesin yang mirip dengan cara kerja otak manusia. Jaringan syaraf tiruan terdiri dari unit pemrosesan kecil yang disebut neuron buatan atau simpul (node), yang terhubung dalam lapisan-lapisan. Neuron-neuron ini menerima masukan, memproses informasi dengan berbagai bobot (weights) dan fungsi aktivasi, dan menghasilkan keluaran. Bobot-bobot ini diatur dan disesuaikan melalui proses pembelajaran, yang memungkinkan jaringan syaraf tiruan untuk memahami pola-pola kompleks dalam data, seperti klasifikasi gambar, prediksi, pengenalan suara, dan banyak lagi.

Jaringan syaraf tiruan telah menjadi alat yang sangat berguna dalam berbagai aplikasi, termasuk pengenalan wajah, pengenalan suara, analisis teks, permainan video, kendaraan otonom, dan banyak lagi. Mereka sangat populer karena kemampuan mereka untuk belajar dari data dan mengatasi masalah yang sulit ditemukan solusinya dengan pendekatan konvensional.

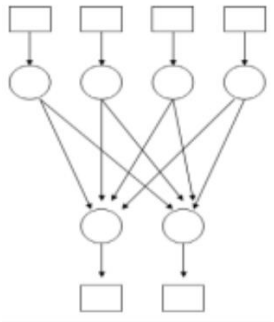


Gambar Arsitektur Jaringan Syaraf Tiruan

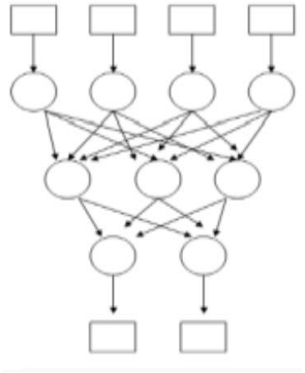
Arsitektur Jaringan Syaraf Tiruan dibagi menjadi 3 jenis yaitu:

1. Single Layer
   - Hanya memiliki satu lapisan dengan bobot-bobot terhubung.
   - Langsung menerima input dan mengolahnya menjadi output tanpa menggunakan hidden layer
2. Multi Layer
   - Memiliki satu atau lebih lapisan input, satu atau lebih lapisan output, dan lapisan tersembunyi
   - Dapat menyelesaikan masalah yang lebih kompleks karena lebih akurat
   - Fungsi pembelajarannya lebih rumit
3. Competitive Model / Recurrent Model
   - Hubungan antar neuron tidak diperlihatkan secara langsung pada arsitektur
   - Hubungan antar neuron dapat digambarkan sebagai jaring yang rumit

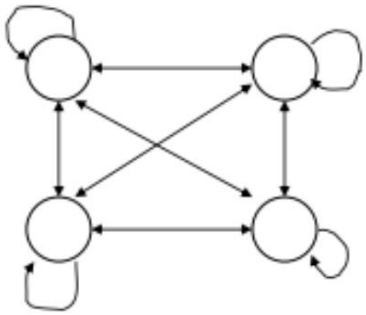
Berikut adalah 3 model dari Jaringan Syaraf Tiruan yang sesuai dengan arsitekturnya:



Model JST Single Layer



Model JST Multi Layer



Model JST Competitive Layer

2.2 Fungsi Aktivasi

Fungsi aktivasi mempunyai tujuan untuk menentukan besarnya bobot. Terdapat beberapa fungsi aktivasi yang sering digunakan sebagai berikut

a. Fungsi Sigmoid Biner

Fungsi sigmoid biner banyak digunakan pada neural network dengan algoritma backpropogation. Fungsi ini mempunyai nilai antara 0 sampai 1. Sehingga bagus dipakai pada jaringan yang mempunyai nilai output antara 0 sampai 1. Fungsi sigmoid biner dapat dirumuskan sebagai berikut:

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

Dengan x adalah hasil perkalian bobot dan input ditambah bias

b. Fungsi Linear

Fungsi linear mempunyai nilai keluaran yang sama dengan nilai masukannya. Fungsi linear dapat dirumuskan sebagai berikut:

$$y = x$$

Dengan x adalah hasil perkalian bobot dan input ditambah bias, dan y adalah output

2.3 Normalisasi dan Denormalisasi

Normalisasi dan Denormalisasi adalah dua konsep penting dalam pengolahan data, khususnya dalam konteks pemrosesan dan analisis data, termasuk dalam machine learning dan basis data. Berikut adalah penjelasan singkat dari kedua konsep tersebut:

- Normalisasi
  - Definisi: Normalisasi adalah proses transformasi data dalam skala tertentu sehingga nilai-nilai tersebut berada dalam rentang tertentu atau format yang lebih standar. Tujuannya adalah untuk memastikan bahwa data memiliki skala yang seragam atau rentang tertentu yang optimal untuk analisis atau pengolahan lebih lanjut.
  - Contoh: Dalam konteks machine learning, normalisasi sering digunakan untuk mengubah rentang variabel numerik. Misalnya, rentang nilai dari satu fitur (fitur A) mungkin antara 0 dan 100, sedangkan fitur lainnya (fitur B) mungkin antara 0 dan 1000. Dengan normalisasi, kedua fitur tersebut dapat diubah sehingga keduanya memiliki rentang antara 0 dan 1, memudahkan proses pelatihan model.
- Denormalisasi
  - Definisi: Denormalisasi adalah kebalikan dari normalisasi. Ini adalah proses mengembalikan data yang telah dinormalisasi ke format atau skala aslinya. Tujuan utamanya mungkin untuk memfasilitasi proses tertentu, seperti penyimpanan data dalam basis data atau presentasi data kepada pengguna dalam format yang lebih familiar.
  - Contoh: Setelah proses pelatihan model dalam machine learning, Anda mungkin telah menormalisasi data masukan untuk model. Namun, ketika model tersebut digunakan untuk prediksi di dunia nyata, data masukan baru yang diterima mungkin perlu denormalisasi sebelum dimasukkan ke dalam model untuk mendapatkan hasil prediksi dalam skala yang asli.

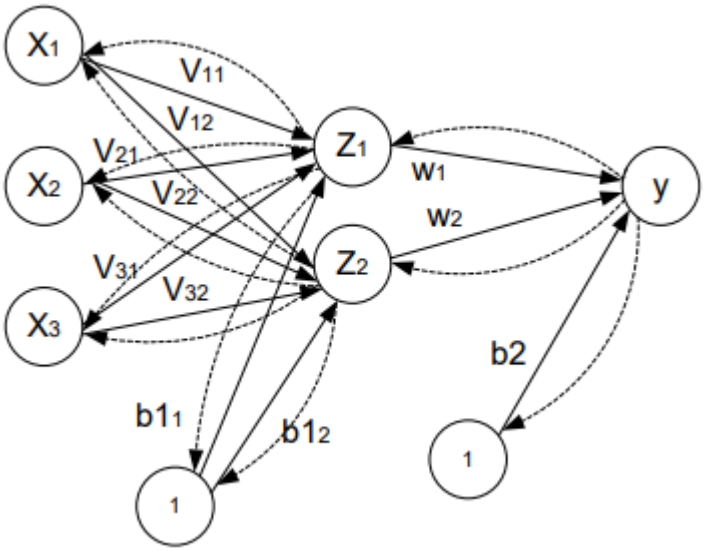
2.4 Algoritma Backpropagation

Algoritma backpropagation adalah teknik yang digunakan dalam pelatihan (training) jaringan syaraf tiruan (neural networks) untuk memperbaiki bobot (weights) yang menghubungkan neuron-neuron dalam jaringan tersebut. Tujuan dari algoritma ini adalah untuk mengurangi kesalahan prediksi atau output jaringan terhadap data pelatihan dengan mengadjust bobot-bobot tersebut.

Keunggulan paling menonjol dari algoritma Backpropagation adalah:

- Cepat, sederhana dan mudah diprogram
- Tidak memiliki parameter tuning selain dari jumlah input
- Fleksibel karena tidak memerlukan pengetahuan mengenai neTwork sebelumnya
- Metode standar yang umumnya bekerja dengan baik
- Tidak perlu fitur khusus dari fungsi yang akan dipelajari.

Berikut adalah contoh arsitektur jaringan dari algoritma Backpropagation:



Gambar Arsitektur Jaringan Algoritma Backpropagation

- Pada gambar, unit input dilambangkan dengan X, hidden unit dilambangkan dengan Z, dan unit output dilambangkan dengan Y.
- Bobot antara X dan Z dilambangkan dengan v sedangkan bobot antara Z dan Y dilambangkan dengan **w.**

Pada intinya, pelatihan dengan metode backpropagation terdiri dari tiga langkah, yaitu:

- Data dimasukkan ke input jaringan (feedforward)
- Perhitungan dan propagasi balik dari error yang bersangkutan
- Pembaharuan (adjustment) bobot dan bias.

Saat umpan maju (feedforward), setiap unit input (Xi) akan menerima sinyal input dan akan menyebarkan sinyal tersebut pada tiap hidden unit (Zj). Setiap hidden unit kemudian akan menghitung aktivasinya dan mengirim sinyal (zj) ke tiap unit output. Kemudian setiap unit output (Yk) juga akan menghitung aktivasinya (yk) untuk menghasilkan respons terhadap input yang diberikan jaringan.

2.5 Ukuran Kesalahan

Ukuran kesalahan suatu prakiraan digunakan untuk memastikan prakiraan berjalan sesuai yang diharapakan. Salah satu ukuran standar yang digunakan untuk menentukan akurasi peramalan adalah Mean Square Error (MSE) dengan rumus sebagai berikut:

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(X_t - y_t)^2$$

Dengan n adalah jumlah data, y_t adalah nilai output (prediksi) pada periode ke-t, dan X_t adalah nilai data aktual atau observasi pada periode ke-t.

2.6 Indeks Harga Saham

Saham adalah satuan nilai atau pembukuan dalam berbagai instrumen finansial yang mengacu pada bagian kepemilikan sebuah perusahaan. Menurut Kusnadi, dkk (1999:94), “Saham adalah sertifikat atau tanda otentik yang mempunyai kekuatan hukum bagi pemegangnya sebagai keikutsertaan di dalam perusahaan serta mempunyai nilai nominal (mata uang) serta dapat diperjualbelikan.” Saham merupakan surat berharga yang bersifat kepemilikan. Artinya si pemilik saham merupakan pemilik perusahaan. Semakin besar saham yang dimiliki, maka semakin besar pula kekuasaannya di perusahaan tersebut. Bentuk fisik saham adalah selembar kertas, pada saham tersebut dinyatakan bahwa pemegang saham adalah pemilik perusahaan. Menerbitkan saham merupakan salah satu pilihan perusahaan ketika memutuskan untuk pendanaan perusahaan. Saham dapat didefinisikan juga sebagai tanda penyertaan modal seseorang atau pihak (badan usaha) dalam suatu perusahaan atau perseroan terbatas. Indeks harga saham adalah indikator atau cerminan pergerakan harga saham. Indeks merupakan salah satu pedoman bagi investor untuk melakukan investasi di pasar modal khususnya saham. Indeks dapat memberikan investor gagasan secara cepat tentang bagaimana kinerja sebuah bursa selama waktu tertentu. Dengan melihat indeks, maka investor dapat memperkirakan dengan cepat bagaimana kinerja portofolio sahamnya.

Salah satu indeks harga saham yang paling terkenal adalah Dow Jones Industrial Average (DJIA) di Amerika Serikat. Indeks ini mencakup 30 perusahaan besar dan beragam dari berbagai sektor ekonomi, dan perubahan dalam DJIA digunakan untuk mengukur kesehatan ekonomi Amerika Serikat secara umum. Selain DJIA, ada juga S&P 500, yang mencakup 500 perusahaan terkemuka di Amerika Serikat dan memberikan pandangan yang lebih luas tentang kinerja pasar. Di luar Amerika Serikat, terdapat juga indeks seperti Nikkei 225 di Jepang, FTSE 100 di Inggris, dan banyak indeks lainnya di seluruh dunia. Indeks harga saham sangat penting dalam analisis pasar keuangan, investasi, dan perencanaan keuangan. Para investor dan analis menggunakan indeks ini untuk melacak kinerja investasi mereka, mengukur risiko, dan membuat keputusan investasi.

Indeks harga saham juga dapat memberikan indikasi umum tentang kondisi ekonomi suatu negara; ketika indeks naik, itu bisa mengindikasikan pertumbuhan ekonomi yang sehat, sementara penurunan indeks bisa menjadi tanda perlambatan ekonomi atau ketidakpastian. Penting untuk diingat bahwa indeks harga saham hanya mencerminkan kinerja perusahaan-perusahaan yang termasuk dalam sampel indeks tersebut, dan kinerja individual saham dalam indeks dapat bervariasi. Oleh karena itu, investor yang ingin menginvestasikan uangnya di pasar saham seringkali memerlukan pemahaman yang mendalam tentang perusahaan-perusahaan yang terdapat dalam indeks serta faktor-faktor yang mempengaruhi pergerakan harga saham mereka.

2.7 Python

Python adalah bahasa pemrograman tujuan umum yang ditafsirkan, tingkat tinggi. Dibuat oleh Guido van Rossum dan pertama kali dirilis pada tahun 1991, filosofi desain Python menekankan keterbacaan kode dengan penggunaan spasi putih yang signifikan. Konstruksi bahasanya dan pendekatan berorientasi objek bertujuan untuk membantu pemrogram menulis kode yang jelas dan logis untuk proyek skala kecil dan besar.

Python diketik secara dinamis dan pengumpulan sampah. Ini mendukung beberapa paradigma pemrograman, termasuk pemrograman terstruktur (terutama, prosedural), berorientasi objek, dan fungsional. Python sering dideskripsikan sebagai bahasa "termasuk baterai" karena perpustakaan standarnya yang komprehensif. Python dibuat pada akhir 1980-an sebagai penerus bahasa ABC. Python 2.0, dirilis pada tahun 2000, memperkenalkan fitur-fitur seperti pemahaman daftar dan sistem pengumpulan sampah dengan penghitungan referensi.

BAB III
METODOLOGI PENELITIAN

Berikut adalah beberapa prosedur untuk Memprediksi Harga Saham PT. Telekomunikasi Indonesia, Tbk dengan menggunakan Algoritma Backpropagation:

1. Mengkaji materi tentang algoritma Backpropagation
2. Menerapkan algoritma Backpropagation untuk memprediksi harga saham
   - Mencari data
   - Melakukan preprocessing data (normalisasi)
   - Membagi data menjadi data training dan data testing
   - Melakukan proses pelatihan dengan algoritma Backpropagation
   - Melakukan proses pengujian dengan algoritma Backpropagation
3. Membuat program dari langkah-langkah metode penelitian di atas
4. Mengimplementasikan program pada persoalan yang dihadapi

Selanjutnya, secara garis besar, algoritma backpropagation dibagi ke dalam 2 tahap, yaitu tahap training dan testing. Berikut adalah langkah-langkahnya:

1. Tahap Training Data

Pelatihan Backpropagation dilakukan melalui langkah-langkah berikut ini :

1. Inisialisasi bobot.
2. Selama kondisi berhenti bernilai salah, kerjakan langkah 3-10.
3. Untuk setiap data training, lakukan 4-9.
4. Untuk langkah 4 hingga 6 merupakan Proses Umpan Maju (Feedforward) Setiap unit input ($X_p i = 1, \dots, n$) menerima sinyal input dan menyebarkan sinyal tersebut ke seluruh unit tersembunyi.
5. Pada setiap unit tersembunyi $\left(Z_j j = 1, \dots, p\right)$ menjumlahkan sinyal-sinyal input yang sudah berbobot (termasuk biasnya)

$$z_{in_j} = v_{0j} + \sum_{i=1}^{n} x_i . v_{ij}$$

Lalu menghitung sinyal output dari unit tersembunyi dengan menggunakan fungsi aktivasi yang telah ditentukan $z_j = f(z_{in_j})$. Sinyal output ini selanjutnya dikirim ke seluruh unit pada unit diatas (unit output).

6. Tiap-tiap output $(y_k, k = 1, \dots, m)$ menjumlahkan bobot sinyal input

$$y_{in_k} = w_{0k} + \sum_{i=1}^{p} z_i . w_{jk}$$

Lalu menghitung sinyal output dari unit output bersangkutan dengan menggunakan fungsi aktivasi yang telah ditentukan $y_k = f(y_{in_k})$. Sinyal output ini selanjutnya dikirim ke seluruh unit output.

7. Untuk langkah 7 hingga 8 merupakan Proses Umpan Mundur (Backward/ Propagasi Error). Setiap unit output $(y_k, k = 1, \dots, m)$ menerima suatu pola target yang sesuai dengan pola input pelatihan, untuk menghitung kesalahan (error) antara target dengan output yang dihasilkan jaringan

$$\delta_k = (t_k - y_y) f'(y_{in_k})$$

Faktor δ_k digunakan untuk menghitung koreksi error (Δw_{jk}) yang nantinya akan dipakai untuk memperbaiki w_{jk}dimana :

$$\Delta w_{jk} = \alpha \delta_k z_j$$

Selain itu juga dihitung koreksi bias (Δw_{0k}) yang nantinya akan dipakai untuk memperbaiki w_{0k} dimana :

$$(\Delta w_{0k}) = \alpha \delta_k$$

Faktor δ_k kemudian dikirimkan ke lapisan yang berada pada langkah 8.

8. Setiap unit tersembunyi $(Z_j j = 1, \dots, p)$ menerima input delta dari langkah 7 yang sudah berbobot :

$$\delta_{in_j} = \sum_{k=1}^{m} \delta_k w_{jk}$$

Kemudian hasilnya dikalikan dengan turunan dari fungsi aktivasi yang digunakan untuk menghitung informasi kesalahan error δ_i dimana :

$$\delta_j = \delta_{in_j} f'\left(z_{in_j}\right)$$

Kemudian hitunglah koreksi bobot dengan :

$$\Delta v_{ij} = \alpha \delta_j x_i$$

Kemudian hitunglah koreksi bias :

$$\Delta v_{0j} = \alpha \delta_j$$

9. Untuki langkah 7 hingga 8 merupakan Proses Update Bobot dan Bias. Setiap unit output $(y_k, k = 1, \dots, m)$ memperbaiki bobot dan bias dari setiap unit tersembunyi $(j = 0, \dots, p)$

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

10. Tes kondisi berhenti apabila error ditemukan. Jika kondisi berhenti terpenuhi, maka pelatihan jaringan dapat dihentikan.

2. Tahap Testing Data

Pengujian dilakukan melalui feedforward dengan langkah-langkah sebagai berikut :

1. Inisialisasi bobot (hasil pelatihan).
2. Untuk setiap vektor input, kerjakan langkah 3-5.
3. Untuk $i = 1, \dots, n$ set aktivasi unit input X_1 .
4. Untuk $j = 1, \dots, p$

$$z_{in_j} = v_{0j} + \sum_{i=1}^{n} x_i . v_{ij}$$

$$z_j = f(z_{in_j})$$

5. Untuk $k = 1, \dots, p$

$$y_{in_k} = w_{0k} + \sum_{i=1}^{p} z_i . w_{jk}$$

$$y_k = f(y_{in_k})$$

Setelah dilakukan proses training dan testing pada pola yang dilatih, maka akan diperoleh hasil bahwa pengujian terhadap pola-pola tersebut apakah telah benar/akurat atau sebaliknya. Untuk menghitung rata-rata error (MSE) jaringan, dapat dilakukan dengan rumus:

$$MSE = \sum_{I=1}^{N} \frac{(y_i - y_n)^2}{N}$$

Dimana :

y_i : Nilai aktual data (target)

y_n : Nilai hasil prediksi (aktual output)

N : Jumlah data yang diujikan

Sedangkan untuk proses denormalisasi atau mengembalikan kembali nilai hasil prediksi jaringan ke bentuk data semula (sebelum normalisasi) dapat menggunakan rumus sebagai berikut :

$$x_i = y_n(x_{max} - x_{\min}) + x_{min}$$

Dimana :

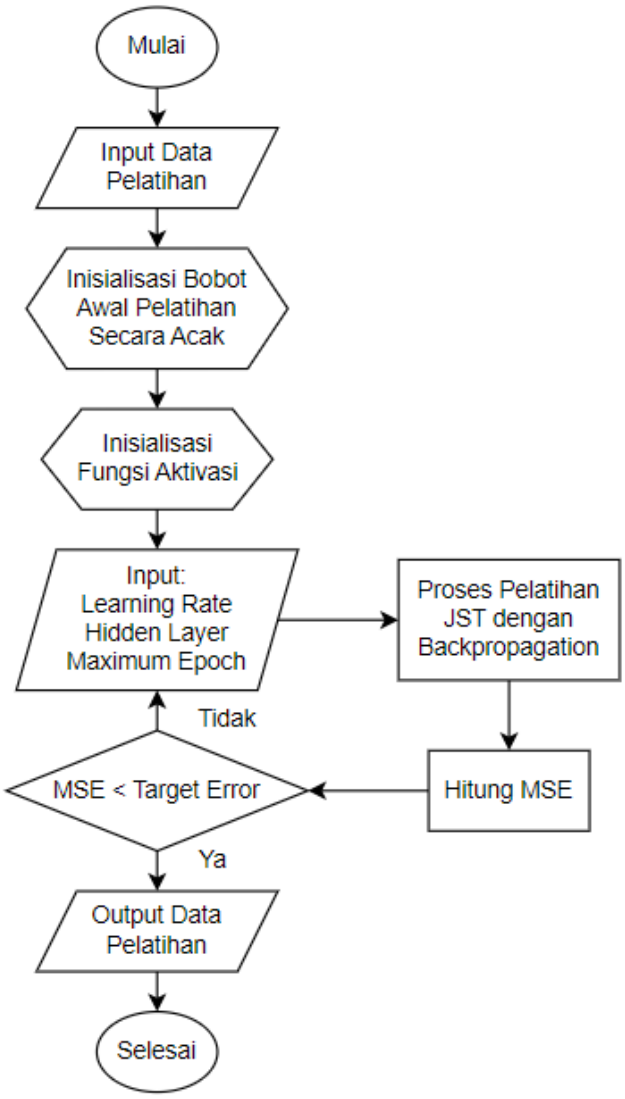
x_i : Nilai X yang akan dilakukan denormalisasi

y_n : Nilai hasil prediksi (aktual output) yang sesuai dengan x_i

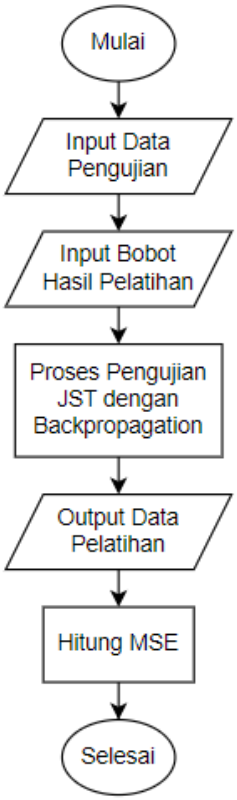
x_{max} : Nilai maksimum pada barisan X

x_{min} : Nilai minimum pada barisan X

Berikut adalah flowchart dari tahap training dan testing:



Flowchart Tahap Training



Flowchart Tahap Testing

BAB IV
PEMBAHASAN

4.1 Perhitungan Manual

Berikut adalah data yang didapatkan dari https://id.investing.com/equities/telkom-indones-historical-data:

Tanggal	Terakhir	Pembukaan	Tertinggi	Terendah	Vol.	Perubahan%
12/01/2023	3,83	3,79	3,84	3,79	81,04M	1,86%
30/11/2023	3,76	3,75	3,82	3,74	258,52M	1,08%
29/11/2023	3,72	3,75	3,77	3,7	56,47M	-1,33%
28/11/2023	3,77	3,68	3,8	3,68	106,83M	2,45%
27/11/2023	3,68	3,64	3,73	3,64	73,89M	1,66%
24/11/2023	3,62	3,6	3,62	3,57	46,92M	0,56%
23/11/2023	3,6	3,61	3,63	3,59	57,39M	-0,28%
22/11/2023	3,61	3,61	3,63	3,59	34,56M	-0,28%
21/11/2023	3,62	3,59	3,62	3,58	68,26M	0,84%
20/11/2023	3,59	3,56	3,6	3,54	49,53M	1,13%
17/11/2023	3,55	3,53	3,55	3,48	55,59M	0,28%
16/11/2023	3,54	3,55	3,55	3,51	68,99M	-0,28%
15/11/2023	3,55	3,52	3,57	3,52	114,10M	1,43%
14/11/2023	3,5	3,5	3,54	3,5	45,99M	-0,57%
13/11/2023	3,52	3,57	3,57	3,51	47,73M	-0,85%
11/10/2023	3,55	3,52	3,55	3,49	48,62M	0,57%

Berdasarkan data yang telah diperoleh, akan dilakukan penyelesaian secara manual untuk menyelesaikan permasalahan memprediksi harga saham. Untuk data diperoleh dari internet harga saham PT. Telekomunikasi, Tbk. Berikut data yang diperoleh, dengan data 1 hingga 7 merupakan data training dan data 8 hingga 10 merupakan data testing.

1. **Langkah 1 : Rotasi Data**

   Langkah awal melakukan rotasi data. Diketahui nilai maximal sebesar 3770, nilai minimal sebesar 3500, dan untuk nilai selisih antara nilai maximal dan nilai minimal sebesar 270. Dapat dilihat pada tabel berikut:

No	Input										Y
	1	2	3	4	5	6	7	8	9	10	Target
1	3570	3650	3570	3520	3530	3550	3520	3500	3550	3540	3550
2	3650	3570	3520	3530	3550	3520	3500	3550	3540	3550	3620
3	3570	3520	3530	3550	3520	3500	3550	3540	3550	3620	3610
4	3520	3530	3550	3520	3500	3550	3540	3550	3590	3620	3610
5	3530	3550	3520	3500	3550	3540	3550	3590	3620	3610	3600

No	Input										Y
	1	2	3	4	5	6	7	8	9	10	Target
6	3550	3520	3500	3550	3540	3550	3590	3620	3610	3600	3620
7	3520	3500	3550	3540	3550	3590	3620	3610	3600	3620	3680
8	3500	3550	3540	3550	3590	3620	3610	3600	3620	3680	3770
9	3550	3540	3550	3590	3620	3610	3600	3620	3680	3770	3720
10	3540	3550	3590	3620	3610	3600	3620	3680	3770	3720	3760

2. Langkah 2 : Normalisasi Data

Menghitung nilai normalisasi dengan rumus sebagai berikut:

$$x = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Sebagai contoh pada perhitungan data pertama sebagai berikut:

$$x = \frac{x - x_{min}}{x_{max} - x_{min}} = \frac{3570 - 3500}{3770 - 3500} = 0{,}2593$$

Dengan cara yang sama dilakukan perhitungan pada data lainnya sehingga diperoleh data hasil normalisasi sebagai berikut :

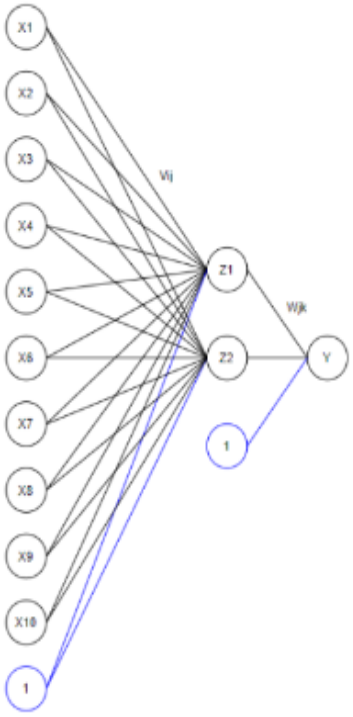
No	Input										Y
	1	2	3	4	5	6	7	8	9	10	Target
1	0.259	0.555	0.259	0.074	0.111	0.185	0.074	0.000	0.185	0.148	0.185
2	0.555	0.259	0.074	0.111	0.185	0.074	0.000	0.185	0.148	0.185	0.444
3	0.259	0.074	0.111	0.185	0.074	0.000	0.185	0.148	0.185	0.444	0.407
4	0.074	0.111	0.185	0.074	0.000	0.185	0.148	0.185	0.333	0.444	0.407
5	0.111	0.185	0.074	0.000	0.185	0.148	0.185	0.333	0.444	0.407	0.370
6	0.185	0.074	0.000	0.185	0.148	0.185	0.333	0.444	0.407	0.370	0.444
7	0.074	0.000	0.185	0.148	0.185	0.333	0.444	0.407	0.370	0.444	0.666
8	0.000	0.185	0.148	0.185	0.333	0.444	0.407	0.370	0.444	0.666	1.000
9	0.185	0.148	0.185	0.333	0.444	0.407	0.370	0.444	0.666	1.000	0.814
10	0.148	0.185	0.333	0.444	0.407	0.370	0.444	0.666	1.000	0.814	0.963

3. Langkah 3 : Inisialisasi

Melakukan inisialisasi, sehingga didapatkan sebagai berikut:

- Jumlah neuron pada input layer : 10
- Jumlah neuron pada hidden layer : 2
- Jumlah neuron pada output layer : 1
- Learning Rate : 0.5
- Maksimum Epoch : 1
- Target Error : 0.01

Berikt adalah ilustrasinya:



4. Langkah 4 : Melakukan Perhitungan Feed Forward

Membangkitkan bobot awal secara random

input ke hidden			**hidden ke output**	
V	Z1	Z2	W	Y
X1	0.8638	0.3452	Z1	0.1625
X2	0.3722	0.6899	Z2	0.3020
X3	0.8126	0.0793	W	Y
X4	0.7979	0.2217	0	0.7015
X5	0.1184	0.0615		
X6	0.8222	0.5811		
X7	0.5418	0.0512		
X8	0.6835	0.8198		
X9	0.9767	0.5849		
X10	0.7938	0.4835		
V	Z1	Z2		
0	0.6012	0.4543		

Melakukan perhitungan data pertama sebagai berikut

Menghitung bobot hidden layer

$$Z_{in_J} = V_{0J} + \sum_{i=1}^{n} X_i V_{ij}$$

$$Z_{in_1} = V_{01} + \sum_{i=1}^{n} X_i V_{i1}$$

$$Z_{in_1} = 0.6012 + ((0.2593 \times 0.8638) + (0.5556 \times 0.3722) + (0.2593 \times 0.8126) + (0.0741 \times 0.7979) + (0.1111 \times 0.1184) + (0.1852 \times 0.8222) + (0.0741 \times 0.5418) + (0.0000 \times 0.6835) + (0.1852 \times 0.9767) + (0.1481 \times 0.7938)$$

$$Z_{in_1} = 1.8057$$

Lakukan dengan cara yang sama untuk data berikutnya.

Lalu, menghitung fungsi aktivasi sigmoid

$$Z_j = \frac{1}{1 + exp^{(-Z_{in_J})}}$$

$$Z_1 = \frac{1}{1 + exp^{(1.8057)}} = 0.8588$$

Lakukan dengan cara yang sama untuk nilai Z_2 dan seterusnya.

Lalu, menghitung bobot output layer

$$Y_{in} = W_{0k} + \sum_{j=1}^{p} Z_j W_{jk}$$

$$Y_{in} = W_{01} + \sum_{j=1}^{2} Z_j W_{j1}$$

$$Y_{in} = 0.7015 + ((0.8588 \times 0.1625) + (0.7794 \times 0.3020))$$

$$Y_{in} = 1.0765$$

Lalu, menghitung fungsi aktivasi sigmoid

$$Y_k = \frac{1}{1 + exp^{(-Y_{in})}}$$

$$Y_k = \frac{1}{1 + exp^{(1.0765)}} = 0.7458$$

5. Langkah 5 : Melakukan Perhitungan Backward

Menghitung faktor kesalahan (δ) - Output layer

$$\delta_k = (t_k - Y_k) f'(Y_{net_k}) = (t_k - Y_k) Y_k (1 - Y_k)$$

Dengan nilai $t_k = 0.1852$ maka,

$$\delta_k = (0.1852 - 0.7458) \times 0.7458 \times (1 - 0.7458) = -0.1063$$

Menghitung koreksi bobot

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

$\Delta W_1 = 0.5 \times (-0.1063) \times Z_1$

$\Delta W_1 = 0.5 \times (-0.1063) \times 0.8588$

$\Delta W_1 = -0.0456$

Menghitung koreksi bias

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

$\Delta W_0 = 0.5 \times (-0.1063) = -0.0531$

Menghitung faktor kesalahan (δ) - hidden layer

$$\delta_{net_j} = \sum_{k=1}^{m} \delta_k W_{jk}$$

$\delta_{net} = (-0.1063 \times 0.1625) + (-0.1063 \times 0.3020) = -0.0494$

Faktor kesalahan setiap neuron pada hidden layer

$\delta_j = \delta_{net_j} f'(Z_{net_j}) = \delta_{net_j} Z_j (1 - Z_j)$

$\delta_j = \delta_{net} Z_1 (1 - Z_1)$

$\delta_j = -0.0494 \times 0.8588 \times (1 - 0.8588) = -0.0060$

Menghitung perubahan bobot yang digunakan untuk memperbaiki V_{ij}

Sebagai contoh perhitungan ΔV_{11} :

$\Delta V_{ij} = \alpha \delta_j V_{ij}$

$\Delta V_{11} = \alpha \delta_1 V_{11} = 0.5 \times (-0.0060) \times 0.8638 = -0.0026$

Menghitung perubahan bobot bias yang digunakan untuk memperbaiki V_{0j}

$\Delta V_{0j} = \alpha \delta_j$

$\Delta V_{01} = 0.5 \times \delta_1 \times V_{01} = 0.5 \times (-0.0060) = -0.0030$

6. Langkah 6 : Update Bobot dan Bias

Update W_{jk} baru

W_{jk} baru $= W_{jk}$ lama $+ \Delta W_{jk}$

W_1 baru $= W_1$ lama $+ \Delta W_1 = 0.1625 + (-0.0456) = 0.1169$

W_2 baru $= W_2$ lama $+ \Delta W_2 = 0.3020 + (-0.0414) = 0.2606$

W_0 baru $= W_0$ lama $+ \Delta W_0 = 0.7015 + (-0.0531) = 0.6484$

Update V_{ij} baru

$V_{ij} baru = V_{ij} lama + \Delta V_{ij}$

$V_{11} baru = V_{11} lama + \Delta V_{11} = 0.8638 - 0.0026 = 0.8613$

$$V_{21}baru = V_{21}lama + \Delta V_{21} = 0.3722 - 0.0011 = 0.3711$$
$$V_{31}baru = V_{31}lama + \Delta V_{31} = 0.8126 - 0.0024 = 0.8102$$
$$V_{41}baru = V_{41}lama + \Delta V_{41} = 0.7979 - 0.0024 = 0.7955$$
$$V_{51}baru = V_{51}lama + \Delta V_{51} = 0.1184 - 0.0004 = 0.1181$$
$$V_{61}baru = V_{61}lama + \Delta V_{61} = 0.8222 - 0.0025 = 0.8197$$
$$V_{71}baru = V_{71}lama + \Delta V_{71} = 0.5418 - 0.0016 = 0.5402$$
$$V_{81}baru = V_{81}lama + \Delta V_{81} = 0.6835 - 0.0020 = 0.6814$$
$$V_{91}baru = V_{91}lama + \Delta V_{91} = 0.9767 - 0.0029 = 0.9738$$
$$V_{101}baru = V_{101}lama + \Delta V_{101} = 0.7938 - 0.0024 = 0.7914$$
$$V_{01}baru = V_{01}lama + \Delta V_{01} = 0.6012 - 0.0030 = 0.5982$$
$$V_{12}baru = V_{12}lama + \Delta V_{12} = 0.3452 - 0.0015 = 0.3437$$
$$V_{22}baru = V_{22}lama + \Delta V_{22} = 0.6899 - 0.0029 = 0.6870$$
$$V_{32}baru = V_{32}lama + \Delta V_{32} = 0.0793 - 0.0003 = 0.790$$
$$V_{42}baru = V_{42}lama + \Delta V_{42} = 0.2217 - 0.0009 = 0.2208$$
$$V_{52}baru = V_{52}lama + \Delta V_{52} = 0.0615 - 0.0003 = 0.0612$$
$$V_{62}baru = V_{62}lama + \Delta V_{62} = 0.5811 - 0.0025 = 0.5786$$
$$V_{72}baru = V_{72}lama + \Delta V_{72} = 0.0512 - 0.0002 = 0.0509$$
$$V_{82}baru = V_{82}lama + \Delta V_{82} = 0.8198 - 0.0035 = 0.8164$$
$$V_{92}baru = V_{92}lama + \Delta V_{92} = 0.5849 - 0.0025 = 0.5825$$
$$V_{102}baru = V_{102}lama + \Delta V_{102} = 0.4835 - 0.0021 = 0.4815$$
$$V_{02}baru = V_{02}lama + \Delta V_{02} = 0.4543 - 0.0042 = 0.4501$$

7. Langkah 7 : Menghitung Nilai MSE Training

Dengan melakukan langkah 4, 5, dan 6 terhadap semua data yang ada (setiap data menggunakan bobot terakhir yang didapatkan dari perhitungan dari data sebelumnya) lalu akan didapatkan bobot terakhir dari perhitungan data training yang terakhir dan akan dihitung MSE nya dari target yang dihasilkan dari setiap data yang telah dihitung.

Menghitung Nilai MSE pada Data Training yang disajikan dengan tabel berikut:

Target	Perhitungan
0.1852	0.7458
0.4444	0.7212
0.4074	0.7067

Target	Perhitungan
0.4074	0.6932
0.3704	0.6791
0.4444	0.6614
0.6667	0.6476

Dapat dihitung dengan rumus berikut:

$$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n}(X_t - Y_t)^2$$

$$\text{MSE} = \frac{(0.1852 - 0.7458)^2 + (0.4444 - 0.7212)^2 + (0.4047 - 0.7067)^2 + \ldots + (0.6667 - 0.6476)^2}{7}$$

$$\text{MSE} = \frac{0.7049}{7} = 0.1007$$

8. Langkah 8 : Cek Pemberhenti

Karena sudah mencapai batas epoch maka selesai, lanjut ke tahap testing

9. Langkah 9 : Menghitung Nilai MSE Testing

Setelah mengecek pemberhenti iterasi selanjutnya masuk ke tahap testing dimana setiap data testing akan dihitung dengan langkah yang sama seperti tahap training dan hanya sampai pada tahap feed forward saja karena pada tahap testing tidak diperlukan update bobot maka tahap backward juga tidak diperlukan pada testing data. Tahap testing ini semua datanya akan memakai bobot terakhir yang sudah didapatkan dari tahap training. Selanjutnya setiap data testing yang dihitung sampai tahap feed forward akan dicari nilai MSE-nya.

Menghitung Nilai MSE pada Data Testing yang disajikan dengan tabel berikut:

Target	Perhitungan
1.0000	0.6502
0.8148	0.6754
0.9630	0.6857

Dapat dihitung dengan rumus berikut

$$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n}(X_t - Y_t)^2$$

$$\text{MSE} = \frac{(1.000 - 0.6502)^2 + (0.8148 - 0.6754)^2 + (0.9630 - 0.6857)^2}{3}$$

$$\text{MSE} = \frac{0.2187}{3} = 0.0729$$

10. Langkah 10 : Data prediksi untuk 5 hari berikutnya

Karena MSE testing kurang dari 10% maka dapat dilanjutkan untuk tahap prediksi 5 hari ke depan. Tahap ini sama seperti tahap testing di mana setiap datanya akan menggunakan bobot terakhir dari tahap training dan hanya sampai pada tahap feed forward saja. Pada tahap ini hanya akan memakai 1 pola data yaitu pola terakhir dari data testing dan akan digunakan untuk memprediksi 5 hari kedepan, logikanya yaitu untuk hasil prediksi dari pola ke $n + 1$ akan disimpan dan digunakan untuk memprediksi nilai di pola ke $n + 2$, begitu seterusnya sampai didapatkan prediksi untuk 5 hari kedepan.

Data prediksi untuk 5 hari berikutnya didapatkan pada tabel berikut ini:

No	**Input**										**Y**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Target**
10	0.148	0.185	0.333	0.444	0.407	0.370	0.444	0.666	1.000	0.814	0.963
11	0.185	0.333	0.444	0.407	0.370	0.444	0.666	1.000	0.814	0.963	**0.704**
12	0.333	0.444	0.407	0.370	0.444	0.666	1.000	0.814	0.963	**0.704**	**0.659**
13	0.444	0.407	0.370	0.444	0.666	1.000	0.814	0.963	**0.704**	**0.659**	**0.609**
14	0.407	0.370	0.444	0.666	1.000	0.814	0.963	**0.704**	**0.659**	**0.609**	**0.558**
15	0.370	0.444	0.666	1.000	0.814	0.963	**0.704**	**0.659**	**0.609**	**0.558**	**0.532**

11. Langkah 11 : Denormalisasi Data

Melakukan tahap denormalisasi data dengan data sebagai berikut:

Data	**Normalisasi**	**Denormalisasi**
21	0.7050	3690
22	0.6594	3678
23	0.6100	3665
24	0.5588	3651
25	0.5324	3644

Dihitung menggunakan rumus berikut ini:

$$x = x' \times (x_{max} - x_{min}) + x_{min}$$

Dari tahap normalisasi didapatkan nilai maksimal sebesar 3770, nilai minimal sebesar 3500, dan nilai maksimal - minimal sebesar 270, maka perhitungan sebagai berikut:

$$x_{21} = 0.7050 \times (3779 - 3500) + 3500 = 3690$$
$$x_{22} = 0.6594 \times (3779 - 3500) + 3500 = 3678$$
$$x_{23} = 0.6100 \times (3779 - 3500) + 3500 = 3665$$
$$x_{24} = 0.5588 \times (3779 - 3500) + 3500 = 3651$$
$$x_{25} = 0.5324 \times (3779 - 3500) + 3500 = 3644$$

4.2 Implementasi Program

Logika yang dipakai atau diimplementasikan ke program python yaitu bahwa algoritma backpropragation ini memiliki 4 tahap utama yaitu preprocessing data di mana data yang dimiliki akan di normalisasi dulu sebelum masuk ke perhitungan selanjutnya ada inisialisasi berbagai parameter yang dibutuhkan. Tahap kedua yaitu tahap training data, tahap training ini bertujuan untuk mencari bobot yang optimal untuk digunakan pada tahap selanjutnya, di mana proses pencarian bobot ini akan dilakukan looping sebanyak max epoch yang sudah ditentukan ataupun akan berhenti jika MSE-nya sudah kurang dari error yang ditentukan pada tahap sebelumnya. Tahap ketiga adalah tahap testing data di mana setelah ditemukan bobot yang optimal dari tahap training selanjutnya akan digunakan pada tahap testing dan akan dicek MSE-nya. Dan tahap terakhir adalah tahap prediksi, pada tahap ini hamper sama seperti tahap testing tetapi memiliki perbedaan pada data yang digunakan yaitu pada tahap prediksi hanya menggunakan 1 pola saja untuk memprediksi beberapa data kedepan, setiap hasil dari prediksi akan disimpan dan digunakan lagi untuk memprediksi data selanjutnya.

Setelah menjalankan program yang sudah dibuat, akan didapatkan hasil yaitu Prediksi saham untuk 5 hari ke depan antara lain 3913, 3902, 3887, 3886 dan 3885, dengan MSE pada tahap training dan testing sebesar 0.0099971 dan 0.0593303.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan Metode prediksi yang dilakukan untuk memprakirakan Harga Saham PT. Telekomunikasi Indonesia, Tbk menggunakan Algoritma Backpropagation diperoleh beberapa kesimpulan diantaranya sebagai berikut :

1. Didapatkan hasil akhir prediksi harga saham 5 hari berikutnya dari perhitungan manual menggunakan Algoritma Backpropagation yaitu, 3913, 3902, 3887, 3886 dan 3885. didapatkan pula nilai MSE dari tahap training dan testing berturut-turut sebesar 0.0099971 dan 0.0593303.
2. Permasalahan prediksi harga saham PT. Telekomunikasi, Tbk dapat diselesaikan menggunakan bahasa pemrograman python dengan hasil akhir prediksi yang serupa.

5.2 Saran

Beberapa saran yang perlu dipertimbangkan adalah:

1. Program yang telah dibuat belum bisa untuk menggunakan big data atau data yang besar untuk tahap training dan testing karena jika datanya terlalu besar maka hasil target yang didapatkan akan memiliki nilai yang sama untuk setiap polanya.
2. Lebih teliti dalam membuat pola data dengan menggunakan perulangan data, dan lebih teliti dalam mengimplemetasikan logika berpikir ke dalam programnya.

DAFTAR PUSTAKA

A. Hermawan, Pengantar Jaringan Syaraf Tiruan, Teori, dan Aplikasi, Yogyakarta: Andi, 2006.

B. B. W. Putra, M. A. Albar dan B. Irmawati, "Implementasi Jaringan Syaraf Tiruan," JTIKA, pp. 233-244, 2019.

D. Herlianto, Manajemen Investasi Plus Jurus Mendeteksi Investasi Bodong, Yogyakarta: Pustaka Baru, 2013.

D. Puspitaningrum, Pengantar Jaringan Syaraf Tiruan, Yogyakarta: Andi Offset, 2006.

D. R. Artha, N. A. Achsani dan H. Sasongko, "Analisis Fundamental, Teknikal Dan Makroekonomi," MK, pp. 175-184, 2014.

E. Pardiansyah, "Investasi dalam Perspektif Ekonomi Islam: Pendekatan Teoritis dan Empiris," Economica: Jurnal Ekonomi Islam , pp. 337-373, 2017.

G. Z. Muflih, Sunardi dan A. Yudhana, "Jaringan Saraf Tiruan Backpropagation Untuk Prediksi Curah Hujan Di Wilayah Kabupaten Wonosobo," MUST: Journal of Mathematics Education, Science and Technology, pp. 45-56, 2019.

J. T. Hardinata, H. Okprana, A. P. Windarto dan W. Saputra, "Analisis Laju Pembelajaran dalam Mengklasifikasi Data Wine Menggunakan Algoritma Backpropagation," Jurnal Sains Komputer & Informatika (J-SAKTI), pp. 422-432, 2019.

M. S. Wibawa, "Pengaruh Fungsi Aktivasi, Optimisasi dan Jumlah Epoch Terhadap Performa Jaringan Saraf Tiruan," Jurnal Sistem Informatika, pp. 167-174, 2017.

N. P. Sakinah, I. Cholissodin dan A. W. Widodo, "Prediksi Jumlah Permintaan Koran Menggunakan Metode Jaringan Syaraf Tiruan Backpropagation," Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, pp. 2612-2618, 2018.

R. B. Afrianto, H. Tjandrasa dan I. Arieshanti, "Prediksi Pergerakan Harga Saham Menggunakan Metode Backpropagation Neural Network," SimanteC, pp. 132-141, 2013.

S. Makridaki, S. Wheelwright dan V. McGee, Metode dan Aplikasi Peramalan, Jakarta: Binarupa Aksara, 2003.

Suyanto, Artificial Intelligence, Bandung: Informatika, 2007.

LAMPIRAN

Script program python

```
import numpy as np

### TAHAP PREPROCESSING DATA DAN INISIALISASI ###

# Fungsi sigmoid dan turunannya
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
def sigmoid_derivative(x):
    return x * (1 - x)
# Fungsi Normalisasi data
def normalize(data):
    min_vals = np.min(data)
    max_vals = np.max(data)
    normalized_data = (data - min_vals) / (max_vals - min_vals)
    return normalized_data, min_vals, max_vals
# Fungsi Denormalisasi data
def denormalize(normalized_data, min_vals, max_vals):
    return normalized_data * (max_vals - min_vals) + min_vals
# Membaca data dari file
with open('data_training.txt', 'r') as file:
    lines = file.readlines()
data = []
output_data = []
for line in lines:
    values = line.strip().split()                    # Menggunakan spasi
sebagai pemisah
    data.append([float(val) for val in values[:-1]])  # Mengambil semua
kecuali yang terakhir sebagai input
    output_data.append([float(values[-1])])          # Mengambil yang
terakhir sebagai target
input_data = np.array(data)
output_data = np.array(output_data)
# Normalisasi data input dan output
normalized_input_data, input_min_vals, input_max_vals =
normalize(input_data)
normalized_output_data, output_min_vals, output_max_vals =
normalize(output_data)
# Inisialisasi bobot dan bias
input_neurons   = input_data.shape[1]
hidden_neurons  = 10
output_neurons  = 1
learning_rate   = 0.5
max_epochs      = 100000
error_threshold = 0.01
```

```
print("\n\nPREDIKSI SAHAM MENGGUNAKAN ALGORITMA BACKPROPAGATION\n\n")
print("INISIALISASI\n")
print("Node pada Input Layer  : 10")
print("Node pada Hidden Layer : 10")
print("Node pada Output Layer : 1")
print("Learning Rate (Alpha)  : 0.5")
print("Maksimum Epoch         : 100000")
print("Maksimum Error         : 0.01")
# Menampilkan hasil normalisasi
print("\n\nPREPROCESSING DATA")
print("\nContoh hasil Normalisasi data :")
print(normalized_input_data[:3])   # Contoh menampilkan 3 baris pertama
print("\nContoh hasil Normalisasi data (Target):")
print(normalized_output_data[:3])  # Contoh menampilkan 3 baris pertama


### TAHAP TRAINING DATA ###

print("\n\nTAHAP TRAINING DATA")
# Generate bobot random
np.random.seed(1)
# Bobot untuk lapisan tersembunyi dan lapisan keluaran
hidden_weights = np.random.uniform(size=(input_neurons, hidden_neurons))
output_weights = np.random.uniform(size=(hidden_neurons, output_neurons))
# Bias untuk lapisan tersembunyi dan lapisan keluaran
hidden_bias = np.random.uniform(size=(1, hidden_neurons))
output_bias = np.random.uniform(size=(1, output_neurons))
# Menghitung MSE
print("\nMSE dari setiap 100 epoch adalah:")
for epoch in range(max_epochs):
    # Feed Forward
    hidden_layer_input = np.dot(normalized_input_data, hidden_weights) +
hidden_bias
    hidden_layer_output = sigmoid(hidden_layer_input)
    output_layer_input = np.dot(hidden_layer_output, output_weights) +
output_bias
    output = sigmoid(output_layer_input)
    # Menghitung error dan Mean Squared Error (MSE)
    error = normalized_output_data - output
    mse = np.mean(np.square(error))
    # Menghentikan pelatihan jika MSE kurang dari threshold
    if mse < error_threshold:
        print(f"Mencapai MSE kurang dari {error_threshold} setelah {epoch}
epochs")
        break
    # Backpropagation
    d_output = error * sigmoid_derivative(output)
    error_hidden_layer = d_output.dot(output_weights.T)
```

```
    d_hidden_layer = error_hidden_layer *
sigmoid_derivative(hidden_layer_output)
    # Memperbarui bobot dan bias
    output_weights += hidden_layer_output.T.dot(d_output) * learning_rate
    output_bias += np.sum(d_output, axis=0, keepdims=True) * learning_rate
    hidden_weights += normalized_input_data.T.dot(d_hidden_layer) *
learning_rate
    hidden_bias += np.sum(d_hidden_layer, axis=0, keepdims=True) *
learning_rate
    # Menampilkan MSE
    if epoch % 100 == 0:
        print(f"Epoch {epoch} - MSE: {mse:.7f}")
# Jika mencapai batas max_epochs
if epoch == max_epochs - 1:
    print("Mencapai batas maksimum epoch tanpa mencapai MSE yang
diinginkan.")
# Feed Forward untuk prediksi
def predict(input):
    hidden_layer_input = np.dot(input, hidden_weights) + hidden_bias
    hidden_layer_output = sigmoid(hidden_layer_input)
    output_layer_input = np.dot(hidden_layer_output, output_weights) +
output_bias
    output = sigmoid(output_layer_input)
    return output
# Prediksi
predictions = predict(normalized_input_data)
# Denormalisasi hasil prediksi
denormalized_predictions = denormalize(predictions, output_min_vals,
output_max_vals)
# Menghitung MSE setelah denormalisasi
mse_denormalized = np.mean(np.square(denormalized_predictions -
output_data))
# Menampilkan MSE terakhir
print(f"\nMSE terakhir setelah Denormalisasi: {mse:.7f}")
# Menampilkan hasil prediksi setelah denormalisasi
print("\nHasil Prediksi Setelah Denormalisasi:")
for i in range(len(normalized_input_data)):
    predicted_value = denormalized_predictions[i][0]
    actual_value = output_data[i][0]
    error = abs(predicted_value - actual_value)
    print(f"Input: {input_data[i]} -> Prediksi: {predicted_value:.4f} |
Aktual: {actual_value} | Selisih: {error:.4f}")
# Menampilkan Output bobot dan bias terakhir
print("\nBobot Terakhir untuk Hidden Layer:")
print(hidden_weights)
print("\nBobot Bias Terakhir untuk Hidden Layer:")
print(hidden_bias)
print("\nBobot Terakhir untuk Output Layer:")
```

```
print(output_weights)
print("\nBobot Bias Terakhir untuk Output Layer:")
print(output_bias)


### TAHAP TESTING DATA ###

print("\n\nTAHAP TESTING DATA\n")
# Normalisasi data
def normalize2(data2):
    min_vals2 = np.min(data2)
    max_vals2 = np.max(data2)
    normalized_data2 = (data2 - min_vals2) / (max_vals2 - min_vals2)
    return normalized_data2, min_vals2, max_vals2
# Denormalisasi data
def denormalize2(normalized_data2, min_vals2, max_vals2):
    return normalized_data2 * (max_vals2 - min_vals2) + min_vals2
# Membaca data dari file
with open('data_testing.txt', 'r') as file:
    lines = file.readlines()
data2 = []
output_data2 = []
for line in lines:
    values = line.strip().split()                       # Menggunakan spasi
sebagai pemisah
    data2.append([float(val) for val in values[:-1]])  # Mengambil semua
kecuali yang terakhir sebagai input
    output_data2.append([float(values[-1])])           # Mengambil yang
terakhir sebagai target
input_data2 = np.array(data2)
output_data2 = np.array(output_data2)
# Normalisasi data input dan output
normalized_input_data2, input_min_vals2, input_max_vals2 =
normalize2(input_data2)
normalized_output_data2, output_min_vals2, output_max_vals2 =
normalize2(output_data2)
# Inisialisasi bobot dan bias
input_neurons2   = input_data2.shape[1]
# Bobot untuk lapisan tersembunyi dan lapisan keluaran
hidden_weights2 = hidden_weights
hidden_bias2 = hidden_bias
# Bias untuk lapisan tersembunyi dan lapisan keluaran
output_weights2 = output_weights
output_bias2 = output_bias
# Feed Forward
hidden_layer_input2 = np.dot(normalized_input_data2, hidden_weights2) +
hidden_bias2
hidden_layer_output2 = sigmoid(hidden_layer_input2)
```

```
output_layer_input2 = np.dot(hidden_layer_output2, output_weights2) +
output_bias2
output2 = sigmoid(output_layer_input2)
# Menghitung error dan Mean Squared Error (MSE)
error2 = normalized_output_data2 - output2
mse2 = np.mean(np.square(error2))
# Backpropagation
d_output2 = error2 * sigmoid_derivative(output2)
error_hidden_layer2 = d_output2.dot(output_weights2.T)
d_hidden_layer2 = error_hidden_layer2 *
sigmoid_derivative(hidden_layer_output2)
# Feed Forward untuk prediksi
def predict2(input):
    hidden_layer_input2 = np.dot(input, hidden_weights2) + hidden_bias2
    hidden_layer_output2 = sigmoid(hidden_layer_input2)
    output_layer_input2 = np.dot(hidden_layer_output2, output_weights2) +
output_bias2
    output2 = sigmoid(output_layer_input2)
    return output2
# Prediksi
predictions2 = predict2(normalized_input_data2)
# Denormalisasi hasil prediksi
denormalized_predictions2 = denormalize2(predictions2, output_min_vals2,
output_max_vals2)
# Menghitung MSE setelah denormalisasi
mse_denormalized2 = np.mean(np.square(denormalized_predictions2 -
output_data2))
# Menampilkan hasil prediksi setelah denormalisasi
print("Hasil Prediksi Setelah Denormalisasi:")
for i in range(len(normalized_input_data2)):
    predicted_value2 = denormalized_predictions2[i][0]
    actual_value2 = output_data2[i][0]
    error2 = abs(predicted_value2 - actual_value2)
    print(f"Input: {input_data2[i]} -> Prediksi: {predicted_value2:.4f} |
Aktual: {actual_value2} | Selisih: {error2:.4f}")
# Menampilkan MSE
print(f"\nMSE: {mse2:.7f}")


### TAHAP PREDIKSI ###

# Menggunakan bobot terakhir dari training
hidden_weights3 = hidden_weights
hidden_bias3 = hidden_bias
output_weights3 = output_weights
output_bias3 = output_bias

new_data1 = np.loadtxt('data_prediksi.txt')
def normalize_new_data1(new_data1):
```

```
    min_vals1 = np.min(new_data1)
    max_vals1 = np.max(new_data1)
    normalized_new_data1 = (new_data1 - min_vals1) / (max_vals1 - min_vals1)
    return normalized_new_data1, min_vals1, max_vals1
def denormalize_new_data1(value, min_vals1, max_vals1):
    return value * (max_vals1 - min_vals1) + min_vals1
normalized_new_data1, min_vals1, max_vals1 = normalize_new_data1(new_data1)
hidden_layer_input1 = np.dot(normalized_new_data1, hidden_weights3) +
hidden_bias3
hidden_layer_output1 = sigmoid(hidden_layer_input1)
output_layer_input1 = np.dot(hidden_layer_output1, output_weights3) +
output_bias3
output1 = sigmoid(output_layer_input1)
denormalized_output1 = denormalize_new_data1(output1, min_vals1, max_vals1)
print("\n\nPREDIKSI DATA SELANJUTNYA\n")
print(new_data1)
print(f"Output prediksi nilai saham 1 hari selanjutnya yang telah
didenormalisasi : {denormalized_output1}\n")

# Looping untuk prediksi berkelanjutan
for i in range(4):
    new_data2 = new_data1[1:10]
    new_data2 = np.append(new_data2, denormalized_output1)
    normalized_new_data2, min_vals2, max_vals2 =
normalize_new_data1(new_data2)
    hidden_layer_input2 = np.dot(normalized_new_data2, hidden_weights3) +
hidden_bias3
    hidden_layer_output2 = sigmoid(hidden_layer_input2)
    output_layer_input2 = np.dot(hidden_layer_output2, output_weights3) +
output_bias3
    output2 = sigmoid(output_layer_input2)
    denormalized_output2 = denormalize_new_data1(output2, min_vals2,
max_vals2)
    new_data1 = np.roll(new_data1, -1)
    new_data1[-1] = denormalized_output2
    denormalized_output1 = denormalized_output2
    print(new_data2)
    print(f"Output prediksi nilai saham {i+2} hari selanjutnya yang telah
didenormalisasi : {denormalized_output2}\n")
```