PREDIKSI CURAH HUJAN MENGGUNAKAN METODE LONG SHORT TERM MEMORY (STUDI KASUS : KOTA BANDUNG)

Disusun Oleh:

Nama : Riza Farikhul Firdaus

NIM : 18523129

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2022

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PREDIKSI CURAH HUJAN MENGGUNAKAN METODE LONG SHORT TERM MEMORY (STUDI KASUS : KOTA BANDUNG)

TUGAS AKHIR

Disusun Oleh:

N a m a : Riza Farikhul Firdaus

NIM : 18523129

Yogyakarta, 17 Oktober 2022

Pembimbing,

17/10/2022

(Irving Vitra Paputungan, S.T., M.Sc., Ph.D.)

HALAMAN PENGESAHAN DOSEN PENGUJI

PREDIKSI CURAH HUJAN MENGGUNAKAN METODELONG SHORT TERM MEMORY (STUDI KASUS: KOTA BANDUNG)

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, Januari 2023

Tim Penguji

Irving Vitra Paputungan, S.T., M.Sc., Ph.D. _____

Anggota 1

Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. _____

Anggota 2

Sheila Nurul Huda, S.Kom., M.Cs. _____

Mengetahui,

Ketua Program Studi Informatika –

Program Sarjana Fakultas Teknologi

Industri Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Riza Farikhul Firdaus

NIM : 18523129

Tugas akhir dengan judul:

PREDIKSI CURAH HUJAN MENGGUNAKAN METODE LONG SHORT TERM MEMORY (STUDI KASUS: KOTA BANDUNG)

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Indramayu, 14 Desember 2022

(Riza Farikhul Firdaus)

HALAMAN PERSEMBAHAN

Assalamualaikum warahmatullahi wabarakatuh alhamdullilahirobbil'alamin, puji syukur kepada Allah SWT berkat ridho dan karunia-Nya yang telah memberikan kelancaran dan keberkahan selama proses pengerjaan tugas akhir ini. Semoga ilmu yang selama ini saya pelajari mendapatkan keberkahan dan akan berguna untuk orang-orang sekitar saya, Allahuma Amiin.

Terima kasih kepada Allah SWT, kedua orang tua saya dan kakak saya yang telah banyak memberikan seluruh tenaganya sampai saya bisa diposisi seperti ini, semua proses dan perjalanan yang saya lalui tak akan lepas dari kerja keras kedua orang tua saya.

Terima kasih untuk Bapak Irving Vitra Paputungan, S.T., M.Sc., Ph.D., selaku pembimbing, dan para dosen Informatika yang selalu mengajarkan ilmu baru yang tidak lupa menambahkan tentang ilmu keislaman di dalamnya yang sangat berharga bagi saya.

Untuk Shafira Ramadhina Putri saya ucapkan terima kasih karena selalu menemani pada saat proses penyusunan skripsi dari awal hingga akhir, selalu memberikan semangat dan selalu ada setahun belakangan ini. Terima kasih karena sudah menjadi *support system*.

Semoga Allah mengganti kebaikan-kebaikan yang kalian berikan dengan kebaikan yang lebih baik lagi. Semoga diberikan kekuatan, kelancaran, kesehatan dan kebahagiaan dunia akhirat untuk kita semua. Aamiin.

HALAMAN MOTO

“Karena sesungguhnya sesudah kesulitan itu pasti ada kemudahan.” - Q.S Al-Insyirah:5

“Dan barang siapa yang bertakwa kepada Allah, niscaya Allah menjadikan baginya kemudahan dalam urusannya.” - Q.S At-Talaq:4)

“Satu-satunya sumber dari pengetahuan adalah pengalaman” - Albert Einstein

KATA PENGANTAR

Assalamualaikum Wr. Wb. Saya ucapkan Alhamdulillah sebagai bentuk rasa syukur dan terima kasih saya kepada Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya serta memberikan ridha dan kekuatan kepada penulis sehingga bisa menyelesaikan skripsi ini dengan baik.

Skripsi yang berjudul "Prediksi Curah Hujan Menggunakan Metode Long Short Term Memory" disusun untuk memenuhi persyaratan mendapatkan gelar Sarjana Komputer (S.Kom) pada Program Studi Informatika di Universitas Islam Indonesia. Penulis mengucapkan terima kasih kepada pihak-pihak yang terlibat dalam memberikan dukungan kepada penulis. Pihak-pihak tersebut yaitu:

1. Kedua orang tua penulis, Ibu dan Bapak yang telah memberikan semangat moral untuk penulis dan bekerja keras mendidik agar menjadi manusia yang pantang menyerah, bertanggung jawab dan bermanfaat bagi orang banyak.
2. Luthfi Himawan sebagai kakak penulis yang telah memberikan semangat dan bantuan kepada penulis berupa konsumsi di saat penyusunan skripsi.
3. Bapak Irving Vitra Paputungan, S.T., M.Sc., Ph.D. selaku Dosen Pembimbing yang telah mengajarkan dan membimbing penulis sehingga penulis dapat menyelesaikan penelitian dengan baik.
4. Shafira Ramadhina Putri yang tidak pernah bosan untuk menyemangati penulis untuk menyusun skripsi dan selalu menghibur dengan cerita-cerita *random* nya.
5. Teman-teman yang tergabung dalam grup WhatsApp "Bubadibako" diantaranya Alfian, Jamor, Abid, dan Doddy yang selalu menghibur di saat penulis sedang pusing.
6. Teman-teman penulis yang terlibat dalam proses perkuliahan maupun di luar perkuliahan seperti Irfan, Dany, Abyan, Rizky Sukek, Arief, Haris, Faris, Farhan, Dhika yang selalu membantu membagikan informasi penting dan saling membantu satu sama lain. Semoga kita dapat menggapai impian kita masing-masing dan bertemu lagi di titik terbaik kita.

Demikian yang dapat penulis sampaikan. Skripsi ini dibuat dengan usaha sendiri, mesikpun masih terdapat banyak kesalahan di dalamnya. Mudah-mudahan bisa bermanfaat bagi yang membaca dan membutuhkannya.

Wassalamualaikum Wr. Wb.

Indramayu, 14 Desember 2022

Riza Farikhul Firdaus

SARI

Salah satu faktor yang sangat penting untuk menunjang kegiatan di berbagai bidang yang dapat dijadikan acuan adalah perkiraan curah hujan. Hal ini memungkinkan mereka untuk melakukan aktivitas tanpa terhambat oleh kondisi cuaca yang tidak mendukung. Penyebab peningkatan dan pengurangan curah hujan menjadi factor penting untuk ini. Memprediksi hujan karena itu penting bagi banyak kelompok yang berbeda, terutama mereka yang terlibat dalam kegiatan di luar ruangan. Metode Long Short Term Memory (LSTM) digunakan dalam penelitian ini untuk menganalisis data curah hujan dari Kota Bandung yang diunduh dari website Badan Meteorologi, Klimatologi dan Geofisika (BMKG). Parameter epoch dan batch siza dapat berdampak pada hasil algoritma LSTM. Nilai RMSE terbaik adalah Train Score 12.21 dan Test Score 8.78, sedangkan akurasi maksimum dicapai dengan epoch 75 dan batch size 1.

GLOSARIUM

Prediksi	proses memperkirakan sesuatu yang mungkin terjadi di masa depan
Batch size	jumlah sampel data yang diberikan pada jaringan
Deep learning	teknik kecerdasan buatan untuk memodelkan hubungan kompleks antara input dan output
LSTM	sistem penyimpanan data untuk memprediksi data yang disimpan dalam waktu lama

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Cuaca adalah kualitas udara yang diamati dalam waktu yang relatif singkat atau di daerah yang kecil. Cuaca menurut WCC (*World Climate* Conference adalah keadaan atmosfer yang diukur secara menyeluruh dengan memperhitungkan perubahan, perkembangan, dan datang atau lenyapnya suatu fenomena udara (Luthfiarta, Febriyanto, Lestiawan, & Wicaksono, 2020). Unsur-unsur pembentuk cuaca yaitu suhu, udara, tekanan udara, kelembaban udara, laju uap air, awan, curah hujan dan angin. Curah hujan itu sendiri merupakan salah satu komponen iklim yang penting dari ekosistem alam. Garis khatulistiwa yang terbentang di Indonesia menjadikan Indonesia sebagai negara yang beriklim tropis (Rachmawati, 2021).

Musim kemarau dan musim hujan adalah dua musim di lingkungan tropis. Intensitas iklim bervariasi menurut wilayah. Misalnya, pada musim hujan, intensitas curah hujan bervariasi tergantung pada garis lintang wilayah, ketinggian, kedekatan dengan sumber air, arah angin, suhu tanah, dan luas lahan. Karena adanya berbagai unsur yang dapat mempengaruhi besarnya curah hujan, maka cuaca di Indonesia tidak selalu berjalan normal atau sesuai dengan musim. Sebaliknya, itu sering berubah tiba-tiba setiap saat. Seperti di Kota Bandung yang intensitas curah hujannya tidak bisa ditebak, karena setiap hari dalam seminggu pasti melihat hujan dengan intensitas yang bervariasi dan bahkan sama sekali tidak terjadi hujan dalam sepekan tersebut (Soekendro, 2021). Akibatnya, kelompok yang berbeda dapat menggunakan prakiraan cuaca, yang dibutuhkan dan sangat berharga oleh berbagai pihak, sebagai panduan untuk menjalankan operasi sehari-hari mereka (Rizki, Basuki, & Azhar, 2020).

Setiap provinsi di Indonesia kini memiliki akses informasi cuaca berkat BMKG (Badan Meteorologi, Klimatologi, dan Geofisika). Memanfaatkan berbagai sumber data dan model analisis cuaca yang dibuat oleh BMKG Pusat sendiri, BMKG Kota Bandung telah menghasilkan prakiraan cuaca untuk digunakan sendiri. Namun, para ahli BMKG sendiri masih menilai akurasi prediksi tersebut masih rendah (Insani & Fadilah, 2020). Untuk menemukan pendekatan yang paling cocok yang dapat memberikan prediksi cuaca terutama prediksi curah hujan dengan tingkat akurasi yang tinggi, metode yang tepat untuk memprediksinya masih dipilih (Insani & Fadilah, 2020).

Terdapat penelitian tentang prediksi di berbagai bidang akhir-akhir ini yang telah dilakukan

menggunakan *deep learning* mampu menghasilkan akurasi di atas 85% (Budi et al., 2021). *Deep learning* membutuhkan lebih sedikit keterlibatan manusia daripada teknik *machine learning* konvensional, tetapi masih membutuhkan data yang cukup besar. Pendekatan lain yang dilakukan yaitu mengubah kata menjadi vektor yang berisikan angka dengan *word embedding*. Kemudian algoritma dilatih dengan nilai dari suatu kata atau kalimat. Metode *deep learning* mengekstraksi dari *neural network* dan kemudian belajar dari *error*. Fungsi aktivasi memetakan beberapa lapisan yang membentuk *neural networks* atau jaringan saraf. CNN, RNN, LSTM, dan *Gated Recurrent Unit* (GRU) adalah model *deep learning* yang umum untuk prediksi *time series* (Zhang et al., 2018). *Long Short Term Memory* (LSTM) merupakan penyempurnaan dari *Reccurent Neural Network* (RNN) karena kemampuannya untuk mengingat ukuran data yang sulit dicapai dengan teknik fitur tradisional (Wiranda & Sadikin, 2019).

Long Short Term Memory (LSTM) mempunyai akurasi yang baik. Oleh sebab itu, menggunakan teknik ini untuk menilai apakah prediksi curah hujan metode Long Short Term Memory (LSTM) lebih akurat daripada algoritma prediksi yang digunakan sebelumnya dalam penelitian. Penelitian ini dilakukan menggunakan data curah hujan Kota Bandung yang diambil dari website BMKG dengan menjadikan RMSE sebagai evaluasi dari prediksi yang dihasilkan.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah di atas, rumusan masalah yang akan dibahas pada penelitian ini adalah:

a. Mengukur performa *Long Short Term Memory* dalam memprediksi curah hujan di Kota Bandung.

1.3 Batasan Masalah

Batasan masalah yang ditentukan berdasarkan rumusan masalah penelitian ini yaitu:

a. Data yang digunakan didapatkan dari website BMKG (Badan Meteorologi, Klimatologi dan Geofisika) dan ukuran curah hujan menggunakan milimeter.
b. Metode yang digunakan yaitu *Long Short Term Memory* (LSTM) dan RMSE untuk mengevaluasi hasil.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini yaitu menerapkan *Long Short Term Memory* untuk mengukur dan melakukan prediksi cuaca di Kota Bandung selama kurun waktu tertentu dengan menjadikan RMSE sebagai evaluasi.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah sebagai berikut:

a. Memberikan pengetahuan terkait metode *Long Short Term Memory* dalam melakukan prediksi curah hujan.
b. Dapat menjadi referensi bagi peneliti selanjutnya tentang metode *Long Short Term Memory*.

1.6 Metodologi Secara Umum

a. Proses Pendahuluan : Berisi latar belakang atau alasan penelitian Prediksi Curah Hujan Menggunakan Metode *Long Short Term Memory* (Studi Kasus : Kota Bandung) dilaksanakan.
b. Proses Studi Literatur : Untuk mencari literatur berupa jurnal maupun informasi mengenai data di website BMKG atau dasar-dasar untuk penelitian.
c. Proses Eksplorasi Data : Mengambil informasi mengenai data yang diperlukan seperti pola curah hujan di website BMKG dan mencari metode prediksi yang tepat.
d. Proses Pengambilan Data yang didapat dari website BMKG yang telah dilakukan saat eksplorasi data.
e. Proses Pengujian : Dilakukan dengan metode yang telah ditentukan.
f. Visualisasi hasil pengujian: Data yang diperoleh dari pengujian diolah dan dimasukkan ke dalam penulisan skripsi.
g. Proses menganalisa hasil penelitian dengan cara mengamati hasil dengan *detail* dan menyusun komponen-komponen untuk dikaji lebih lanjut.

1.7 Sistematika Penulisan

Sistematika penulisan penelitian ini adalah sebagai berikut:

a. BAB 1 Pendahuluan

Pada bab ini membahas membahas tentang latar belakang penelitian dan juga menjelaskan rumusan, batasan, tujuan, manfaat penelitian.

b. BAB 2 Tinjauan Pustaka dan Teori

Landasan teori yang terdapat di dalam penelitian ini.

c. BAB 3 Metodologi Penelitian

Pada bab ini menjelaskan langkah-langkah yang digunakan untuk mengerjakan penelitian yang dilakukan.

d. BAB 4 Hasil dan Pembahasan

Bab ini berisikan hasil dari penelitian yang telah dilakukan menggunakan metode yang telah ditentukan, yaitu *long short term memory*.

e. BAB 5 Kesimpulan

Bab ini berisikan kesimpulan mengenai penelitian yang telah selesai dilakukan serta saran dari penulis.

BAB II
TINJAUAN PUSTAKA DAN TEORI

2.1 Prediksi

Untuk mengurangi kesalahan (perbedaan antara apa yang sebenarnya terjadi dan hasil yang diproyeksikan), prediksi adalah proses mengevaluasi secara metodis apa yang paling mungkin terjadi di masa depan berdasarkan peristiwa masa lalu dan masa kini yang diketahui. Tujuan prediksi adalah untuk menghasilkan tanggapan yang sedekat mungkin dengan peristiwa yang akan terjadi, daripada memberikan jawaban yang pasti (Irfan et al., 2018).

Prediksi juga merupakan hasil dari memproyeksikan nilai ke masa depan berdasarkan informasi dari masa lalu. Prediksi memberikan informasi untuk perencanaan dan proses pengambilan keputusan dengan memprediksi apa yang akan terjadi dalam situasi tertentu (Mukhlisin et al., 2020).

2.2 Curah Hujan

Jumlah air yang jatuh ke tanah selama periode waktu tertentu dikenal sebagai curah hujan, dan diukur dalam milimeter ketinggian di atas permukaan horizontal. Pengertian lain dari curah hujan adalah kenaikan air hujan yang terakumulasi pada permukaan datar tanpa menguap, merembes, atau mengalir. Karena ketinggian negara yang bervariasi, Indonesia mengalami curah hujan yang beragam. Satu milimeter hujan setara dengan satu liter air, dan satu milimeter hujan berarti satu meter persegi tanah datar dapat menampung satu milimeter air (Triatmodjo, 2008).

Keuntungan dari kondisi curah hujan sendiri yaitu udara menjadi lebih sejuk, tanah-tanah menjadi subur, tidak kekurangan air/kekeringan. Sedangkan untuk kekurangannya yaitu berpotensi terjadinya banjir, masyarakat kesulitan untuk beraktivitas (Noerhayati et al., 2017). Mengenali pola curah hujan sangatlah penting untuk masyarakat yang sering beraktivitas di luar ruangan karena bisa memberikan keuntungan bagi masyarakat sehingga bisa menjalankan aktivitas di luar ruangan tanpa terkendala cuaca/hujan. Tetapi apabila masyarakat tidak bisa mengenali pola curah hujan maka akan memberikan kerugian seperti terkendala cuaca/hujan saat beraktivitas di luar ruangan (Fitri & Taufiq, 2020).

2.3 Deret Waktu (*Time Series*)

Pengaturan nilai dan variabel berdasarkan waktu dikenal sebagai deret waktu (*time series*). Pola pergerakan nilai variabel dalam interval waktu seperti mingguan, bulanan, dan tahunan dianalisis dan dipelajari dengan menggunakan time series. Pendekatan deret waktu didasarkan pada gagasan bahwa pola sebelumnya akan berulang. Pengukuran yang dapat digunakan untuk membuat keputusan saat ini, termasuk memprediksi dan mempersiapkan masa depan, dapat digunakan untuk menuai manfaat dari analisis deret waktu (Jayanti, 2013).

2.3.1 Jenis-Jenis Deret Waktu

Berdasarkan berapa banyak variabel yang peneliti gunakan sebagai pengamatan, analisis data deret waktu dapat dibagi ke dalam kategori yang berbeda. Deret waktu univariat terdiri dari data dari satu variabel pengamatan. Data nilai tukar dolar terhadap rupiah merupakan gambaran dari time series univariat. Pengamatan deret waktu juga dapat dilihat pada dua variabel, yang dikenal sebagai deret waktu bivariat, selain hanya satu. Analisis deret waktu bivariat adalah jenis analisis deret waktu yang berfokus pada satu variabel tetapi akan menghasilkan temuan yang lebih baik jika juga mempertimbangkan variabel tambahan yang dapat digunakan untuk menjelaskan keragaman variabel yang diteliti. Misalnya, diyakini bahwa berat badan bayi berdampak pada berapa lama bayi baru lahir diukur sepanjang waktu. untuk mencatat panjang dan berat bayi pada setiap pengamatan. Model fungsi transfer dapat digunakan untuk mempelajari situasi seperti ini, yang dikategorikan sebagai analisis deret waktu bivariat. Selain itu, ketika mengevaluasi variabel dalam pengamatan yang terkait dengan variabel lain yang dikelompokkan dalam sistem terkait, analisis deret waktu dengan multivariat digunakan. Agar pengamatan suatu variabel dipengaruhi oleh variabel yang sedang diselidiki dan juga faktor-faktor lain yang terkait (Wizsa, 2018).

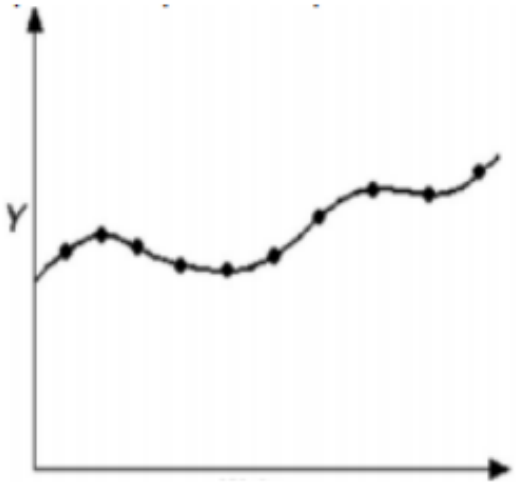
2.3.2 Jenis Data Deret Waktu Berdasarkan Plot Data

Data yang terkandung dalam tipe deret waktu dapat dipetakan berdasarkan waktu. Pola data harus diamati untuk selanjutnya menentukan langkah-langkah analisis yang dilakukan. Menurut polanya, data deret waktu dibagi menjadi empat bagian. Data terutama terdiri dari satu atau lebih komponen utama yaitu (Hansun, 2013) :

a. Siklus (*Cycles*/C)

   Siklus adalah rangkaian fluktuasi atau siklus gelombang yang berlangsung lama dan tidak merata. Jika gerakan kembali setelah istirahat lebih dari satu tahun, itu terlihat sebagai

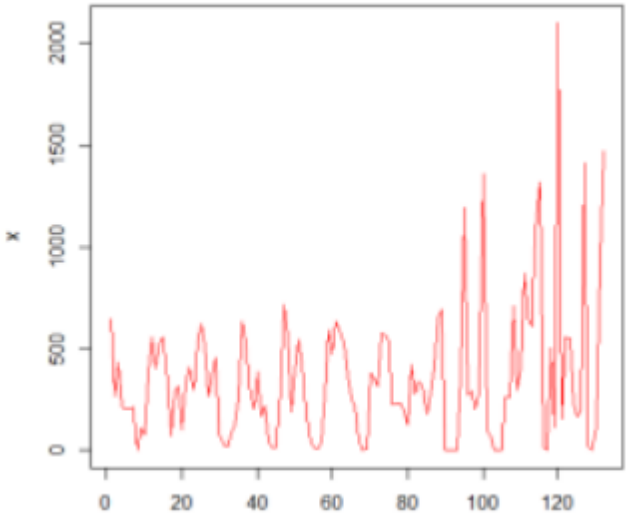
siklus. Data curah hujan dari database internet BMKG merupakan gambaran data dengan komponen siklus.



Gambar 2.1 Pola Siklus
Sumber : (Andini & Auristandi, 2016)

b. *Irregular* (I)

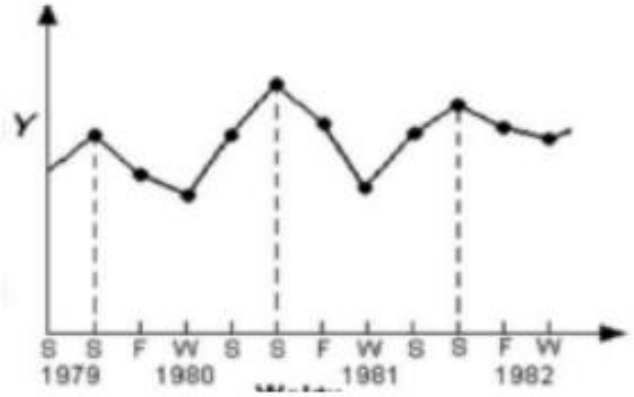
Irregular merupakan gerakan fluktuasi yang terjadi akibat kejadian yang tak terprediksi atau kejadian non-periodik contohnya seperti kejadian bencana alam yang tidak bisa diprediksi kejadiannya.



Gambar 2.2 Pola *Irregular*
Sumber : (Darmawan et al., 2016)

c. Musiman (*Seasonality*/S)

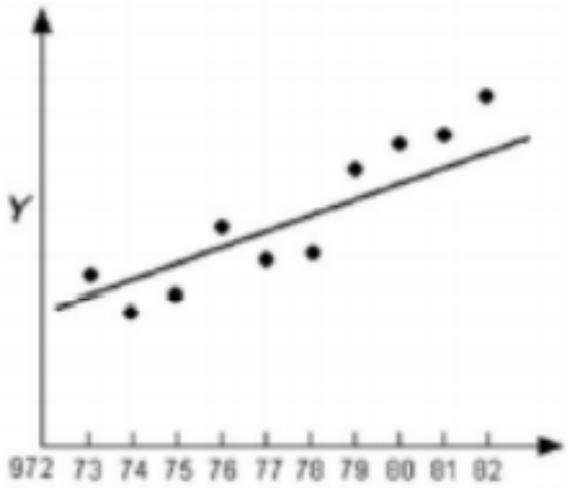
Jenis data runtun waktu ini merupakan pola fluktuasi permintaan di atas atau di bawah garis tren yang terjadi setiap tahun. Apa yang dimaksud dengan fluktuasi musiman adalah bahwa mereka dapat dipecah setiap triwulan, bulanan, mingguan atau harian, menunjukkan pola yang berubah secara teratur dari waktu ke waktu. Misalnya, pola jumlah orang yang menggunakan angkutan umum padat pada waktu-waktu tertentu, seperti pada jam-jam sibuk atau musim liburan.



Gambar 2.3 Pola Musiman
Sumber : (Andini & Auristandi, 2016)

d. *Trend* (T)

Dalam data deret waktu, tren adalah komponen jangka panjang yang menampilkan kenaikan atau penurunan selama periode waktu yang telah ditentukan. Tren adalah, secara sederhana, kurva yang menggambarkan kecenderungan keseluruhan dari serangkaian data waktu. Contohnya termasuk perubahan populasi global dan inflasi.

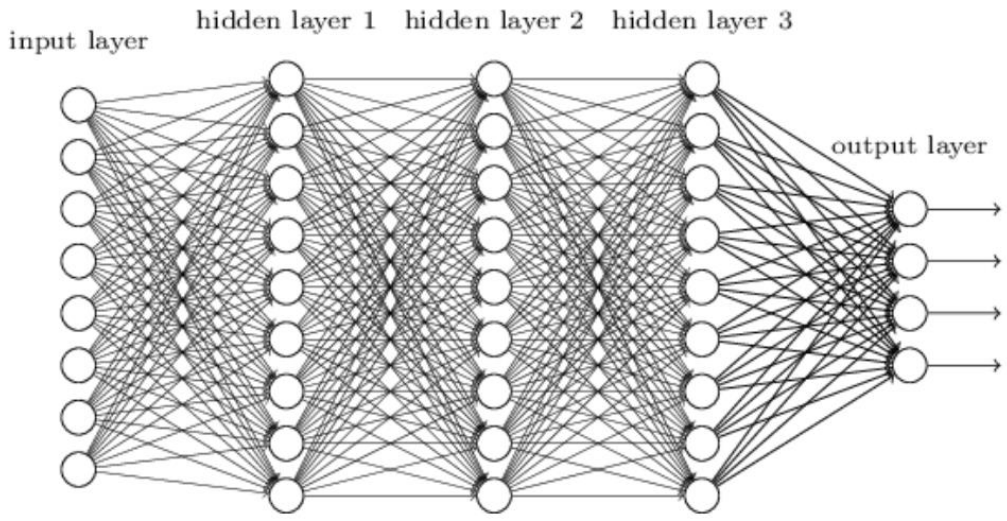


Gambar 2.4 Pola Trend
Sumber : (Darmawan et al., 2016)

2.4 Deep Learning

Deep learning merupakan teknik kecerdasan buatan untuk memodelkan hubungan kompleks antara input dan output. Selain itu *deep learning* memiliki atribut khas dibandingkan dengan kontruksi model dan pelatihan model lainnya (Yunanto, Purfini, & Prabuwisesa, 2020). *Deep learning* mengintegrasikan pembelajaran fitur dan konstruksi model dalam satu model. Representasi fitur abstrak ini kemudian dimasukkan ke lapisan klasifikas untuk melakukan tugas klasifikasi atau regresi (Yunanto et al., 2021).

Deep learning adalah algoritma pembelajaran mesin yang didasarkan pada pembelajaran tanpa pengawasan (*unsupervised learning*) dan terdiri dari beberapa tingkat fungsional dan ekstraksi data. Untuk membentuk representasi hierariki harus mendapatkan fitur yang lebih tinggi dari yang lebih rendah. Pada Gambar 2.5 menunjukkan gambaran dari *deep* learning yang mana terdapat 4 layer, dan pada setiap layer mempunyai jumlah node yang berbeda-beda



Gambar 2.5 Ilustrasi *Deep Learning*

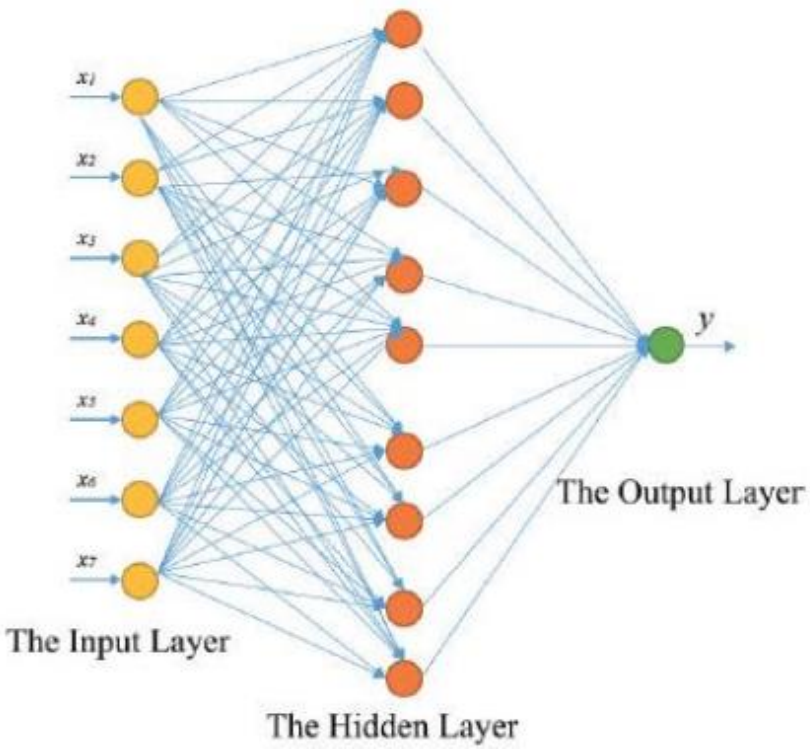
2.5 Jaringan Saraf Tiruan (*Artificial Neural Network*)

Jaringan Syaraf Tiruan (JST) juga dikenal sebagai *Artificial Neural Network* (ANN) dalam bahasa Inggris adalah model kecerdasan yang meniru struktur otak manusia. Kemudian diimplementasikan dengan menggunakan perangkat lunak komputer yang dapat melakukan proses perhitungan saat pembelajar memperoleh pengetahuan baru. Jaringan Syaraf Tiruan (JST) adalah model berpikir yang terinspirasi dari otak manusia. Jaringan saraf tiruan (JST) terdiri dari banyak prosesor yang saling berhubungan dan sangat dasar yang dikenal sebagai neuron (Kurniawansyah, 2018). Fitur neuron sangat mirip dengan Jaringan Saraf Tiruan (JST). Ini dibagi menjadi kelompok-kelompok yang dikenal sebagai *layer*. Neuron dalam satu *layer* akan terhubung dengan yang ada di *layer* di bawahnya. Bobot asosiasi atau kekuatan hubungan antar neuron yang berdekatan mencerminkan kekuatan hubungan tersebut (Dharma et al., 2011).

Penentuan bobot matriks melalui pelatihan (*training*) merupakan langkah penting dalam pembangunan Jaringan Syaraf Tiruan (JST). Dua teknik pelatihan yang berbeda, pelatihan yang diawasi (*supervised training*) dan pelatihan tanpa pengawasan (*unsupervised training*), digunakan dalam jaringan saraf tiruan (JST). Pengawasan eksternal diperlukan dalam mekanisme *supervised training* untuk mengarahkan proses *training*. Algoritma ini menggunakan data *input-output* sebagai contoh, data contoh harus menggunakan data yang telah diverifikasi secara independen. Untuk menentukan perbedaan antara *output* yang diperkirakan dan *output* yang sebenarnya, *output* jaringan akan dibandingkan dengan data *output* yang diharapkan (*output* sampel). Selain itu, bobot

jaringan dimodifikasi untuk menciptakan hasil yang sama atau cukup dekat dengan hasil yang diinginkan menggunakan perbedaan ini. Algoritma *back-propagation training*, yang digunakan dalam aplikasi *engineering* adalah nama lain untuk mekanisme *supervised training*. Meskipun Jaringan Syaraf Tiruan (JST) adalah model *blackbox* dan tidak mempertimbangkan fisik dari permasalahan, model Jaringan Syaraf Tiruan (JST) yang terlatih dapat mendeteksi proses fisik (Dharma et al., 2011).

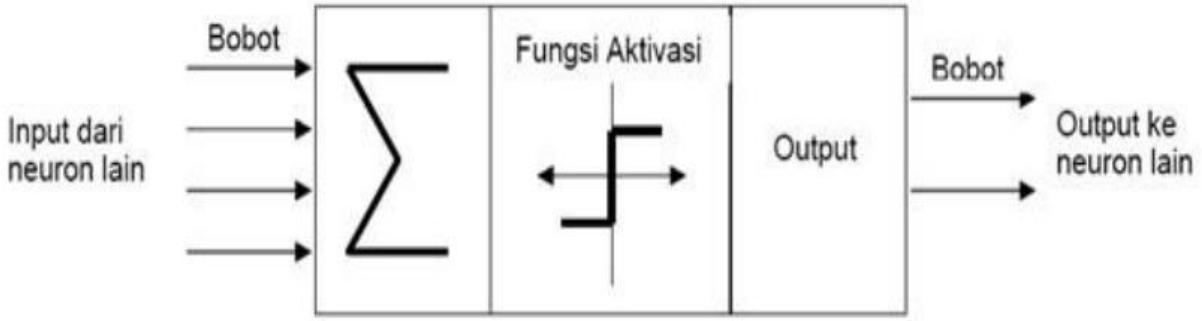
Input, *hidden*, dan *output layer* membentuk struktur jaringan saraf tiruan (JST). *Input layer* menerima informasi (α) menggunakan bobot kedatangan tertentu (w). Bobot kemudian ditambahkan ke *hidden layer* setelah itu. Temuan tersebut kemudian dijumlahkan dan dibandingkan dengan titik potong. *Output layer* tidak akan mendapatkan nilai hasil jika kurang dari ambang batas. Namun, jika memenuhi ambang batas, nilai hasil akan diteruskan (Habibi & Riksakomara, 2017). Gambar 2.6 menggambarkan arsitektur Jaringan Syaraf Tiruan (JST)



Gambar 2.6 Struktur Jaringan ANN
Sumber : (Prathama et al., 2018)

Neuron di *input layer* adalah mereka yang mengambil informasi dari dunia luar. Penjelasan tentang masalah ini adalah *input* dalam kasus ini. Selain itu, *hidden layer* terdiri dari neuron yang mengambil *input* dari *input layer* dan mengirimkan *output* ke *layer* berikutnya. Neuron yang

menerima *output* dari *hidden layer* dan mengirimkannya ke pengguna membentuk *output layer*, terkadang disebut juga sebagai modul *output* (Dharma et al., 2011).



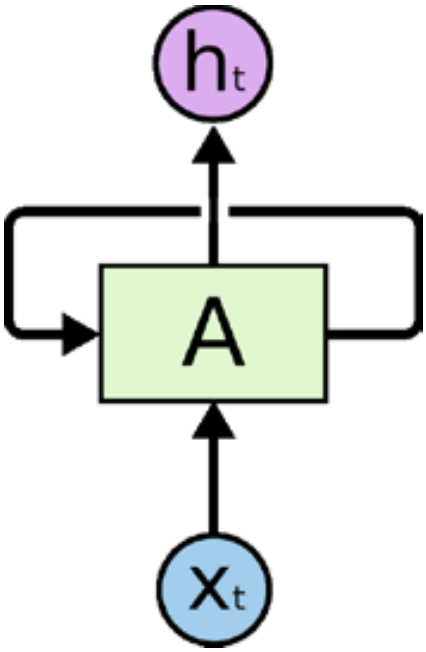
Gambar 2.7 Struktur Dasar ANN
Sumber : (Suhartono, 2012)

Gambar 2.7 menampilkan struktur Jaringan Syaraf Tiruan (JST) dasar di mana setelah masuk ke neuron, nilai *input* yang tersedia dijumlahkan dengan fungsi difusi tampak (pooling function) dengan sigma simbolik (∑) seperti terlihat pada gambar. Hasil penjumlahan tersebut diproses oleh fungsi aktivasi masing-masing neuron, dan hasil penjumlahan tersebut dibandingkan dengan suatu limit tertentu. Ketika nilainya melebihi ambang batas, neuron dinonaktifkan. Sebaliknya, di bawah ambang batas, neuron diaktifkan. Ketika diaktifkan, neuron mengirimkan nilai *output* ke semua neuron yang terhubung melalui bobot *output*. Proses tersebut berulang pada *input* berikutnya (Suhartono, 2012).

2.6 *Reccurent Neural Network*

Manusia tidak selalu memilih pilihan yang sama dalam kehidupan sehari-harinya. Saat mengambil keputusan, manusia akan mempertimbangkan masa lalu atau informasi yang ada. Pengembangan RNN (*Reccurent Neural Network*) dibangun atas dasar cara berpikir ini. RNN (*Recurent Neural Network*) tidak menggunakan pengetahuan dari masa lalu dalam proses pembelajaran, seperti analogi. Ini membedakan RNN (*Reccurent Neural Network*) dari ANN (*Artificial Neural Network*) konvensional (Lapian et al., 2018).

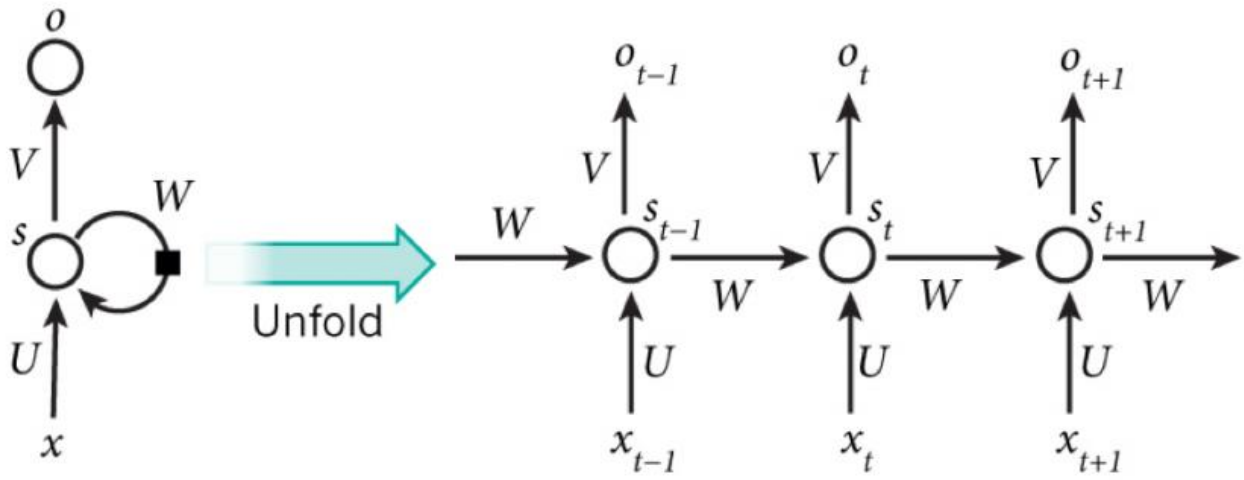
Jeff Elman menciptakan *Reccurent Neural Network* (RNN) untuk pertama kalinya pada tahun 1990. Varian *Artificial Neural Network* (ANN) yang dikenal sebagai RNN digunakan terutama untuk memproses data kontinu atau berurutan. RNN sering digunakan untuk mengatasi masalah dengan data deret waktu (*time series*).



Gambar 2.8 Struktur Umum *Reccurent Neural Network*
Sumber : (colah, 2015)

Apabila struktur pada gambar 2.8 dijabarkan, maka struktur RNN (Recurrent Neural Network) akan lebar di tengah dan cukup panjang untuk menyelidiki pola data yang bersangkutan. Karena hal tersebut RNN diperuntukkan khusus untuk menangani data yang berurutan. *Input* x_t menghasilkan *output* h_t.

Sinyal dalam model *Reccurent Neural Network* (RNN) mampu bergerak maju dan mundur berulang kali. *Layer* baru yang dikenal sebagai *context layer* harus ditambahkan agar ini terjadi. *Output* dari setiap *layer* juga dikirim ke *context layer*, di mana ia digunakan sebagai *input* untuk *time step* berikutnya, selain menyampaikan informasi lintas lapisan. *Context layer* adalah tempat RNN (*Reccurent Neural Network*) menyimpan informasi, hal tersebut yang membuat RNN (*Reccurent Neural Network*) dapat menghasilkan *output* atau urutan lainnya setelah mempelajari urutan data. RNN memiliki memori yang menyimpan hasil rekaman informasi yang dibuat sebelumnya, jika ditarik kesimpulan (Juanda et al., 2018).



Gambar 2.9 Proses *Reccurent Neural Network*
Sumber : (R.A.Y., 2018)

Gambar 2.9 merupakan proses membuka gulungan RNN (*Reccurent Neural Network*), maka kita cukup mengeluarkan semua jaringan bersama dengan urutan yang lengkap. Simbol-simbol yang sekarang digunakan dijelaskan sebagai berikut (Juanda et al., 2018) :

1. X_t merupakan *input* pada setiap *time step*.
2. S_t merupakan *hidden state* pada setiap *time step t*.
3. O_t merupakan *output* untuk setiap *step t*.

Pemetaan satu simpul S_t dan *output* O_t dapat ditulis sebagai (Tian et al., 2018) :

$$S_t = f((U \times X_t) + (W \times S_t - 1) \quad (2.1)$$

$$O_t = g(V \times S_t) \quad (2.2)$$

Dalam persamaan di atas, S_t dikatakan memori jaringan pada waktu t, U, W dan V adalah matriks bobot berbagi di setiap lapisan. X_t dan . O_t memiliki *input* dan *output* pada waktu t, dam f(.) dan g(.) mewaikili fungsi nonlinear.

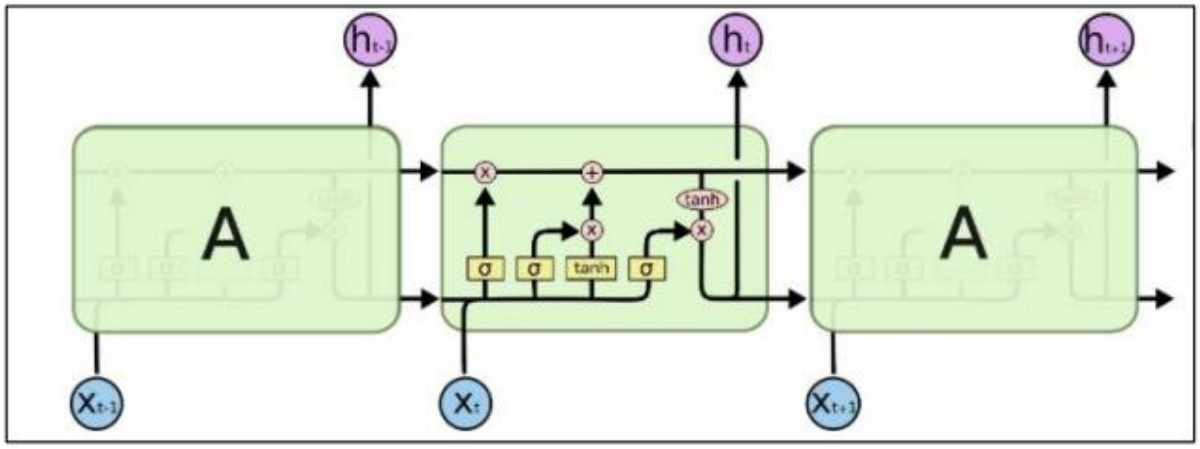
2.7 *Long Short Term Memory*

Sebuah sistem penyimpanan data yang disebut *long short term memory* (LSTM) mampu menganalisis, meramalkan, dan mengkategorikan informasi yang telah disimpan untuk waktu yang lama (Wiranda & Sadikin, 2019). *Long Short Term Memory* (LSTM) memiliki keuntungan karena mampu mempertahankan *sekuens long term* (ukuran data) yang sulit dilakukan dengan menggunakan pendekatan fitur konvensional (Wiranda & Sadikin, 2019).

Recurrent Neural Network (RNN) yang dimodifikasi disebut Long Short Term Memory yang

dibuat untuk memperkirakan variabel secara akurat. Keakuratan ramalan tergantung pada tingkat kesalahan prediksi, semakin rendah tingkat kesalahan, semakin akurat ramalannya (Owen et al., 2022).

Reccurent Neural Network (RNN) memiliki kekurangan yaitu kemampuan untuk mengelola informasi dalam periode yang lama. Untuk mengatasi permasalahan tersebut, *Reccurent Neural Network* (RNN) mengembangkan *Long Short Term Memory* (LSTM) yang mampu mengelola informasi dalam periode yang lama. Pertama kali yang mengusulkan *Long Short Term Memory* (LSTM) yaitu Sepp Hochreiter dan Jurgen Schmidhuber pada tahun 1997. *Long Short Term Memory* (LSTM) sangat dikenal dan banyak dipilih untuk prediksi berbasis waktu atau *time series* karena kemampuannya untuk memprediksi dalam waktu yang lama disbanding algoritma lain (Zahara et al., 2019). Struktur jaringan LSTM bisa dilihat pada gambar 2.10



Gambar 2.10 Struktur Jaringan LSTM
Sumber : (colah, 2015)

Struktur gerbang digunakan dalam model *Long Short Term Memory* (LSTM) untuk mempertahankan dan memperbarui status sel memori melalui penyaringan informasi. *Input gate, forget gate,* dan *output gate* termasuk di antara struktur *gate* yang ada. Tiga lapisan sigmoid dan satu lapisan tanh hadir di setiap sel memori (Qiu et al., 2020).

Fungsi sigmoid dapat dilihat pada persamaan 2.3 (Ma et al., 2015) :

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (2.3)$$

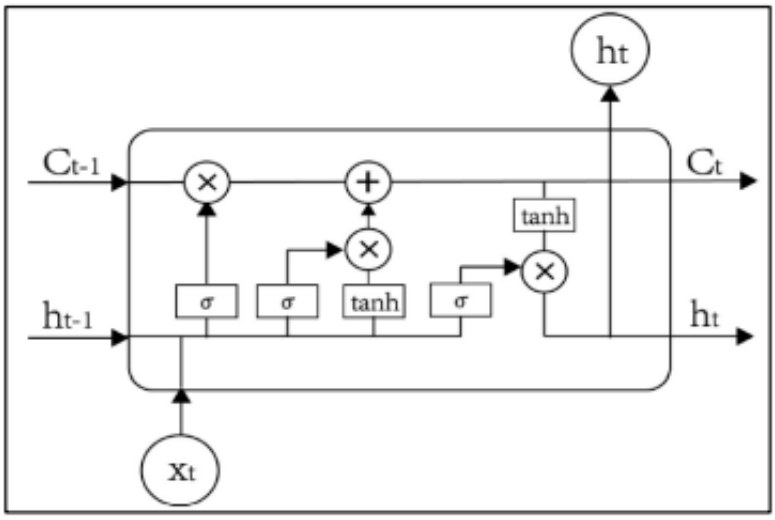
Fungsi tanh dapat dilihat sebagai berikut :

$$\tanh(x) = 2\sigma(2x) - 1 \quad (2.4)$$

Keterangan:

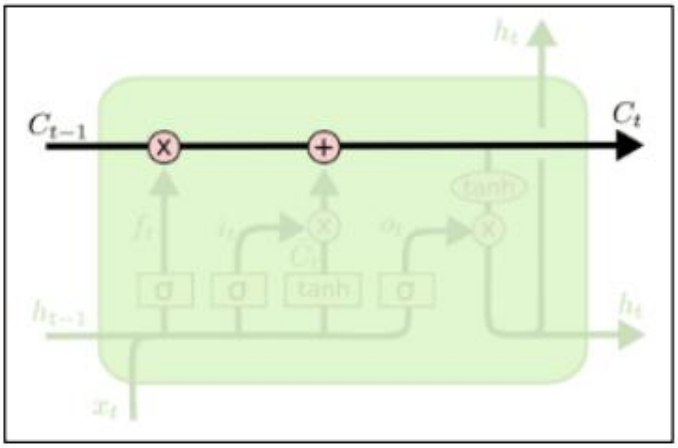
σ = Fungsi aktivasi sigmoid

x = Data *input*



Gambar 2.11 Struktur Dalam Satu Sel LSTM
Sumber : (Qiu et al., 2020)

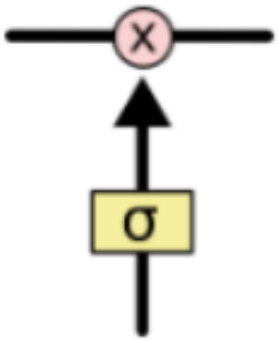
Pada gambar 2.11 diperlihatkan struktur dalam satu sel Long Short Term Memory (LSTM), khususnya jalur yang menghubungkan sel memori lama (C_{t-1}) ke sel memori baru (C_t), yang merupakan kunci *Long Short Term Memory* (LSTM). Garis horizontal yang menghubungkan semua level output di *Long Short Term Memory* (LSTM) adalah definisi sel memori. Nilai sel memori lama dapat ditransfer ke sel memori baru menggunakan jalur ini dengan upaya minimal.



Gambar 2.12 *Memory Cell*
Sumber : (colah, 2015)

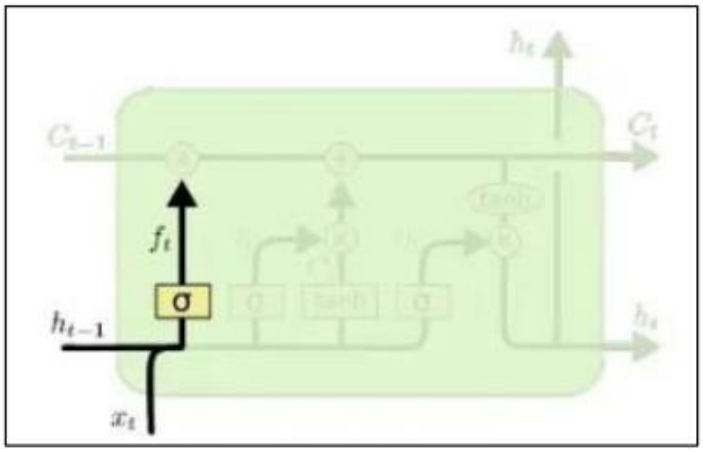
Kemampuan untuk menambah atau menghapus data masa lalu yang masuk ke sel saat ini adalah salah satu fungsi dari Long Short Term Memory (LSTM). Besarnya setiap komponen yang

harus dibiarkan masuk ditunjukkan dengan angka antara nol dan satu pada lapisan sigmoid. Definisi nol adalah "tidak lolos" tetapi definisi satu adalah "lolos" (colah, 2015).



Gambar 2.13 Lapisan Sigmoid
Sumber : (colah, 2015)

(colah, 2015) menjelaskan secara rinci langkah kerja *Long Short Term Memory* (LSTM), di bawah ini penjelasan langkah kerjanya itu sendiri. Pilih data yang ingin Anda hapus dari status sel terlebih dahulu. Hasilnya adalah angka antara 0 dan 1, yang ditentukan oleh lapisan sigmoid yang dikenal sebagai *forget gate* dan terlihat pada h_{t-1} dan x_t. Hasilnya dapat dibulatkan menjadi 1 jika keluaran sigmoid lebih besar dari 0,5 dan menjadi 0 jika kurang dari 0,5.



Gambar 2.14 Alur Informasi Pada *Forget Gate*
Sumber : (colah, 2015)

Informasi status sel mana yang harus dihapus dari model ditentukan oleh *forget gate*. Seperti terlihat pada Gambar 2.13, sel memori mengambil sebagai input informasi eksternal x_t dari saat ini dan *output* h_{t-1} dari momen sebelumnya, Informasi eksternal x_t dari momen saat ini sebagai *input* dan menggabungkannya dalam vektor panjang $[h_{t-1}, x_t]$ melalui transformasi σ

menjadi :

$$f_t = \sigma(W_f . [h_{t-1}, x_t] + b_f) \quad (2.5)$$

Keterangan:

f_t : *Forget gate*

σ : Fungsi sigmoid

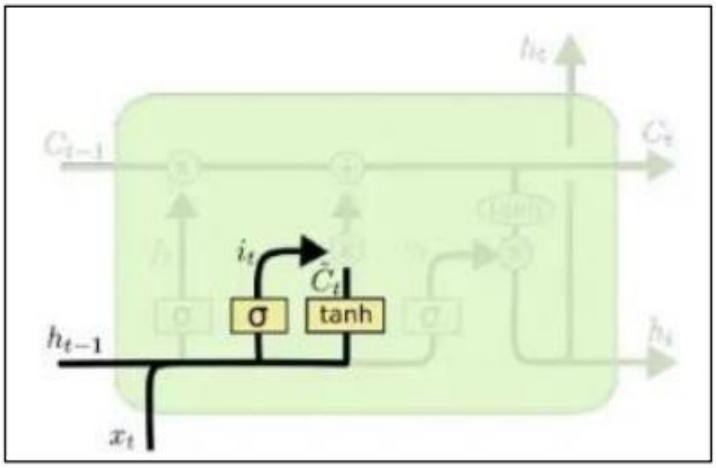
W_f : Nilai *weight* untuk *forget gate*

h_{t-1} : Nilai *output* sebelum orde ke t

x_t : Nilai *input* pada order ke t

b_f : Nilai bias pada *forget gate*

W_f dan b_f merupakan bobot dan bias *forget gate*. sedangkan σ merupakan fungsi sigmoid. Fungsi utama *forget gate* yaitu untuk merekam seberapa banyak status sel C_{t-1} dari waktu sebelumnya dicadangkan ke status sel C_t dari waktu saat ini. Kemudian yang akan ditampilkan oleh gerbang yaitu berupa nilai antara 0 dan 1. Berdasar pada h_{t-1} dan x_t, apabila menunjukkan nilai 1 maka artinya reservasi lengkap. Namun apabila menunjukkan nilai 0 maka artinya pembuangan lengkap. Penulisan $[h_{t-1}, x_t]$ merupakan operasi konkatenasi (penggabungan dua himpunan atau lebih) yang artinya menambahkan baris x_t dengan baris h_{t-1}.



Gambar 2.15 Alur Informasi Yang Melewati *Input Gate*
Sumber : (colah, 2015)

Langkah selanjutnya adalah memilih apakah ada data baru yang akan ditambahkan ke status sel. *Input gate* melayani dua tujuan. Pertama, menentukan keadaan sel yang perlu diperbarui dan dipilih oleh lapisan sigmoid. Kedua, ia menciptakan vektor kandidat baru C_t melalui lapisan *tanh*, yang berfungsi untuk mengatur berapa banyak informasi baru yang ditambahkan, seperti

yang ditunjukkan pada persamaan 2.6 dan 2.7 untuk memperbarui status sel memori:

$$i_t = \sigma(W_i.[h_{t-1}.x_t] + b_t) \quad (2.6)$$

$$C_t = tanh\ (W_i.[h_{t-1}, x_t] + b_c) \quad (2.7)$$

Keterangan:

i_t : *Input gate*

σ : Fungsi sigmoid

W_i : Nilai *weight* untuk *input gate*

h_{t-1} : Nilai *output* sebelum orde ke t

x_t : Nilai *input* pada orde ke t

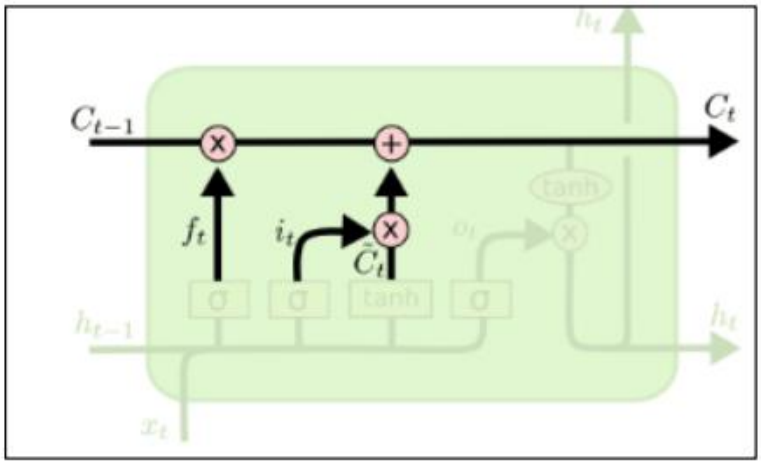
b_t : Nilai bias pada *input*

C_t : Nilai baru yang dapat ditambahkan ke *cell state*

b_c : Nilai bias pada *cell state*

C_{t-1} : *Cell state* sebelum orde ke t

f_t : *Forget gate*



Gambar 2.16 Memperbaharui Status Sel
Sumber : (colah, 2015)

Konversi keadaan sel lama C_{t-1} menjadi keadaan sel baru C_t digambarkan pada Gambar 2.15. Langkah-langkahnya adalah menambahkan operasi $i_t.C_t$, kemudian menghapus hal yang sebelumnya terhapus setelah beralih ke *state* lama menggunakan f_t. Bergantung pada berapa banyak nilai status yang ingin Anda ubah, nilai kandidat baru ini akan diskalakan. Persamaan 2.8 di bawah ini memungkinkan untuk perumusan ini :

$$C_t = f_t.C_{t-1} + i_t.C_t \quad (2.8)$$

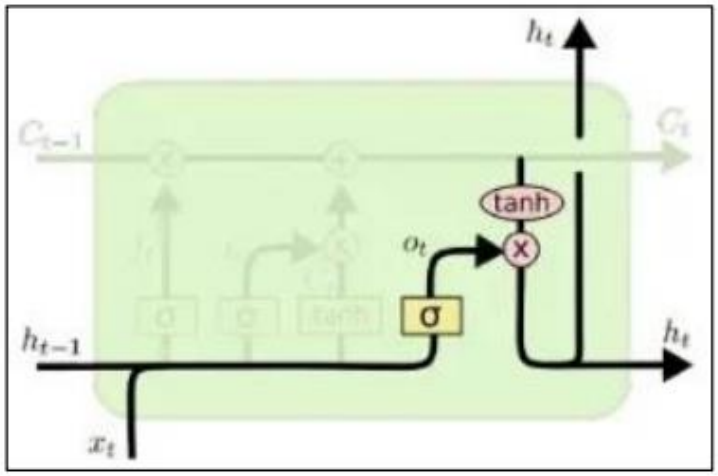
Keterangan:

C_t : *Cell state* orde ke t

f_t : *Forget gate*

C_{t-1} : *Cell state* sebelum orde ke t

i_t : *Input gate*



Gambar 2.17 Alur Informasi Yang Melewati *Output Gate*
Sumber : (colah, 2015)

Output gate mengatur berapa banyak keadaan sel saat ini yang dibuang. Lapisan sigmoid menentukan data keluaran awal, yang kemudian diolah dengan *tanh* dan dikalikan dengan keluaran lapisan sigmoid untuk mendapatkan *output* akhir:

$$O_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \quad (2.9)$$

Keterangan:

O_t : *Output gate*

σ : Fungsi sigmoid

W_o : Nilai *weight* untuk *output gate*

h_{t-1} : Nilai *output* sebelum orde ke t

x_t : Nilai *input* pada orde ke t

b_o : Nilai bias pada *output gate*

Nilai *output* akhir sel didefinisikan pada persamaan berikut:

$$h_t = O_t.tanh\,(C_t) \quad (2.10)$$

Keterangan:

h_t : Nilai *output* orde ke t

O_t : *Output gate*

$tanh$: Fungsi tanh

C_t : *Cell state*

2.8 RMSE (*Root Mean Square Error*)

Root Mean Square Error (RMSE) adalah rumus yang digunakan untuk memperkirakan suatu pengamatan dan menilai penyimpangan dari nilai yang diharapkan dari suatu model. Akar kuadrat dari kesalahan kuadrat rata-rata menghasilkan *Root Mean Square Error*. Adanya angka RMSE yang rendah menunjukkan keakuratan metode estimasi kesalahan pengukuran. Teknik pendugaan dengan RMSE kecil diyakini lebih akurat daripada teknik dengan RMSE besar (Hariany et al., 2021).

Dalam dunia peramalan, metode *Root Mean Square Error* sering digunakan untuk menentukan apakah metode peramalan yang dimaksud layak atau tidak untuk digunakan untuk memperkirakan. Metode untuk mengurangi *Root Mean Square Error* melibatkan membandingkan nilai aktual dan prediksi, diikuti dengan penggunaan banyak data (Ariyadi & Effendi, 2022). Rumus untuk persamaan RMSE terlihat pada persamaan 2.11

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\widehat{Yi} - Yi)^2} \qquad (2.11)$$

Keterangan:

$\tilde{y}i$ = Nilai hasil peramalan

yi = Nilai aktual / Nilai sebenarnya

n = Jumlah data

2.9 Tabel Penelitian Terkait

Pada bagian penelitian terkait ini berisi penelitian-penelitian yang telah dilakukan sebelumnya dan menjadi sumber rujukan pada penelitian ini. Tabel 2.1 menunjukkan penelitian yang pernah dilakukan. Beberapa penelitian yang ada dalam tabel 2.1 dan tabel 2.2 hanya menyimpulkan garis besar dari penelitian yang sudah diteliti.

Tabel 2.1 Penelitian Terdahulu Menggunakan Metode LSTM

No.	Metode	Penulis	Judul	Hasil
1.	LSTM	(Khumaidi et al., 2020)	Pengujian Algoritma Long Short Term Memory Untuk Prediksi Kualitas Udara dan Suhu Kota Bandung	Pembuatan model LSTM dan penerapannya pada data time series dengan 4 hidden layer, 32 batch size, Adam sebagai pengoptimal, dan epoch senilai 1000 menghasilkan perhitungan data kualitas udara di kota Bandung untuk parameter PM10, ISPU, serta suhu dan kelembaban, yang menunjukkan bahwa model menghasilkan cukup banyak akurasi prediksi untuk 3 parameter (suhu, kelembaban, ISPU). Hal ini ditunjukkan oleh nilai RMSE yang diantisipasi, yang lebih rendah dari nilai standar deviasi dataset uji. Namun, prakiraan empat parameter uji untuk kelembapan adalah yang paling akurat, diikuti oleh prakiraan suhu, ISPU, dan PM10.
2.	LSTM	(Rizki et al., 2020)	Implementasi Deep Learning Menggunakan Arsitektur Long Short Term Memory Untuk Prediksi Curah Hujan Kota Malang	Arsitektur Long Short Term Memory (LSTM) digunakan dalam implementasi Deep Learning untuk prediksi curah hujan di Malang, dan setelah dilakukan analisis, desain, implementasi, dan pengujian, ternyata hasil prediksi cukup akurat. Dengan demikian, dapat dikatakan bahwa arsitektur Deep Learning dengan Long Short Term Memory (LSTM) dapat berfungsi cukup optimal pada tahap akhir proyek penelitian.
3.	LSTM	(Lipton et al., 2016)	Learning To Diagnose With LSTM Reccurent Neural Networks - Revised	Serangkaian pengamatan multivariat yang dibuat untuk penyelidikan data medis klinis ini, terutama di unit perawatan intensif (ICU), digunakan. Data sulit untuk ditambang dengan sukses sambil mungkin memberikan banyak wawasan karena durasinya yang bervariasi, pengambilan sampel yang tidak menentu, dan data yang hilang. RNN, khususnya LSTM, adalah model pembelajaran yang efektif dan banyak digunakan untuk data urutan. Mereka secara akurat mensimulasikan tangkapan ketergantungan jarak jauh dan urutan panjang yang berbeda. Penyelidikan empiris pertama ditawarkan dalam penelitian.
4.	LSTM	(Badriyah et al., 2022)	Prediksi Curah Hujan Menggunakan Long Short Term	Karena penelitian ini dilakukan dengan pengujian memanfaatkan perbandingan 2 algoritma seperti RNN dan GRU, maka hasilnya menunjukkan bahwa prediksi curah

No.	Metode	Penulis	Judul	Hasil
			Memory	hujan di kota Surabaya menggunakan Long Short Term Memory (LSTM) beroperasi secara optimal dibandingkan penelitian sebelumnya.
5.	LSTM	(Aldi et al., 2018)	Analisis dan Implementasi Long Short Term Memory Neural Network untuk Prediksi Harga Bitcoin	Arsitektur Long Short Term Memory Neural Networks merupakan teknik jaringan syaraf tiruan yang digunakan untuk membangun sistem pada penelitian ini. Agar teknik ini menghasilkan hasil prediksi yang akurat, parameter yang tepat harus digunakan. menganalisis dampak dari banyak faktor, termasuk kuantitas pola deret waktu, jumlah *hidden neuron*, jumlah maksimum epoch, dan susunan data pelatihan dan pengujian, pada ketepatan prediksi yang dibuat. Dengan tingkat akurasi rata-rata 93,5% untuk data pengujian, temuan analisis menunjukkan bahwa sistem yang dikembangkan mampu membuat prediksi akurat tentang harga Bitcoin.
6.	LSTM	(Lattifia et al., 2022)	Model Prediksi Cuaca Menggunakan Metode LSTM	Parameter hasil penelitian ini, epoch dan ukuran batch, adalah yang berdampak pada hasil dari metode LSTM yang diantisipasi dalam penelitian ini. Ukuran batch 50 dan epoch 100 memberikan akurasi rata-rata terbaik, sedangkan nilai RMSE dan MAPE terbaik masing-masing adalah 1,7444 dan 1,9499%.
7.	LSTM	(Wiranda & Sadikin, 2019)	Penerapan Long Short Term Memory Pada Data Time Series Untuk Memprediksi Penjualan Produk PT. Metiska Farma	Untuk meramalkan penjualan masa depan perusahaan, penelitian ini menggunakan Long Short Term Memory pada dataset penjualan obat "X" pada PT. Metiska Farma. Eksperimen kelima tampil lebih baik berdasarkan hasil skenario eksperimen dua parameter. Menurut temuan penelitian, pendekatan yang paling efektif memanfaatkan LSTM dengan rentang interval [-1.1], 1500 epoch, 90% data pelatihan, dan 10% data uji. Rata-rata kesalahan model menggunakan MAPE antara nilai proyeksi dan nilai harian riil terkecil adalah 12%. Metode ini menghasilkan hasil perhitungan RMSE dalam bentuk rupiah sebesar 13.762.154,00.

No.	Metode	Penulis	Judul	Hasil
8.	LSTM	(Freecenta et al., 2022)	Prediksi Curah Hujan Di Kab. Malang Menggunakan LSTM (Long Short Term Memory)	Temuan penelitian ini menunjukkan bahwa skenario uji coba pertama menggunakan empat *layer* LSTM dengan masing-masing 100 neuron. Skenario uji coba kedua memiliki dua *layer* LSTM, dengan masing-masing 50 neuron. Dalam penelitian ini, nilai akurasi model terbesar adalah MAE sebesar 7,90, RMSE sebesar 10,16, dan MSE sebesar 103,37.

Tabel 2.2 Penelitian Terdahulu Menggunakan Metode Lain

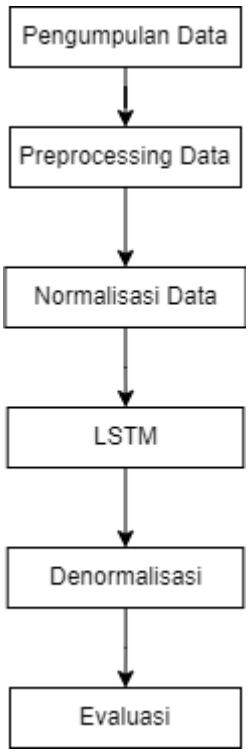
No.	Metode	Penulis	Judul	Hasil
1.	Artificial Neural Network (ANN)	(Panggabean et al., 2021)	Prediksi Tinggi Curah Hujan Dan Kecepatan Angin Berdasarkan Data Cuaca Dengan Penerapan Algoritma Artificial Neural Network (ANN).	Temuan penelitian menunjukkan bahwa pendekatan Back-Pagation Neural Network, dengan nilai RMSE 0,079535, 20 neuron, 1000 epoch, dan validasi split 0,1, merupakan pemodelan terbaik untuk memprediksi curah hujan.
2.	MLSTM-FCN	(Karim et al., 2019)	Multivariate LSTM-FCNs For Time Series Classification	Klasifikasi deret waktu multivariat telah menarik banyak minat baru-baru ini. Untuk meningkatkan akurasi lebih banyak lagi, penelitian ini menyarankan untuk mengubah model klasifikasi deret waktu univariat LSTM-FCN dan Attention LSTM-FCN (ALSTM-FCN) saat ini menjadi model klasifikasi deret waktu multivariat dengan memasukkan blok konvolusi penuh dengan pemerasan dan eksitasi blok. Model yang disarankan berperforma lebih baik daripada sebagian besar model sementara membutuhkan sedikit pra-pemrosesan. Pendekatan yang disarankan berkinerja baik dalam berbagai aplikasi klasifikasi deret waktu multivariat yang menantang,

				termasuk pengenalan tindakan atau aktivitas. Model yang disarankan juga sangat efektif selama pengujian.

BAB III
MEOTODOLOGI PENELITIAN

3.1 Tahapan Penelitian

Pada gambar 3.1 ini menampilkan tahapan penelitian.



Gambar 3.1 Tahapan Penelitian

Berikut adalah tahapan-tahapan penelitian yang dilakukan dalam mengerjakan penelitian ini seperti Pengumpulan Data, *Preprocessing* Data, Normalisasi Data, LSTM, Denormalisasi dan Evaluasi.

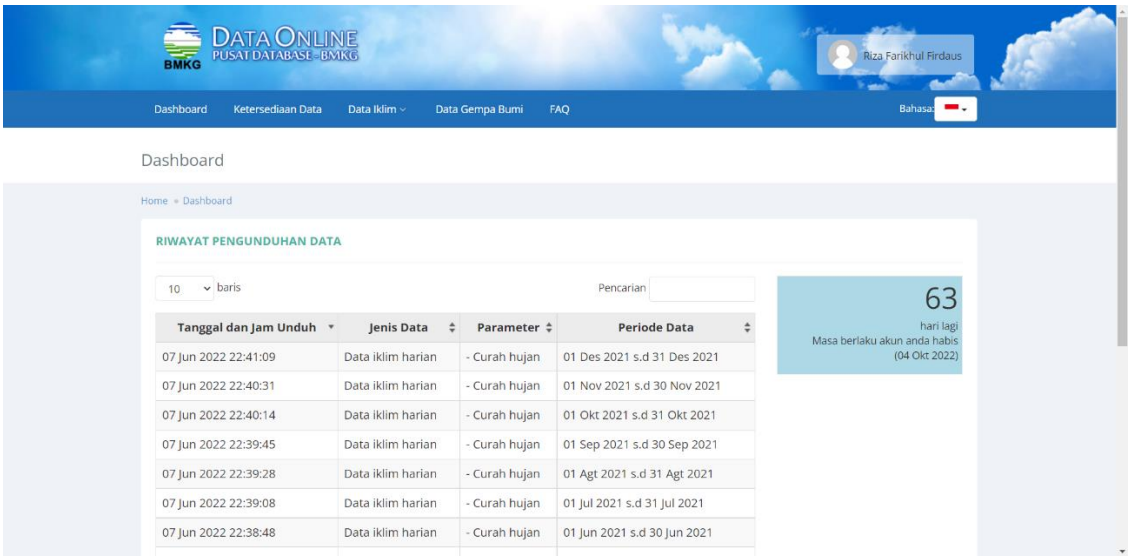
3.2 Uraian Metodologi

Di bawah ini merupakan uraian dari metodologi yang dilakukan pada penelitian ini.

3.2.1 Pengumpulan Data Curah Hujan

Pada tahap ini, data akan dikumpulkan sebelum melanjutkan ke proses selanjutnya. Menurut (Rizki et al., 2020; Soekendro, 2021), data yang dibutuhkan yaitu data curah hujan dari tahun 2017-2021 di Kota Bandung yang didapatkan dari website BMKG (http://dataonline.bmkg.go.id/dashboard_user). Tampilan *dashboard* dari pusat database BMKG

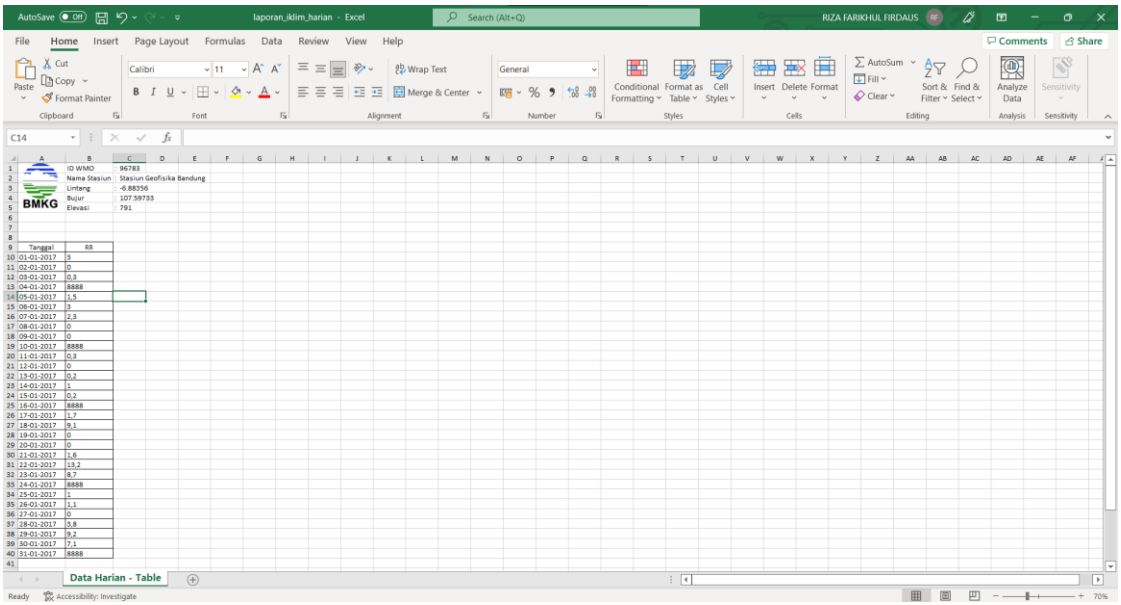
bisa dilihat di Gambar 3.1 dan contoh datanya bisa dilihat di Gambar 3.2. Data tersebut dijadikan acuan sebagai informasi yang akan digunakan pada proses pelatihan data (data training) dan data testing menggunakan metode *Long Short Term Memory* (LSTM).



Tanggal dan Jam Unduh	Jenis Data	Parameter	Periode Data
07 Jun 2022 22:41:09	Data iklim harian	- Curah hujan	01 Des 2021 s.d 31 Des 2021
07 Jun 2022 22:40:31	Data iklim harian	- Curah hujan	01 Nov 2021 s.d 30 Nov 2021
07 Jun 2022 22:40:14	Data iklim harian	- Curah hujan	01 Okt 2021 s.d 31 Okt 2021
07 Jun 2022 22:39:45	Data iklim harian	- Curah hujan	01 Sep 2021 s.d 30 Sep 2021
07 Jun 2022 22:39:28	Data iklim harian	- Curah hujan	01 Agt 2021 s.d 31 Agt 2021
07 Jun 2022 22:39:08	Data iklim harian	- Curah hujan	01 Jul 2021 s.d 31 Jul 2021
07 Jun 2022 22:38:48	Data iklim harian	- Curah hujan	01 Jun 2021 s.d 30 Jun 2021

Gambar 3.2 Dashboard Pusat Database BMKG

Sumber : (BMKG, 2022)



Gambar 3.3 Tampilan Dataset di Excel

Sumber : (BMKG, 2022)

3.2.2 Preprocessing Data

Menyiapkan data yang akan digunakan untuk pelatihan data (data *training*) dikenal sebagai

data *preprocessing*. Ini melibatkan persiapan data terlebih dahulu untuk menangani data yang hilang atau kosong dalam berbagai metode, seperti menghitung rata-rata atribut untuk kelas yang sama. Data kemudian dinormalisasi menggunakan *MinMaxScaler* dengan rentang (0, 1). Dengan menerapkan modifikasi linier pada data awal, pendekatan Min-Max menormalkan data akan lebih mudah bagi peneliti untuk menggunakan kelas prapemrosesan *MinMaxScaler* dari *library scikit-learn* (sklearn) untuk menormalkan kumpulan data sebagai hasilnya. Rumus *MinMaxScaler* adalah sebagai berikut.

$$X' = \frac{(x-Xmin)}{(Xmax-Xmin)} \quad (3.1)$$

Keterangan:

X’ = Hasil normalisasi data

Xmax = Data maksimum dari data X

Xmin = Data minimum dari data X

Setelah tahap *preprocessing* data selesai dilakukan kemudian dataset dibaagi menjadi dua, dataset *training* dan dataset *testing*. Pada tahapan training terdiri dari beberapa tahap yaitu: Training LSTM, LSTM Model, Save LSTM Model. Sedangkan Load LSTM Model Predicted Denormalisasi, Evaluasi.

3.2.3 Normalisasi Data

Normalisasi data adalah tahapan yang digunakan untuk menghilangkan data tidak terstruktur dan redundansi untuk memastikan penyimpanan data logis. Normalisasi min-max sering memungkinkan transformasi data dengan skala yang berbeda, mencegah satu dimensi mendominasi statistik dan menghilangkan persyaratan untuk KNN dan ANN, yang keduanya bergantung pada asumsi distribusi yang kuat. Normalisasi min-max, bagaimanapun, tidak menangani outlier dengan baik. Standardisasi, di sisi lain, memudahkan pengguna untuk menangani oulier dan mempercepat konvergensi untuk beberapa teknik komputasi, seperti penurunan gradien.

3.2.4 Long Short Term Memory Network

Long Short Term Memory (LSTM) biasanya disebut sebagai jaringan saraf dengan desain

yang dapat disesuaikan, yang memungkinkannya berubah bentuk sesuai kebutuhan. Selain itu, teknik RNN (*Recurrent Neural Network*) adalah turunan dari *long short term memory* (LSTM). Rentang nilai dalam arsitektur berubah dari satu lapisan ke lapisan berikutnya sebagai akibat dari masalah gradien RNN yang menghilang dan berkembang. Saat berhadapan dengan gradien yang menghilang dan meledak, LSTM dibuat dan direkayasa untuk menghindari masalah hilangnya gradien dari RNN. *Input layer*, *output layer*, dan *hidden layer* membentuk arsitektur LSTM. Sel memori membentuk *hidden layer*. Setiap sel memori memiliki tiga gerbang (*gate*).

3.2.5 Denormalisasi

Setelah mendapatkan hasil prediksi, maka sebelum menghitung akurasi hasil prediksi harus dilakukan denormalisasi yaitu data diubah menjadi nilai real. Dikarenakan data hasil prediksi masih berupa data berbentuk range interval yang dilakukan pada normalisasi data. Di bawah ini adalah rumus untuk denormalisasi.

$$d = d'(max - \text{mix}) + min \tag{3.2}$$

Keterangan:

d = Nilai hasil denormalisasi

d’ = Nilai data normalisasi

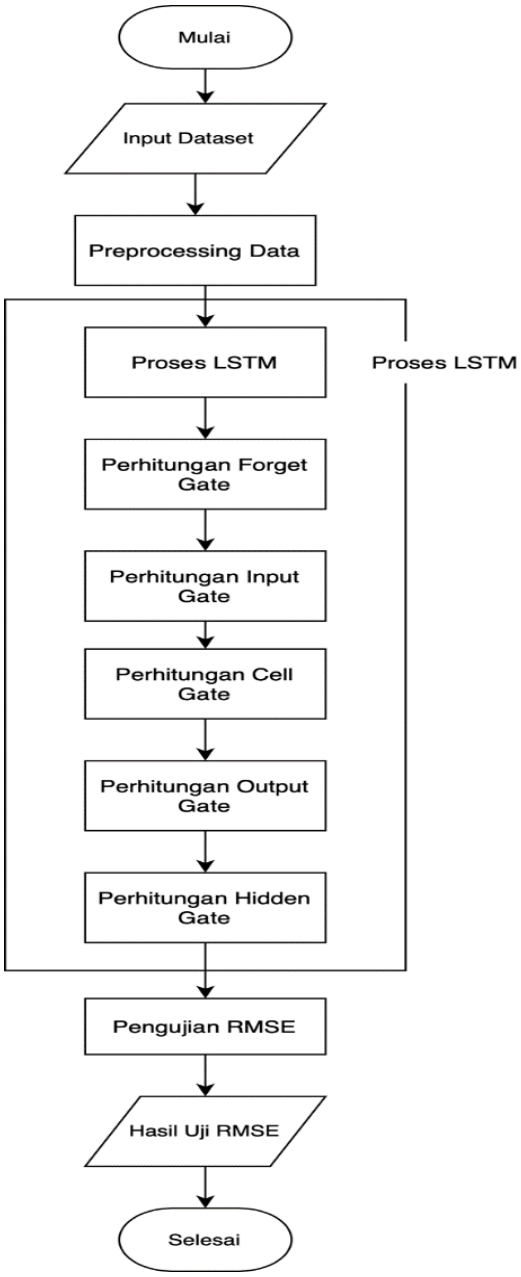
max = Nilai maksimum dari data actual

min = Nilai minimum dari data aktual

3.2.6 Evaluasi

Cara lain untuk menilai keakuratan keluaran prediksi model adalah dengan melihat *Root Mean Square Error* (RMSE) dari metodologi prediksi. Rerata kuadrat jumlah error pada model prediksi menghasilkan nilai RMSE. *Root Mean Square Error* (RMSE) adalah metode sederhana yang sering diterapkan dalam berbagai investigasi prediksi.

3.3 Perhitungan Manualisasi LSTM



Gambar 3.4 Diagram Alir LSTM

Dalam melakukan perhitungan manualisasi LSTM yang terdapat pada Gambar 3.4 terdapat tahapan-tahapan yang harus dilakukan, seperti Input Dataset, Preprocessing Data, Proses LSTM (Perhitungan Forget Gate, Perhitungan Input Gate, Perhitungan Cell Gate, Perhitungan Output Gate, Perhitungan Hidden Gate), Pengujian RMSE, Hasil Uji RMSE, Selesai. Berikut merupakan uraian pada tahapan LSTM:

3.3.1 Dataset

Data yang digunakan pada manualisasi adalah data curah hujan yang diambil dari stasiun geofisika Bandung selama tahun 2017-2021 seperti yang terlihat pada Tabel 3.1 Dataset dibawah ini merupakan contoh salah satu dataset curah hujan di bulan November 2021. RR merupakan data curah hujan yang diperoleh dari BMKG.

Tabel 3.1 Dataset Curah Hujan Bulan November 2021

Tanggal	RR
01-11-2021	20,9
02-11-2021	10,6
03-11-2021	65,6
04-11-2021	27,7
05-11-2021	18,2
06-11-2021	5
07-11-2021	0
08-11-2021	11
09-11-2021	12,8
10-11-2021	34,4
11-11-2021	38,5
12-11-2021	0,3
13-11-2021	16,4
14-11-2021	44,2
15-11-2021	8,8
16-11-2021	16,1
17-11-2021	3,7
18-11-2021	1,8
19-11-2021	4,7
20-11-2021	30,3
21-11-2021	8,3
22-11-2021	14,6
23-11-2021	8
24-11-2021	5,1
25-11-2021	0
26-11-2021	20
27-11-2021	3,6
28-11-2021	33,8
29-11-2021	9
30-11-2021	16,9

3.3.2 Preprocessing Data

Pada tahap ini, peneliti meminimalisir *error* dengan cara normalisasi. Normalisasi berguna untuk menghindari banyak anomali data dan inkonsistensi data. Kegunaan lainnya yaitu untuk merubah ukuran data menjadi lebih kecil tanpa harus merubah data asli. Teknik normalisasi yang digunakan pada penelitian ini yaitu *min-max scaling*.

$$X' = \frac{(x - Xmin)}{(Xmax - Xmin)} \tag{3.3}$$

Keterangan:

X’ = Hasil normalisasi data

Xmax = Data maksimum dari data X

Xmin = Data minimum dari data X

Berikut merupakan contoh perhitungan normalisasi pada 1 November 2021:

$X' = \frac{(20{,}9-0)}{(8888-0)}$ = 0,002351

Tabel 3.2 Hasil Normalisasi Tahun 2021 bulan November

Normalisasi
0,002351
0,001193
0,007381
0,003117
0,002048
0,000563
0
0,001238
0,00144
0,00387
0,004332
3,38E-05
0,001845
0,004973
0,00099
0,001811

0,000416
0,000203
0,000529
0,003409
0
0
1
1
0
0,00225
0,000405
0,003803
0,001013
0,001901

Nilai normalisasi yang telah didapatkan kemudian dijadikan ke dalam pola *time series*, dengan X adalah curah hujan yang dianalisis dengan tahapan-tahapan pelatihan LSTM adalah sebagai berikut

3.3.3 Proses LSTM

(a) Setelah mendapatkan data pada tahap sebelumnya, selanjutnya yaitu memproses data *training*.

(b) Selanjutnya dilakukan inisialisasi bobot untuk *forget gate*, *input gate*, *cell state*, dan *output gate* serta menginisialisasikan nilai bias pada setiap *gate* ($W_{f,}$ $W_{i,}$ $W_{ct,}$ W_o dan bias ($b_{f,}$ $b_{i,}$ $b_{ct,}$ b_o). Untuk mencari nilai bobot W, gunakan persamaan (3.4). Nilai bobot yang telah diperoleh melalui persamaan (3.3) selanjutnya diinisialisasikan sebagai bobot $W_{f,}$ $W_{i,}$ $W_{ct,}$ W_o

$$W = \left(-\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}}\right) \quad (3.4)$$

Keterangan:

W = bobot

d = banyaknya data

$$W = \left(-\frac{1}{\sqrt{30}}, \frac{1}{\sqrt{30}}\right) = (-0{,}18574;\ 0{,}18574).$$

Nilai d=30 diperoleh dari banyaknya hari pada bulan November 2021

(c) Menghitung semua fungsi *gates* unit pada setiap neurons merupakan bagian dari proses data *training* menggunakan LSTM. Dengan berurut fungsi *gates* yang akan dihitung adalah *forget gates* (ft) dengan Persamaan (3.5), fungsi *input gates* (t) dengan Persamaan, *cell state*, *output gate*, dan *hidden gate*.

LSTM memiliki *gates* (gerbang) yang terdiri dari *sigmoid layer* dan *pointwise operation*. *Gates* ini dapat menambah dan menghapus informasi yang akan diteruskan ataukah diberhentikan. Hasil *output* dari *sigmoid layer* akan dipadatkan menjadi *range* [0.1]. Angka 0 berarti informasi akan di*stop* (diberhentikan) sedangkan angka 1 berarti diteruskan. Persamaan sigmoid bisa dilihat pada Persamaan 3.5

$$\sigma\,(x) = \frac{1}{(1+ e^{-x})} \tag{3.5}$$

3.3.4 Perhitungan *Forget Gate*

Forget gate merupakan gerbang awal masuknya informasi yang akan diproses menggunakan algoritma LSTM. *Forget gate* ini akan memutuskan informasi apa yang akan di hilangkan dari *cell state*. Ini terjadi karena adanya *sigmoid layer* yang menghasilkan *output* angka antara 0 dan 1. *Forget Gate* memiliki persamaan rumus (3.6)

$$f_t = \sigma\,(W_f\,.\,[s_{t-1}, x_t] + b_f) \tag{3.6}$$

Keterangan:

W_f = Bobot dari *forget gate*.

S_{t-1} = *State* sebelumnya atau *state* pada waktu t-1

X_t = *Input* pada waktu t.

σ = Fungsi aktivasi *sigmoid*

Contoh perhitungan:

$f_1 = \sigma\,(0.182574\,[0, 0] + 1) = 1.0004$

$f_1 = \frac{1}{(1+ e^{-1,0004})} = 0.7311$

Nilai 0,182574 merupakan nilai W (bobot) yang sudah dihitung menggunakan persamaan (3.4), Xt merupakan nilai variabel yang diperoleh dari dataset dan b_f merupakan nilai bias forget gate.

	1,00042932
Forget gate	0,73114298

Nilai 1.0042932 merupakan hasil dari forget gate sebelum memasuki fungsi sigmoid, sedangkan 0.73114298 merupakan hasil forget gate yang telah diproses fungsi sigmoid.

3.3.5 Perhitungan *Input Gate*

Input gate berperan mengambil *output* sebelumnya dan *input* baru serta melewatkan mereka melalui lapisan *sigmoid*. *Gate* ini mengembalikan nilai 0 atau 1.

$$i_t = \sigma\ (W_i\ .\ [s_{t-1},\ x_t] + b_i) \qquad (3.7)$$

Keterangan:

W_i = Bobot dari input *gate*.

S_{t-1} = *State* sebelumnya atau *state* pada waktu t-1

X_t = *Input* pada waktu t.

σ = Fungsi aktivasi *sigmoid*

Contoh perhitungan:

$i_1 = \sigma\ (0.182574\ [0,\ 0] + 0.5) = 0{,}50004$

$i_1 = \frac{1}{(1+\ e^{-0{,}50004})} \qquad = 0.6225$

Nilai 0,182574 merupakan nilai W (bobot) yang sudah dihitung menggunakan persamaan (3.2), Xt merupakan nilai variabel yang diperoleh dari dataset dan b_i merupakan nilai bias *input gate*.

	0,50042932
Input gate	0,622560218

Nilai 0.50042932 merupakan hasil dari *input gate* sebelum memasuki fungsi sigmoid, sedangkan 0,622560218 merupakan hasil *input gate* yang telah diproses fungsi sigmoid.

3.3.6 Perhitungan *Cell Gate*

$$\bar{C}_t = \tanh\ (W_c\ .\ [h_{t-1},\ x_t] + b_c) \qquad (3.8)$$

Keterangan

$\bar{C}_t$ =*Intermediate cell state*.

Wc = Bobot dari *cell state*.

$St-1$ = *State* sebelumnya atau *state* pada waktu t– 1.

Xt = *Input* pada waktu t.

Contoh perhitungan:

$\bar{C}_t$= σ (0.182574 [0, 0] + 0) = 0.0004

$\bar{C}_t$= tanh (0,0004) = 0,0004

Nilai 0,182574 merupakan nilai W (bobot) yang sudah dihitung menggunakan persamaan (2.2), Xt merupakan nilai variabel yang diperoleh dari dataset dan b_i merupakan nilai bias *input gate*, serta nilai 0,0004 merupakan hasil dari perhitungan *cell state*.

3.3.7 Perhitungan *Output Gate*

Output gate mengontrol seberapa banyak *state* yang lewat ke *output* dan bekerja dengan cara yang sama dengan *gate* lainnya. Dan terakhir menghasilkan *cell state* yang baru (ht). Rumus dari ot dan ht adalah:

$$o_t = \sigma\ (W_f\ .\ [h_{t-1}, x_t] + b_o) \tag{3.9}$$

Keterangan:

Wo = Bobot dari *output gate*.

$St{-}1$ = *State* sebelumnya atau *state* pada waktu t - 1.

Xt = *Input* pada waktu t.

σ = Fungsi aktivasi *sigmoid*.

Contoh Perhitungan :

o_1= σ (0.182574 [0, 0] + 0,1)

$o_1 = \frac{1}{(1+ e^{-0,50004})}$ = 0,525

Xt merupakan nilai variabel yang diperoleh dari dataset dan b_o merupakan nilai bias *output gate*, serta nilai 0,10004 merupakan hasil dari perhitungan *output gate*.

	0,10042932
Output gate	0,525086249

3.3.8 Perhitungan *Hidden Gate*

Hidden gate merupakan hasil akhir ataupun prediksi yang telah diproses di setiap gerbang LSTM. Hasil *hidden gate* ini merupakan nilai prediksi.

$$ht = o_t * tanh\ (c_t) \tag{3.10}$$

Keterangan:

O_t = Hasil *output gate*
C_t = *Cell state*

Contoh Perhitungan :

h_t= 0,525 . tanh (0,00014)

Langkah-langkah perhitungan LSTM menggunakan rumus-rumus telah diuraikan satu persatu pada bagian ini, dengan menggunakan sampel data pada tanggal 1 November 2021. *Cell state* adalah kunci utama di dalam metode LSTM. *Cell state* adalah garis horizontal yang melewati bagian atas diagram yang menghubungkan semua *output* layer pada LSTM. Hasil yang diperoleh pada masing-masing *gate*, ditunjukkan pada Tabel 3.3.

Tabel 3.3 Nilai Perhitungan *Gate* LSTM Pada 1 November 2021

	1,00042932
Forget gate	0,73114298
	0,50042932
Input gate	0,622560218
	0,00042932
Ct	0,00042932
	0,10042932
Output gate	0,525086249
Cell state	0,000267278
ht	0,000140344

Perhitungan dilakukan selama bulan Oktober sampai Desember pada data Tahun 2021 ($n_{prediksi}$ = 90 hari), diperoleh dengan hasil dari perhitungan pada ht, yang merupakan nilai prakiraan curah hujan seperti yang telah dihitung dengan menggunakan persamaan 3.10, dimana variabel yang diperhitungkan adalah RR yang merupakan data curah hujan.

Tabel 3.4 Hasil Perhitungan ht Pada bulan November 2021

Tanggal	ht
1	0,00014
2	0,000182
3	0,000585
4	0,000649
5	0,000635
6	0,000536

7	0,000424
8	0,000409
9	0,000409
10	0,000555
11	0,000697
12	0,000553
13	0,370178
14	0,232581
15	0,187799
16	0,15152
17	0,121473
18	0,096993
19	0,077258
20	0,061619
21	0,048886
22	0,038756
23	0,102542
24	0,152651
25	0,114125
26	0,091122
27	0,072519
28	0,057855
29	0,045952
30	0,036543

3.3.9 Pengujian RMSE

Evaluasi hasil akan menentukan apakah prediksi yang dibuat pada metode di atas memberikan hasil terbaik. Dalam penelitian ini, peneliti menggunakan RMSE (*Root Mean Square Error*) untuk memverifikasi keakuratan hasil prediksi. Berikut adalah contoh perhitungan manual menggunakan RMSE dengan data pengujian pada tanggal 1 November 2021.

Perhitungan RMSE

Di mana Yi merupakan nilai curah hujan, dan $\bar{Y}\imath$ adalah hasil prediksi, yang berupa nilai ht.

RMSE = $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\widehat{Y}\imath - Yi)^2}$ (3.11)

Keterangan:

$\tilde{y}i$: Nilai hasil peramalan

yi : Nilai aktual / Nilai sebenarnya

N : Jumlah data

Contoh perhitungan

RMSE = $\sqrt{14{,}56^2}$ = 3,81

Berdasarkan hasil perhitungan dalam memprediksi curah hujan selama 90 hari dengan menggunakan LSTM memberikan hasil yang cukup akurat dengan nilai RMSE yang rendah sesuai dengan penelitian yang dilakukan oleh Ashari & Sadikin, 2020 yang menyatakan bahwa nilai RMSE terendah menggambarkan bahwa nilai tersebut merupakan prediksi terbaik.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Spesifikasi Kebutuhan Hardware dan Software

Di bawah ini dijelaskan *hardware* dan *software* yang dibutuhkan untuk melakukan penelitian ini.

4.1.1 Kebutuhan Hardware

Hardware yang digunakan untuk melakukan penelitian ini adalah sebagai berikut:

1. Laptop Asus Vivobook S430UN
2. RAM 12 GB
3. Processor Intel Core i5 8th Gen

4.1.2 Kebutuhan Software

Software yang digunakan untuk melakukan penelitian ini adalah sebagai berikut:

1. Sistem Operasi Windows 10
2. Python
3. Text Editor (Jupyter Notebook)

4.2 Membaca Dataset Curah Hujan

Tahapan ini merupakan tahapan dimana akan membaca data curah hujan yang digunakan, dapat dilihat pada Gambar 4.1, seperti dibawah ini:

```
datapath  = 'Data Curah Hujan/'
full_path = []
for path, subdirs, files in os.walk(datapath):
    for name in files:
        src = os.path.join(path, name)
        full_path.append(src)
df_path = pd.DataFrame({'Path':full_path})
df_path
```

Gambar 4.1 *Source Code* Proses Membaca Dataset

Hasil dari *source code* pada gambar 4.1 dapat dilihat pada Gambar 4.2 yang menampilkan dataset curah hujan yang digunakan.

	Path
0	Data Curah Hujan/Curah Hujan 2017/laporan_ikli...
1	Data Curah Hujan/Curah Hujan 2017/laporan_ikli...
2	Data Curah Hujan/Curah Hujan 2017/laporan_ikli...
3	Data Curah Hujan/Curah Hujan 2017/laporan_ikli...
4	Data Curah Hujan/Curah Hujan 2017/laporan_ikli...
5	Data Curah Hujan/Curah Hujan 2017/laporan_ikli...
6	Data Curah Hujan/Curah Hujan 2017/laporan_ikli...
7	Data Curah Hujan/Curah Hujan 2017/laporan_ikli...
8	Data Curah Hujan/Curah Hujan 2017/laporan_ikli...
9	Data Curah Hujan/Curah Hujan 2017/laporan_ikli...
10	Data Curah Hujan/Curah Hujan 2017/laporan_ikli...

Gambar 4.2 Dataset Curah Hujan

4.3 Transform Raw Dataset

Tahap ini merupakan tahap mentransformasikan raw dataset atau kumpulan dataset mentah untuk dilakukan pemrosesan data. Gambar 4.3 merupakan *source code* untuk membaca data. *def process_data_raw(raw_data)* merupakan fungsi pemanggilan untuk proses data. Data akan dibuat menjadi 2 kolom dengan kolom pertama tanggal dan kolom kedua yairu RR (curah hujan). Berikut *source code* proses transform raw dataset, sebagai berikut:

```
def process_data_raw(raw_data):
    data = {'Tanggal':list(), 'RR':list()}
    format_date = "%d-%m-%Y"
    for index, row in raw_data.iterrows():
        try:
            date =
datetime.strptime(row['Tanggal'], format_date)
            data['Tanggal'].append(date)
            data['RR'].append(row['RR'])
            # print(row['Tanggal']+' Adalah
tanggal')
        except:
            continue
            # print(str(row['Tanggal'])+' Bukan
Tanggal')
    return data
```

Gambar 4.3 *Source Code* Proses Transform Raw Dataset

Gambar 4.4 menjelaskan bahwa kode tersebut untuk menampilkan data dalam bentuk tabel dengan kolom tanggal dan RR. *DataFrame* merupakan tabel atau data tabular dengan array dua dimensi yaitu baris dan kolom.

```
all_data = pd.DataFrame({'Tanggal':list,
'RR':list()})
for index, row in df_path.iterrows():
    raw_data = pd.read_excel(row['Path'],
header=8)
    data = process_data_raw(raw_data)
    df_data = pd.DataFrame(data)
    all_data = pd.concat([all_data,df_data])
```

Gambar 4.4 *Source Code* Proses Pemanggilan Data Raw

Pada *source code* program Gambar 4.5 menjelaskan bahwa kode tersebut untuk melakukan proses pemanggilan data secara keseluruhan.

```
all_data = all_data.reset_index(drop=True)
all_data['RR'] = all_data['RR'].fillna(0)
all_data.to_excel('Hasil.xlsx', index=False)
```

Gambar 4.5 *Source Code* Proses Pemanggilan Data Raw

Gambar 4.6 merupakan hasil dari transformasi dataset, dataset yang digunakan mulai dari tahun 2017 sampai dengan tahun 2021, menjelaskan bahwa tampilam data raw curah hujan berdasarkan tanggal.

Tanggal	RR
2017-02-01	1.6
2017-02-02	6.0
2017-02-03	2.0
2017-02-04	1.7
2017-02-05	2.1
...	...
2021-01-27	10.6
2021-01-28	2.9
2021-01-29	3.6
2021-01-30	12.4
2021-01-31	0.3

1826 rows × 1 columns

Gambar 4.6 Data Raw curah hujan harian

4.4 Prediksi Curah Hujan

Pengujian dilakukan menggunakan data dari BMKG Kota Bandung yang diperoleh dari website BMKG (http://dataonline.bmkg.go.id/dashboard_user) untuk melihat curah hujan Kota Bandung dan untuk melihat nilai kesalahan dari peramalan digunakan. Gambar 4.7 merupakan *source code* untuk mengimport *library* yang digunakan.

```
import pandas as pd
import numpy as np
from datetime import datetime
from matplotlib import pyplot
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
import tensorflow as tf

from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
```

Gambar 4.7 *Source Code Import Library*.

Fungsi-fungsi dari library di atas yaitu *pandas* untuk mengolah *dataframe* seperti menambah atau menghapus data. *Numpy* berfungsi untuk memproses sebuah data berbentuk array. *Datetime* berfungsi untuk mengoperasikan sesuatu yang berhubungan dengan waktu. *Matplotlib* berfungsi untuk memvisualisasikan data dalam bentuk grafik. *MinMaxScaler* berfungsi untuk menormalisasi data. *Tensorflow* berfungsi untuk *machine learning*, *deep learning*, dsb.

```
# convert an array of values into a dataset matrix
def create_dataset(dataset, look_back=1):
    dataX, dataY = [], []
    for i in range(len(dataset)-look_back-1):
        a = dataset[i:(i+look_back), 0]
        dataX.append(a)
        dataY.append(dataset[i + look_back, 0])
    return np.array(dataX), np.array(dataY)
# fix random seed for reproducibility
tf.random.set_seed(10)
```

Gambar 4.8 Mengubah array ke matrix

Pada *source code* program Gambar 4.8 menjelaskan bahwa kode tersebut untuk melakukan proses mengubah data array ke dalam matrix sesuai dataset yang digunakan. `def create_dataset` berfungsi untuk membuat dataset.

```
data = pd.read_excel('Hasil.xlsx',
                     index_col ='Tanggal',
                     parse_dates = True)
data
```

Gambar 4.9 Proses Membaca Data Excel

Pada *source code* program Gambar 4.9 menjelaskan bahwa kode tersebut untuk melakukan proses membaca dataset yang formatnya excel.

	RR
Tanggal	
2017-02-01	1.6
2017-02-02	6.0
2017-02-03	2.0
2017-02-04	1.7
2017-02-05	2.1
...	...
2021-01-27	10.6
2021-01-28	2.9
2021-01-29	3.6
2021-01-30	12.4
2021-01-31	0.3

1826 rows × 1 columns

Gambar 4.10 Data curah hujan harian

Gambar 4.10 merupakan hasil dari transformasi dataset, dataset yang digunakan mulai dari tahun 2017 sampai dengan tahun 2021, sebagai berikut

```
data_value = {'satuan':0, 'puluhan':0, 'ratusan':0}
for d in data.values:
    if d <10:
        data_value['satuan']+=1
    elif d>=10 and d<100:
        data_value['puluhan']+=1
    elif d>=100 and d<1000:
        data_value['ratusan']+=1
print(data_value)
```

Gambar 4.11 Prososе Pemanggilan Data Value

Pada *source code* program Gambar 4.11 menjelaskan bahwa kode tersebut untuk melakukan proses pemanggilan data value dengan value satuan, puluhan, dan ratusan.

```
{'satuan': 1463, 'puluhan': 361, 'ratusan': 2}
```

Gambar 4.12 Hasil dari Pemanggilan Data Value

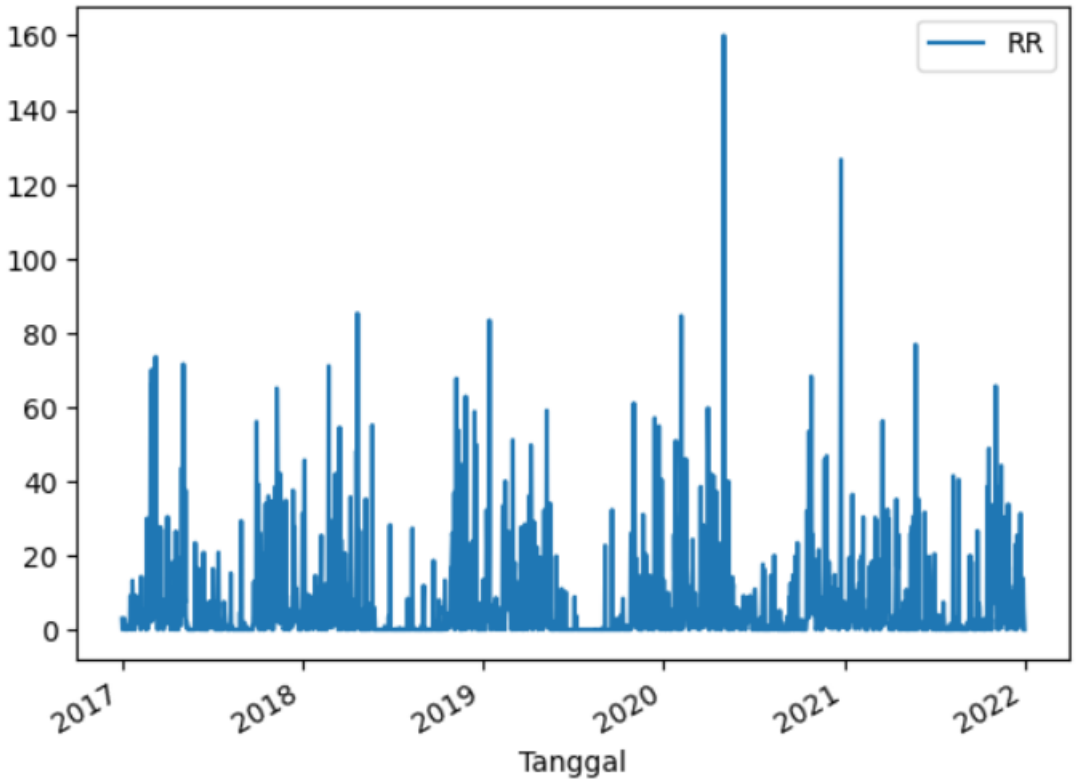
Pada *source code* program Gambar 4.12 menampilkan tampilan data value satuan, puluhan, ratusan.

Kemudian menampilkan data berupa grafik plot, pada *source code* program Gambar 4.13 menjelaskan bahwa kode tersebut untuk melakukan proses menampilkan data diagram yang jenisnya line plot.

```
# line plot
data.plot()
pyplot.show()
```

Gambar 4.13 Proses Menampilkan Data Diagram

Berikut tampilan data berupa grafik plot, hasil dari *souce code* gambar 4.14

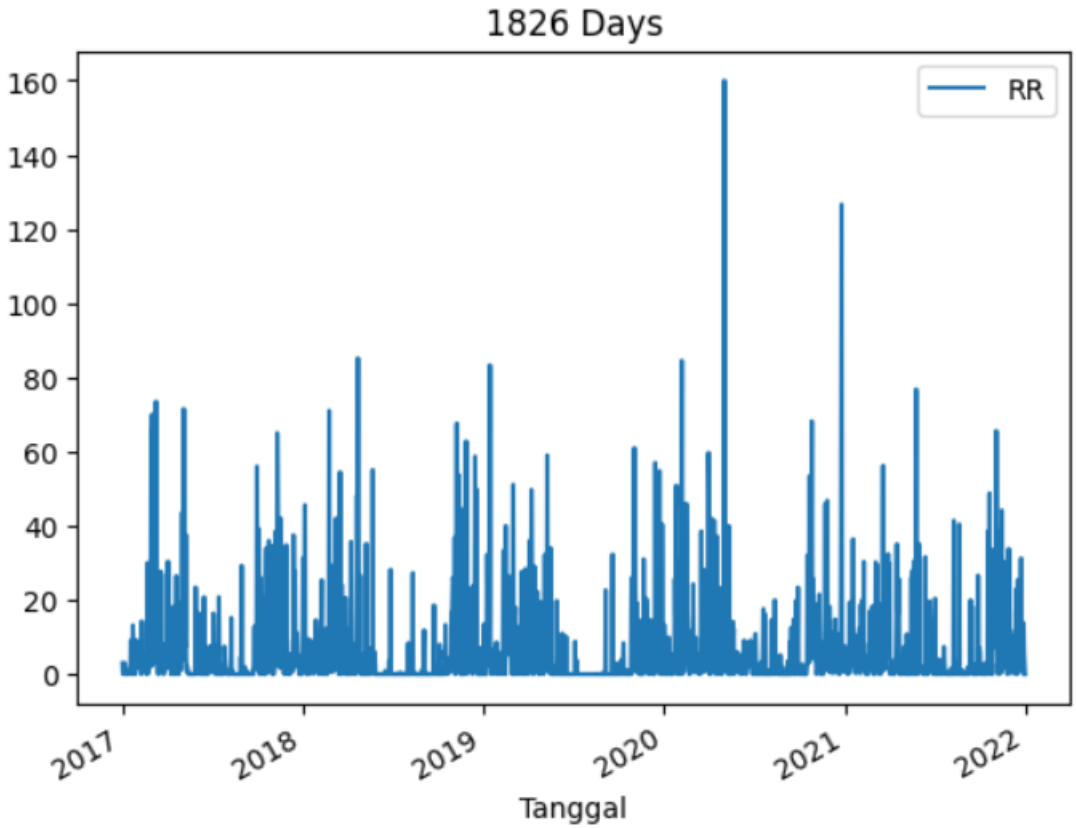


Gambar 4.14 Tampilan Grafik Plot

```
# Get first n days
n = 1826
d_n = data[:n]
d_n.plot()

pyplot.title(str(n)+' Days')
```

Gambar 4.15 Proses Tampilan Data Per Hari

Pada *source code* program Gambar 4.15 menjelaskan bahwa kode tersebut untuk proses tampilan data perhari, yang mana banyaknya data sebanyak 1826. Pada grafik, curah hujan tertinggi terjadi pada tahun 2020 sebesar 160. Berikut tampilan data berupa grafik plot Data Perhari



Gambar 4.16 Hasil Tampilan Data Per Hari

```
dataset = data.values
dataset = dataset.astype('float32')
dataset
```

Gambar 4.17 Proses Dataset Array

Pada *source code* program Gambar 4.17 menjelaskan bahwa kode tersebut untuk proses dataset array, dengan menggunakan type float.

Hasil dari proses dataset values menjadi array, dapat dilihat pada Gambar 4.18, sebagai berikut:

```
array([[  1.6],
       [  6. ],
       [200. ],
       ...,
       [  3.6],
       [ 12.4],
       [  0.3]], dtype=float32)
```

Gambar 4.18 Hasil Dataset Array

Setelah *library* diimpor, tahap selanjutnya adalah normalisasi pada data curah hujan harian, *scaler = MinMaxScaler(feature_range=(0, 1))* mengubah fitur dengan menskalakan setiap fitur ke rentang tertentu pada set pelatihan, misalnya antara nol dan satu. Gambar 4.19 merupakan *source code* yang digunakan.

```
# normalize the dataset
scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(dataset)
```

Gambar 4.19 *Source code* normalisasi

Setelah dilakukan normaliasi maka selanjutnya adalah membagi dataset menjadi data pelatihan dan data uji. Jika dilihat dari *source code* data yang digunakan untuk model pelatihan adalah 90% dan sisanya 10% digunakan debagai data uji. Gambar 4.20 merupakan *source code* yang digunakan:

```
# split into train and test sets
train_size = int(len(dataset) * 0.9)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size,:], dataset[trai
n_size:len(dataset),:]
```

Gambar 4.20 *Source Code* Split Data

Gambar 4.21 merupakan *source code* yang berfungsi membentuk rangkaian data sebelum menerapkannya ke dalam model LSTM

```
# reshape into X=t and Y=t+1
look_back = 3
trainX, trainY = create_dataset(train, look_back)
testX, testY = create_dataset(test, look_back)
```

Gambar 4.21 *Source Code* Persiapan Data

Gambar 4.22 merupakan merupakan proses penyesuaian data pada model LSTM yang dapat dilihat pada *source code* berikut:

```
# create and fit the LSTM network
batch_size = 1
model = Sequential()
model.add(LSTM(6, batch_input_shape=(batch_size, look_back, 1),
stateful=True, return_sequences=True))
model.add(LSTM(6, batch_input_shape=(batch_size, look_back, 1),
stateful=True))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
```

Gambar 4.22 Proses Membuat dan Memasang LSTM

Gambar 4.23 merupakan merupakan proses pengujian epoch, di mana epoch yang digunakan itu 75, yang dapat dilihat pada *source code* berikut

```
model.fit(trainX, trainY, epochs=75, batch_size=batch_size, s
                        huffle=False)
```

Gambar 4.23 *Source Code* Pengujian Epoch

Source code diatas berfungsi untuk menguji epoch, dengan nilai trainX dan trainY dengan nilai epoch sebesar 75. Di bawah ini menampilkan hasil pengujian epoch yang dapat dilihat pada gambar di bawah. Proses ini menggambarkan proses pengujian epoch dari data yang telah dinormalisasi. Gambar 4.24 merupakan hasil dari pengujian *epoch*

```
1639/1639 [==============================] - 5s 3ms/step - loss: 0.0062
Epoch 4/75
1639/1639 [==============================] - 5s 3ms/step - loss: 0.0062
Epoch 5/75
1639/1639 [==============================] - 4s 3ms/step - loss: 0.0061
Epoch 6/75
1639/1639 [==============================] - 5s 3ms/step - loss: 0.0061
Epoch 7/75
1639/1639 [==============================] - 5s 3ms/step - loss: 0.0061
Epoch 8/75
1639/1639 [==============================] - 5s 3ms/step - loss: 0.0061
Epoch 9/75
1639/1639 [==============================] - 5s 3ms/step - loss: 0.0061
Epoch 10/75
1639/1639 [==============================] - 5s 3ms/step - loss: 0.0061
Epoch 11/75
1639/1639 [==============================] - 6s 3ms/step - loss: 0.0060
Epoch 12/75
1639/1639 [==============================] - 5s 3ms/step - loss: 0.0060
Epoch 13/75
1639/1639 [==============================] - 5s 3ms/step - loss: 0.0060
Epoch 14/75
1639/1639 [==============================] - 5s 3ms/step - loss: 0.0060
Epoch 15/75
1639/1639 [==============================] - 5s 3ms/step - loss: 0.0060
Epoch 16/75
1639/1639 [==============================] - 5s 3ms/step - loss: 0.0060
Epoch 17/75
1639/1639 [==============================] - 5s 3ms/step - loss: 0.0060
Epoch 18/75
1639/1639 [==============================] - 5s 3ms/step - loss: 0.0060
Epoch 19/75
1639/1639 [==============================] - 5s 3ms/step - loss: 0.0060
Epoch 20/75
```

Gambar 4.24 Pengujian Epoch

Kemudian dilakukan prediksi data train dan data test yang dapat dilihat pada gambar 4.25 dan gambar 4.26

```
# make predictions
trainPredict = model.predict(trainX, batch_size=batch_size)
model.reset_states()
testPredict = model.predict(testX, batch_size=batch_size)
```

Gambar 4.25 *Source Code* melakukan prediksi

Pada *source code* program Gambar 4.25 menjelaskan bahwa tampilan prediksi data train dan data testing. Tampilannya bisa dilihat pada Gambar 4.26.

```
1639/1639 [==============================] - 3s 1ms/step
179/179 [==============================] - 0s 1ms/step
```

Gambar 4.26 Prediksi data train dan data test

Kemudian dilakukan pembalikan prediksi, hal ini untuk memastikan bahwa model yang dihasilkan sama dengan data asli. *Source code* dapat dilihat pada gambar 4.27, sebagai berikut:

```
# invert predictions
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform([trainY])
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform([testY])
```

Gambar 4.27 Proses Pembalikan Prediksi

Kemudian dilakukan evaluasi yang bertujuan mengukur kerja dari model yang telah dibuat. Metode evaluasi yang digunakan adalah RMSE. Metode ini dilakukan pengujian terhadap dua model yaitu data train dan data test. Berikut merupakan kode program yang dapat dilihat pada gambar 4.28, sebagai berikut:

```
# calculate root mean squared error
trainScore = np.sqrt(mean_squared_error(trainY[0],
trainPredict[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = np.sqrt(mean_squared_error(testY[0],
testPredict[:,0]))
print('Test Score: %.2f RMSE' % (testScore))
```

Gambar 4.28 Proses Menghitung RMSE

Hasil dari *source code* Gambar 4.28 berupa persentase kesalahan, di mana score data training yaitu 12.24 sedangkan testing score yaitu 8.86, dapat dilihat pada Gambar 4.29, sebagai berikut:

```
Train Score: 12.17 RMSE
Test Score: 8.77 RMSE
```

Gambar 4.29 Hasil Menghitung Kesalahan

Dari hasil di atas menunjukkan skor dari data testing yaitu 8.86. Semakin kecil skor data testing, maka semakin baik akurasi dari prediksi yang dihasilkan. Apabila skor data testing besar, maka

hasil tersebut masih tergolong kurang baik. Selanjutnya membuat data hasil prediksi data training berupa grafik plotting, berikut *source code* nya.

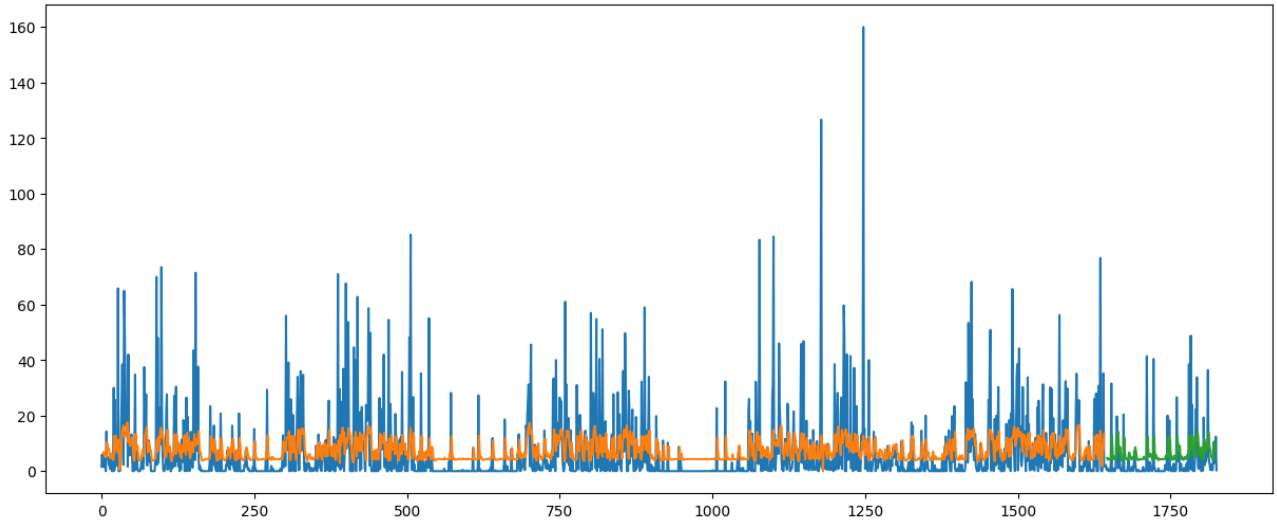
```
# shift train predictions for plotting
trainPredictPlot = np.empty_like(dataset)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[look_back:len(trainPredict)+look_back, :] =
trainPredict

# shift test predictions for plotting
testPredictPlot = np.empty_like(dataset)
testPredictPlot[:, :] = np.nan
testPredictPlot[len(trainPredict)+(look_back*2)+1:len(dataset
)-1, :] = testPredict
```

Gambar 4.30 Membentuk grafik plotting hasil prediksi

```
# plot baseline and predictions
plt.figure(figsize=(15,6))
plt.plot(scaler.inverse_transform(dataset))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()
```

Gambar 4.31 Membentuk grafik plotting hasil prediksi (lanjutan)

Hasil dari *source code* gambar 4.30 dan 4.31 merupakan tampilan grafik setelah data diplot, menunjukkan set data asli dengan warna biru, prediksi untuk set data pelatihan berwarna orange, dan prediksi pada set data uji dengan warna hijau. Berikut merupakan visualisasi prediksi yang dapat dilihat pada gambar 4.32.



Gambar 4. 32 Visualisasi Prediksi

Di bawah ini merupakan tabel hasil uji performa yang di mana pada pengujian performa dilihat berdasarkan Root Mean Square Error (RMSE). Pada uji performa ini, epoch yang menjadi acuan utama untuk menghasilkan nilai RMSE Data Test terendah atau performa terbaik.

Tabel 4.1 Hasil Uji Performa

Epoch	Data Train	Data Test	RMSE Data Train	RMSE Data Test
25	1639	179	12.33	8.90
50	1639	179	12.24	8.86
75	1639	179	12.21	8.78
100	1639	179	12.15	8.84
150	1639	179	12.06	9.39
200	1639	179	11.90	9.05

Dari hasil uji performa yang terdapat pada tabel di atas, performa terendah didapatkan dengan nilai epoch 150 yang berhasil memperoleh nilai RMSE Data Train sebesar 12.06 dan RMSE Data Tes sebesar 9.39. Sedangkan performa tertinggi didapatkan dengan nilai epoch 75 yang berhasil memperoleh nilai RMSE Data Train sebesar 12.21 dan RMSE Data Test 8.78.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah melakukan analisis, perancangan dan implementasi serta hasil pengujian dari implementasi deep learning menggunakan arsitektur *Long Short Term Memory* (LSTM) untuk prediksi curah hujan harian di Kota Bandung, menunjukkan hasil prediksi yang diperoleh tergolong cukup baik. Berikut pembahasan tentang penelitian yang telah dilakukan

1. Penelitian ini berhasil melakukan prediksi terhadap curah hujan di Kota Bandung dengan menggunakan arsitektur LSTM. Dalam melakukan prediksi curah hujan harian di Kota Bandung dilakukan pengumpulan data, preprocessing, normalisasi, dan evaluasi dengan menggunakan RMSE.
2. Penelitian ini menghasilkan nilai akurasi tertinggi yaitu dengan Train Score RMSE sebesar 12.21 dan Test Score RMSE sebesar 8.78 pada prediksi curah hujan harian Kota Bandung.

5.2 Saran

Peneliti berharap agar penelitian ini dapat dikembangkan oleh peneliti lain di masa yang akan datang. Saran yang dapat diberikan oleh peneliti antara lain sebagai berikut:

1. Peneliti perlu menambah beberapa parameter/metode lain yang dapat membuat penelitian semakin lebih baik dapat membandingkan mana metode yang lebih baik.
2. Dapat menambah data yang digunakan untuk mendapatkan nilai prediksi yang semakin baik lagi.

DAFTAR PUSTAKA

Aldi, M. W. P., Jondri, & Aditsania, A. (2018). Analisis dan Implementasi Long Short Term Memory Neural Network untuk Prediksi Harga Bitcoin. *Jurnal Informatika*, *5*, *No*(2), 3548. http://openlibrarypublications.telkomniversity.ac.id

Andini, T. D., & Auristandi, P. (2016). Peramalan Jumlah Stok Alat Tulis Kantor di UD Achmad Jaya Menggunakan Metode Double Exponential Smoothing. *Jurnal Ilmiah Teknologi Informasi Asia*, *10*(1), 1–10.

Ashari, M. L., & Sadikin, M. (2020). Prediksi Data Transaksi Penjualan Time Series Menggunakan Regresi Lstm. *Jurnal Nasional Pendidikan Teknik Informatika (JANAPATI)*, *9*(1), 1. https://doi.org/10.23887/janapati.v9i1.19140

Badriyah, J., Fariza, A., & Harsono, T. (2022). *JURNAL MEDIA INFORMATIKA BUDIDARMA Prediksi Curah Hujan Menggunakan Long Short Term Memory*. *6*, 1297–1303. https://doi.org/10.30865/mib.v6i3.4008

Budi, R. S., Patmasari, R., & Saidah, S. (2021). *Klasifikasi Cuaca Menggunakan Metode Convolutional Neural Network (CNN)*. *8*(5), 5047–5052.

colah. (2015). *Retrieved from Colahsblog*. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Darmawan, G., Pontoh, R. S., Tantular, B., Padmadisastra, S., Handoko, B., & F., Y. K. (2016). PERAMALAN PADA DATA IRREGULAR SINUSOIDAL DENGAN MENGGUNAKAN MODEL HOLT-WINTERS. *Paper Knowledge . Toward a Media History of Documents*, *3*(April), 49–58.

Freecenta, H. F., Yulia Puspaningrum, E., & Maulan, H. (2022). Prediksi Curah Hujan Di Kab. Malang Menggunakan LSTM (Long Short Term Memory). *Jurnal Informatika Dan Sistem Informasi*, *3*(1), 51–55. https://doi.org/10.33005/jifosi.v3i1.448

Habibi, M. Y., & Riksakomara, E. (2017). Peramalan Harga Garam Konsumsi Menggunakan Artificial Neural Network Feedforward-Backpropagation (Studi Kasus : PT. Garam Mas, Rembang, Jawa Tengah). *Jurnal Teknik ITS*, *6*(2). https://doi.org/10.12962/j23373539.v6i2.23200

Juanda, R. A., Jondri, & Rohmawati, A. A. (2018). Prediksi Harga Bitcoin Dengan

Menggunakan Recurrent Neural Network. *EProceedings of Engineering*, *5*(2), 3682–3690.

Karim, F., Majumdar, S., Darabi, H., & Harford, S. (2019). Multivariate LSTM-FCNs for time series classification. *Neural Networks*, *116*, 237–245. https://doi.org/https://doi.org/10.1016/j.neunet.2019.04.014

Khumaidi, A., Raafi'udin, R., & Solihin, I. P. (2020). Pengujian Algoritma Long Short Term Memory untuk Prediksi Kualitas Udara dan Suhu Kota Bandung. *Jurnal Telematika*, *15*(1), 13–18. https://journal.ithb.ac.id/telematika/article/view/340

Kurniawansyah, A. S. (2018). Implementasi Metode Artificial Neural Network Dalam Memprediksi Hasil Ujian Kompetensi Kebidanan (Studi Kasus Di Akademi Kebidanan Dehasen Bengkulu). *Pseudocode*, *5*(1), 37–44. https://doi.org/10.33369/pseudocode.5.1.37-44

Lattifia, T., Wira Buana, P., & Rusjayanthi, N. K. D. (2022). Model Prediksi Cuaca Menggunakan Metode LSTM. *JITTER Jurnal Ilmiah Teknologi Dan Komputer*, *3*(1), 994–1000. https://ojs.unud.ac.id/index.php/jitter/article/view/85000/43781

Lipton, Z. C., Kale, D. C., Elkan, C. P., & Wetzel, R. C. (2016). Learning to Diagnose with LSTM Recurrent Neural Networks. *CoRR*, *abs/1511.0*.

Ma, X., Tao, Z., Wang, Y., Yu, H., & Wang, Y. (2015). Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, *54*, 187–197. https://doi.org/https://doi.org/10.1016/j.trc.2015.03.014

Noerhayati, E. N., Suprapto, B., & Syahid, A. A. (2017). Peningkatan Keuntungan Melalui Optimasi Sistem Pemberian Air Daerah Irigasi Molek Dengan Program Linier. *Jurnal Teknika*, *9*(1), 13. https://doi.org/10.30736/teknika.v9i1.6

Panggabean, D. A. H., Sihombing, F. M., & Aruan, N. M. (2021). Prediksi Tinggi Curah Hujan Dan Kecepatan Angin Berdasarkan Data Cuaca Dengan Penerapan Algoritma Artificial Neural Network (Ann). *Seminastika*, *3*(1), 1–7. https://doi.org/10.47002/seminastika.v3i1.237

Prathama, A. Y., Aminullah, A., & Saputra, A. (2018). Pendekatan Ann (Artificial Neural Network) Untuk Penentuan Prosentase Bobot Pekerjaan Dan Estimasi Nilai Pekerjaan Struktur Pada Rumah Sakit Pratama. *Jurnal Teknosains*, *7*(1), 14. https://doi.org/10.22146/teknosains.30139

Qiu, J., Wang, B., & Zhou, C. (2020). Forecasting stock prices with long-short term memory neural network based on attention mechanism. *PLoS ONE*, *15*(1), 1–15. https://doi.org/10.1371/journal.pone.0227222

Rizki, M., Basuki, S., & Azhar, Y. (2020). Implementasi Deep Learning Menggunakan Arsitektur Long Short Term Memory(LSTM) Untuk Prediksi Curah Hujan Kota Malang. *Jurnal Repositor*, *2*(3), 331. https://doi.org/10.22219/repositor.v2i3.470

Soekendro, C. A. (2021). *Prediksi Curah Hujan di Kab. Bandung Dengan Analisis Time Series, Menggunakan Model SARIMA (Seasonal Autoregressive Integrated Moving Average)*. *8*(2), 2865–2875.

Suhartono, D. (2012). *Retrieved from Binus Uversity School of Computer*. https://socs.binus.ac.id/2012/07/26/konsep-neural-network/

Tian, C., Ma, J., Zhang, C., & Zhan, P. (2018). A deep neural network model for short-term load forecast based on long short-term memory network and convolutional neural network. *Energies*, *11*(12). https://doi.org/10.3390/en11123493

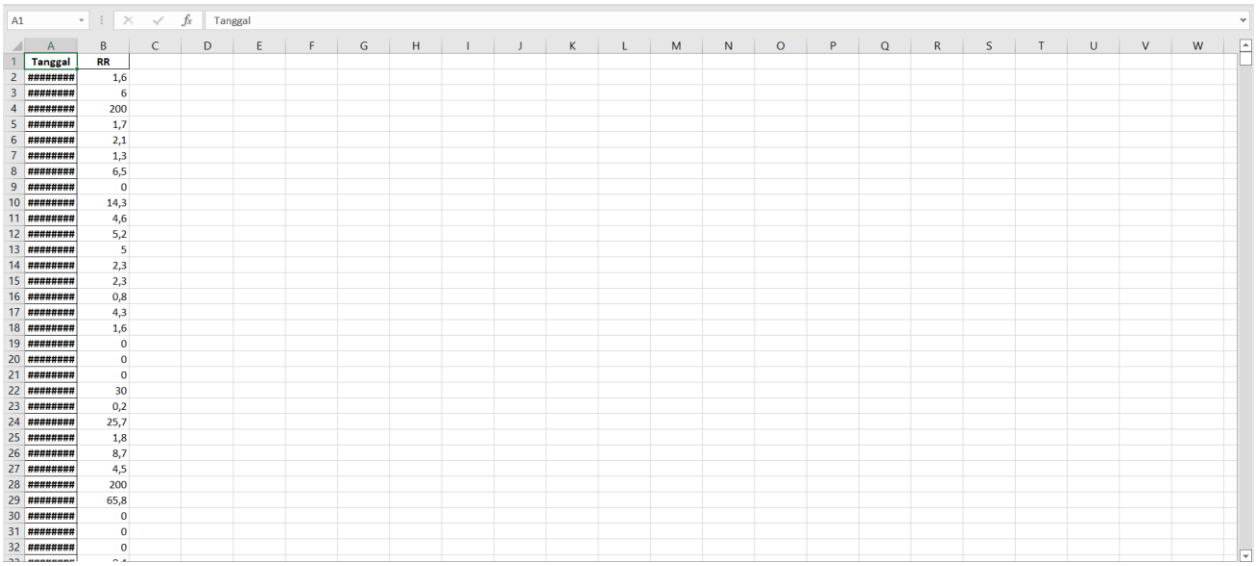
Wiranda, L., & Sadikin, M. (2019). Penerapan Long Short Term Memory Pada Data Time Series Untuk Memprediksi Penjualan Produk Pt. Metiska Farma. *Jurnal Nasional Pendidikan Teknik Informatika : JANAPATI*, *8*(3), 184–196. https://ejournal.undiksha.ac.id/index.php/janapati/article/view/19139

Yunanto, R., Purfini, A. P., & Prabuwisesa, A. (2021). Survei Literatur: Deteksi Berita Palsu Menggunakan Pendekatan Deep Learning. *Jurnal Manajemen Informatika (JAMIKA)*, *11*(2), 118–130. https://doi.org/10.34010/jamika.v11i2.5362

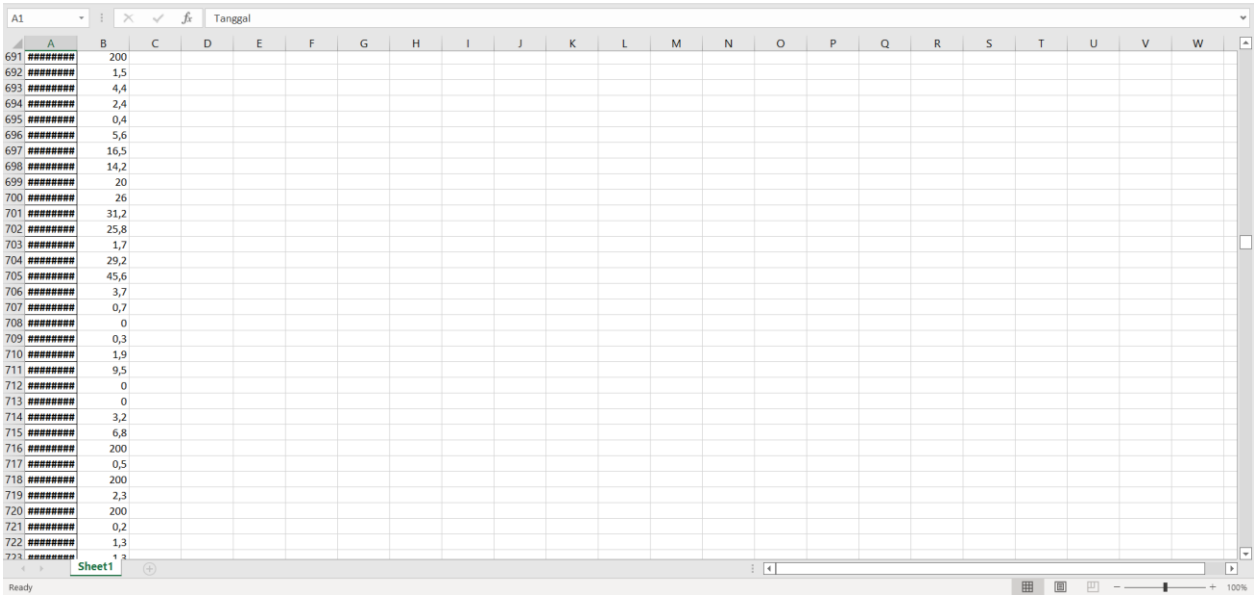
Zahara, S., Sugianto, & Ilmiddafiq, M. B. (2019). Prediksi Indeks Harga Konsumen Menggunakan Metode Long Short Term Memory (LSTM) Berbasis Cloud Computing. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *3*(3), 357–363. https://doi.org/10.29207/resti.v3i3.1086

LAMPIRAN

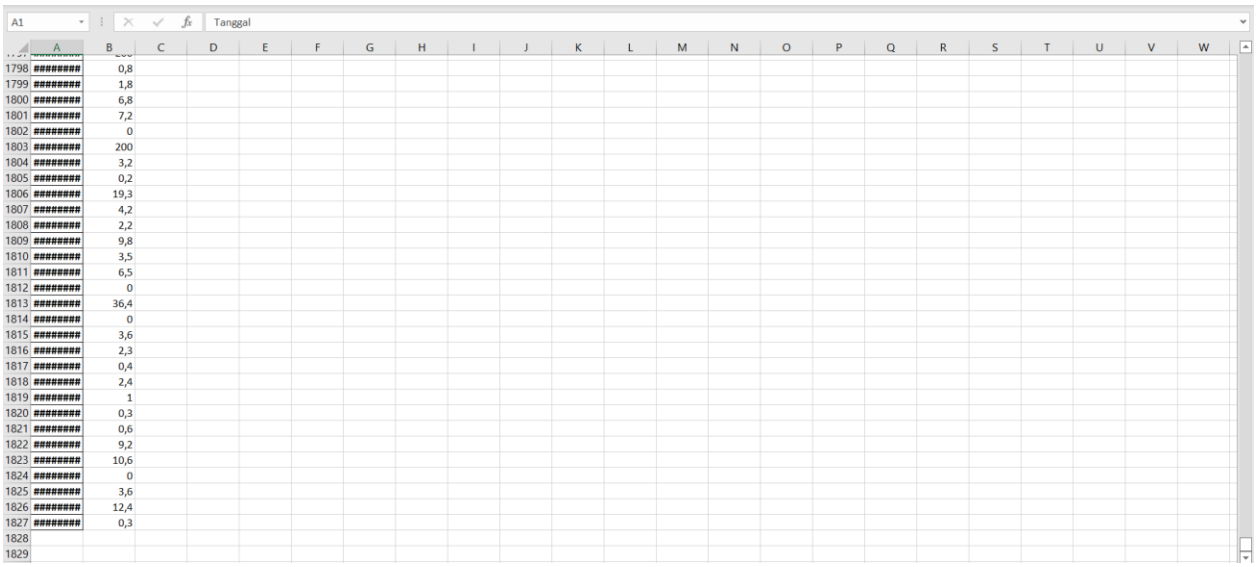
Lampiran 1 Visualisasi Tabel Data



	Tanggal	RR
2	########	1,6
3	########	6
4	########	200
5	########	1,7
6	########	2,1
7	########	1,3
8	########	6,5
9	########	0
10	########	14,3
11	########	4,6
12	########	5,2
13	########	5
14	########	2,3
15	########	2,3
16	########	0,8
17	########	4,3
18	########	1,6
19	########	0
20	########	0
21	########	0
22	########	30
23	########	0,2
24	########	25,7
25	########	1,8
26	########	8,7
27	########	4,5
28	########	200
29	########	65,8
30	########	0
31	########	0
32	########	0



691	########	200
692	########	1,5
693	########	4,4
694	########	2,4
695	########	0,4
696	########	5,6
697	########	16,5
698	########	14,2
699	########	20
700	########	26
701	########	31,2
702	########	25,8
703	########	1,7
704	########	29,2
705	########	45,6
706	########	3,7
707	########	0,7
708	########	0
709	########	0,3
710	########	1,9
711	########	9,5
712	########	0
713	########	0
714	########	3,2
715	########	6,8
716	########	200
717	########	0,5
718	########	200
719	########	2,3
720	########	200
721	########	0,2
722	########	1,3



	A	B
1798	########	0,8
1799	########	1,8
1800	########	6,8
1801	########	7,2
1802	########	0
1803	########	200
1804	########	3,2
1805	########	0,2
1806	########	19,3
1807	########	4,2
1808	########	2,2
1809	########	9,8
1810	########	3,5
1811	########	6,5
1812	########	0
1813	########	36,4
1814	########	0
1815	########	3,6
1816	########	2,3
1817	########	0,4
1818	########	2,4
1819	########	1
1820	########	0,3
1821	########	0,6
1822	########	9,2
1823	########	10,6
1824	########	0
1825	########	3,6
1826	########	12,4
1827	########	0,3
1828		
1829		