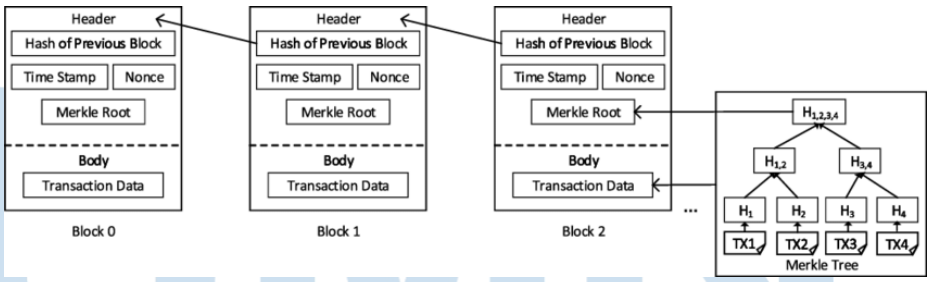
BAB II

LANDASAN TEORI

2.1 Teori yang digunakan

2.1.1 *Blockchain*

Blockchain adalah sebuah teknologi *distributed ledger* yang menyimpan sebuah data. Teknologi *blockchain* memberikan integritas dan ketersediaan yang memungkinkan para partisipan dalam sebuah jaringan *blockchain* untuk melakukan menulis, membaca, dan memverifikasi sebuah transaksi yang tercatat pada *blockchain* [11]. Teknologi *blockchain* sesuai dengan namanya, terdiri dari rangkaian *block* yang saling terhubung dengan satu sama lain dan di dalamnya tersimpan informasi atau data terkait sebuah transaksi. Teknologi *blockchain* biasanya sering dikaitkan dengan *cryptocurrency* , teknologi *blockchain* digunakan dalam jaringan *cryptocurrency* dikarenakan data transaksi yang disimpan dalam *distributed ledger* dan divalidasi serta dikelola oleh jaringan komputer diseluruh dunia [12].



Gambar 2.1 Struktur dari sebuah *block* pada *blockchain* [13]

Setiap *block* dalam terdapat beberapa informasi seperti *Hash* dari *block* itu sendiri, *Nonce*, *Timestamp* , dan *Hash* dari *block* sebelumnya. *Nonce* adalah singkatan dari "*Number only used once*" yang merupakan angka acak 32 bit yang digunakan untuk proses otentikasi , *hashing* dan identifikasi. *Timestamp* merupakan sebuah nilai waktu yang digunakan untuk menentukan urutan posisi *block* pada sebuah *blockchain*. Hash adalah sebuah fungsi *cryptography* yang digunakan untuk melakukan

verifikasi integritas data dan memastikan keamanan jaringan [2]. Teknologi *blockchain* dalam sebuah jaringan *cryptocurrency* didorong dengan sifat dari *blockchain* itu sendiri yaitu *Decentralize, Immutable* dan *Transparent*. *Decentralize* memiliki arti bahwa data pada jaringan *blockchain* tidak disimpan pada satu tempat, melainkan seluruh partisipan pada sebuah *blockchain* memiliki salinan data yang sama. *Immutable* dalam teknologi *blockchain* berarti data yang sudah diinput ditulis dalam sebuah *block* tidak dapat diubah karena pengubahan data dalam sebuah *block* akan mengubah *hash addresses* dari *block* tersebut, hal ini membuat data dalam jaringan *blockchain* sangat sulit atau bahkan hampir tidak mungkin untuk diubah. *Transparent,* transparansi dalam teknologi *blockchain* memiliki arti semua transaksi yang terjadi dalam jaringan *blockchain* dapat dilihat dan divalidasi oleh partisipan dari jaringan *blockchain* itu sendiri.

Dalam penelitian ini, *blockchain* dijelaskan sebagai pemberi konteks dalam mempelajari *cryptocurrency*, namun tidak melibatkan pembuatan atau implementasi teknologi *blockchain*. Pemahaman tentang konsep dasar *blockchain* memberikan landasan penting dalam menganalisis *cryptocurrency*. Penelitian ini berfokus pada analisis *cryptocurrency*, terutama dalam mengidentifikasi tren harga di pasar *cryptocurrency*.

2.1.2 Cryptocurrency

Cryptocurrency adalah sebuah nama dari sistem yang menggunakkan kriptografi, dimana kata *cryptocurrency* itu sendiri adalah sebuah gabungan dari 2 kata, yaitu "*Cryptography*" yang memiliki arti kode rahasia, dan "*currency*" yang memiliki arti mata uang [14]. Keamanan dari *cryptocurrency* disebabkan oleh penggunaan teknologi *blockchain* yang digunakan sebagai tulang punggung dari sebuah jaringan *Cryptocurrency*.

2.1.3 Bitcoin dan Altcoin

Bitcoin (BTC) sering dikenal sebagai *digital currency* atau *Cryptocurrency* pertama yang menggunakan konsep kriptografi milik teknologi *blockchain* sebagai sistem keamanannya [15]. Bitcoin sendiri ciptakan oleh seseorang yang bernama Satoshi Nakamoto pada tahun 2008. Satoshi Nakamoto menerbitkan sebuah artikel yang berjudul "*Bitcoin A peer-to-peer Electronic Cash System*", dalam artikel ini dia menjelaskan tentang kelemahan dari sistem pembayaran elektronik yang dilakukan oleh lembaga keuangan sebagai pihak ketiga dengan menggunakan sistem *peer-to-peer network* dengan *proof-of-work* untuk merekam sebuah transaksi, tanpa memerlukan keterlibatan dari pihak ketiga.

Ethereum (ETH) adalah *distributed ledger open-source* berbasis *blockchain* yang memiliki *smart contract*. Konsep dari Ethereum pertama kali dikenalkan oleh Vitalik Buterin lewat *White Paper* dengan judul "*Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform*" pada tahun 2014 yang kemudian *projec*t Ethereum melakukan *launching* pada 2015. Ether adalah sebuah *Cryptocurrency* asli yang digunakan dalam Ethereum *network*, Ether juga digunakan untuk memberikan kompensasi kepada penambang. Ether digunakan dalam Ethereum *network* untuk berbagai macam transaksi, seperti digunakan untuk membayar "*gas fee*" atau *transaction fee*, Ether dapat digunakan sebagai jaminan pada beberapa aplikasi *open market* seperti "MakerDao" dan "Compound", dan yang paling *trending* pada saat ini Ether digunakan untuk melakukan transaksi NFT (*Non-Fungible tokens*), yang dimana pada saat ini Ethereum *network* adalah jaringan yang paling banyak digunakan dalam hal NFT.

Dogecoin (DOGE) adalah sebuah *cryptocurrency* terdesentralisasi berdasarkan sebuah meme anjing ras Shiba Inu. Dogecoin diciptakan oleh seorang bernama Billy Markus dan Jackson Palmer pada tahun 2013, mereka menciptakan Dogecoin candaan sebagai respon spekulasi-

spekulasi mengenai *Cryptocurrency* pada waktu itu, oleh karena itu Dogecoin disebut sebagai *Cryptocurrency* meme pertama. Dogecoin dipromosikan sebagai "*The People's Currency*" yang artinya mata uang rakyat dikarenakan komunitas dari Dogecoin yang sangat besar.

2.1.4 *Machine Learning*

Machine Learning adalah sebuah aplikasi komputer dan algoritma matematika yang diadopsi dengan cara pembelajaran atau *learning* yang dilakukan dengan menggunakan data dan hasil prediksi di masa yang akan datang. Algoritma *Machine Learning* adalah sebuah algoritma yang digunakan untuk melakukan *training model*, dalam *Machine Learning* tipe pembelajaran dibagi menjadi 3 yaitu, *Supervised Learning*, *Unsupervised Learning*, dan *Reinforcement Learning*. *Supervised Learning*, dalam *supervised learning dataset* yang digunakan sudah diberikan label. Kumpulan sampel dari metode ini digunakan untuk meringkas karakteristik distribusi serta perilaku dalam setiap jenis aplikasi sehingga dapat membentuk sebuah model perilaku. *Unsupervised Learning*, dalam *unsupervised learning dataset* yang diberikan tidak diberikan label. *Unsupervised learning* berfungsi untuk menentukan struktur yang tersembunyi dari *dataset* yang tidak berlabel. *Reinforcement Machine Learning* adalah sebuah metode pembelajaran yang tidak memerlukan pengetahuan sebelumnya, karena metode ini melakukan pembelajaran secara mandiri untuk mendapatkan pengetahuan dengan melalui coba-coba atau terus berinteraksi dengan lingkungan yang dinamis [16].

2.1.5 *Deep Learning*

Deep Learning merupakan sebuah cabang dari *machine learning* dimana *deep learning* menggunakan jaringan atau *networks* yang terdiri dari beberapa *layer*. *Layer-layer* yang terdapat dalam *deep learning* ini merupakan kumpulan dari node yang dimana sebuah node berfungsi sebagai tempat perhitungan [17]. Secara umum, *layer* dalam *deep learning* dapat dibagi menjadi 3 bagian, yaitu *input layer, hidden layer* dan *output layer*. *Input layer* adalah tempat dimana data diinput dalam *neural network, hidden layer* adalah tempat dimana di antara *input* dan *output layer* dimana dilakukan transformasi data dan *output layer* adalah

layer yang memberikan hasil dari transformasi data pada *hidden layer* yang ditampilkan sebagai *output* [18].

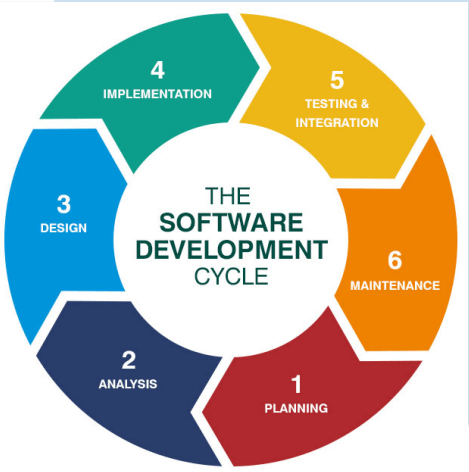
2.1.6 *Multi-Step Forecasting*

Multi-step forecasting atau prediksi beberapa langkah ke depan adalah sebuah proses yang melakukan prediksi untuk beberapa langkah waktu di masa depan dalam sebuah data *time series*. Mudahnya prediksi yang dihasilkan dari *multi-step forecasting* ini adalah beberapa nilai prediksi untuk masa depan tidak seperti *forecasting* pada umumnya yang menghasilkan satu nilai di masa depan [9]. Terdapat beberapa cara untuk melakukan *multi-step forecasting*, diantaranya adalah :

1. *Recursive*: Seperti dengan namanya, *recursive forecasting* dilakukan dengan cara menggunakan sebuah model untuk memprediksi sebuah nilai dan kemudian menggunakan nilai tersebut untuk menjadi *input* ke model selanjutnya. Strategi ini tidak banyak menggunakan komputasi tetapi setiap *error* yang dihasilkan pada setiap iterasi prediksi, akan terakumulasi ke langkah-langkah berikutnya.
2. *Direct*: *Direct forecasting* menggunakan metode *multi-model* untuk melakukan *multi-step forecasting*, dimana untuk melakukan *direct forecasting*, dibutuhkan *n* banyak model untuk memprediksi *n output*. Mudahnya strategi ini menggunakan satu model untuk memprediksi satu *target output*, hal ini membuat *direct forecasting* tidak memiliki masalah seperti *recursive* tetapi membuat kompleksitas *architecture* dan memakan banyak sumber daya komputasi.
3. *MIMO*: Strategi ini adalah menggunakan satu model, untuk membuat banyak prediksi dalam satu kali jalan. Hal ini membuat strategi ini lebih efisien sumber daya komputasi dan juga *error* tidak akan terakumulasi seperti metode *recursive*.

2.1.7 *Software Development Life Cycle*

Software Development Life Cycle biasa disingkat sebagai *SDLC* adalah sebuah metodologi yang sering digunakan dalam sebuah pengembangan sistem informasi [19]. Terdapat beberapa tahapan yang akan dilalui pada *SDLC* yaitu *Planning, Analysis, Design, Implementation, Testing & Integration,* dan *Maintenance.*



Gambar 2.2 Tahapan pada *software development life cycle* [20]

Terdapat beberapa model dari *SDLC* yang sering digunakan untuk melakukan pengembangan sistem informasi, diantaranya adalah *Waterfall, Prototyping,* dan *Agile*.

2.1.8 *User Acceptance Testing*

User Acceptance Testing atau biasa disingkat *UAT* adalah sebuah proses verifikasi yang dilakukan oleh klien atau pengguna akhir (*end user*) yang dilakukan untuk memastikan bahwa solusi dibuat dalam sebuah sistem sudah sesuai untuk digunakan oleh pengguna. Tidak seperti dengan pengujian sistem yang dilakukan untuk memastikan sistem tidak *crash* dan sesuai dengan *requirement* pengguna, *User Acceptance Testing* memastikan bahwa solusi dalam sistem tersebut dapat bekerja untuk pengguna [21]. Pada penelitian ini *User Acceptance Testing* akan dilakukan pada investor *cryptocurrency*.

2.2 *Framework* yang digunakan

2.2.1 CRISP-DM

CRISP-DM atau *Cross-Industry Standard Process for Data Mining* adalah sebuah metodologi yang umum digunakan dalam proses *Data Mining*. Metodologi ini terdiri dari enam tahapan yang saling terkait dan terintegrasi dengan baik. Tahapan-tahapan tersebut meliputi *Business Understanding, Data Understanding, Data Preparation, Data Modeling, Model Evaluation* dan *Deployment* [22].

2.2.2 *Prototyping*

Prototyping adalah sebuah pendekatan yang berdasarkan dari pandangan evolusioner dari pengembangan sistem dan memiliki dampak pada proses pengembangan secara keseluruhan. *Prototyping* melibatkan pembuatan sebuah *prototype* dari sebuah sistem aplikasi untuk bereksperimen [23]. Pembuatan *prototype* digunakan untuk memungkinkan *user* mengevaluasi dan menguji proposal pengembang sebelum diimplementasikan. Ini juga membantu dalam memahami *requirement* khusus *user* yang mungkin tidak dipertimbangkan oleh *developer* selama *design* produk.

2.2.3 Prophet

Prophet adalah prosedur untuk *forecasting* data *time series* berdasarkan *additive model*. Model prophet Ini bekerja paling baik dengan data *time series* yang memiliki efek musiman yang kuat dan beberapa musim pada *historical data*. Prophet kuat terhadap data yang hilang dan perubahan *trend* data, dan biasanya dapat menangani *outlier* dengan baik. Model yang dihasilkan oleh prophet mirip dengan model yang dihasilkan oleh *Generalized Additive Model* (GAM). Prophet dapat mendeteksi perubahan pada *trend* dengan memilih titik perubahan pada data [24].

2.2.4 *Long Short Term Memory* (LSTM)

Long Short Term Memory (LSTM) adalah sebuah model dari varian *Recurrent Neural Network* (RNN) dimana model ini menambahkan *memory cell* untuk menyimpan informasi dalam waktu yang lama [25]. LSTM sendiri diciptakan sebagai solusi dari masalah *vanishing* and *exploding gradient* dan masalah dari *training* dalam jangka waktu yang panjang serta mempertahankan *memory*, maka dari itu LSTM diusulkan untuk meningkatkan algoritma RNN. Struktur dalam dari *hidden layer* milik LSTM lebih kompleks dibandingkan dengan struktur milik RNN.

Struktur *block* pada LSTM dikonfigurasi dengan adanya *cell state memory, forget gate, input gate,* dan *output gate.* Elemen penting dari LSTM yaitu *memory cell* berjalan sepanjang rantai keseluruhan untuk secara selektif menambahkan atau menghapus informasi ke dalam *cell state* dengan bantuan *forget gate, input gate,* dan *output gate* [26]. *Memory cell* inilah yang menjadi komponen utama yang membuat LSTM menjadi jenis RNN yang dapat secara efektif belajar dan mengingat *long-term dependencies* pada *data sequential.*

Setiap *gate* pada LSTM *memory cell* memiliki tugas tersendiri yaitu:

1. *Forget Gate*, mengubah skala *state* internal sebuah *cell* sebelum menambahkannya sebagai masukan ke *cell* melalui koneksi *recurrent* sendiri dari *cell*, sehingga secara adaptif melakukan *forgetting* atau *resetting memory cell*. *Forget Gates* dapat melakukan pembelajaran untuk melakukan pengaturan ulang *memory cell* ketika informasi yang dimiliki sudah tidak diperlukan lagi.
2. *Input Gate*, memiliki tugas untuk mengontrol sinyal-sinyal dari *network* ke *memory cell* dengan mengubah skalanya dengan tepat. Ketika *gate* ini ditutup, *activation* mendekati angka nol.
3. *Output Gate*, akan melakukan pembelajaran untuk mengontrol akses ke konten yang terdapat di *memory cell*, yang akan menjaga *memory cell* dari adanya gangguan.

Prosedur dari sebuah LSTM *network* adalah sebagai berikut :

1. Menentukan apa informasi yang harus dihapus dari *state cell* sebelumnya pada *forget gate* f_t.

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (2.1)\ [26]$$

2. Melakukan identifikasi apakah informasi dari *input* x_t seharusnya disimpan pada *cell state* C_t pada *input gate* dimana *input information* i dan kandidat *cell state* $\tilde{C}_t$ harus diperbarui.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (2.2)\ [26]$$

$$\tilde{C}_t = tanh\ (W_c x_t + U_c h_{t-1} + b_c) \quad (2.3)\ [26]$$

3. Melakukan pembaruan *cell state* pada langkah waktu saat ini C_t, yang menggabungkan kandidat *memory* c dan *long term memory* $\tilde{C}_t$.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (2.4)\ [26]$$

4. Mengkonfirmasi hasilnya h_t dari *forget gates* dimana informasi o_t*output* dan *cell state* C_t digunakan [26].

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (2.5)\ [26]$$

$$h_t = o_t * tanh(C_t) \quad (2.6)\ [26]$$

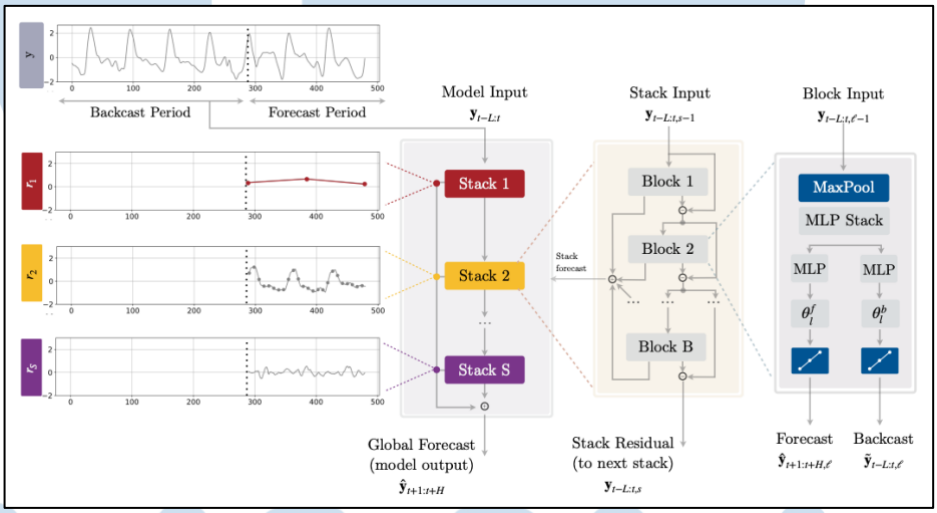
Penjelasan symbol :

1. W = *Input Weights*
2. U = *Recurrent Weights*
3. B = Bias
4. f = *Forget gate*
5. i = *Input gate*
6. o = *Output gate*
7. σ = Fungsi Sigmoid

2.2.5 N-Hits

N-Hits atau *Neural Hierarchical Interpolation for Time Series* adalah sebuah algoritma *Deep Learning* pengembangan dari pendekatan algoritma N-Beats (*Neural Basis Expansion Analysis*) dalam beberapa

aspek penting, yang menghasilkan prediksi lebih akurat dan juga lebih efisien sumber daya komputasi. Sama seperti NBeats, N-Hits melakukan proyeksi *non linear* pada fungsi dasar melalui beberapa *block*. Setiap block terdiri dari *Multilayer Perceptron* (*MLP*), yang akan belajar untuk menghasilkan nilai koefisien untuk keluaran *backcast* dan *forecast* dari basisnya. Singkatnya algoritma N-Hits menggunakan pendekatan *multirate sampling* dari *input signal* dan *multi-scale synthesis forecast*, yang menghasilkan konstruksi hirarkis dari sebuah *forecast* dan juga mengurangi kebutuhan komputasi serta meningkatkan akurasi peramalan.



Gambar 2.3 Arsitektur N-Hits [8]

Gambar 2.3 merupakan gambaran dari arsitektur N-Hits, dimana model N-Hits terdiri dari *S stacks*, *B blocks*. Setiap *block* terdiri dari beberapa MLP dengan *Activation Relu*. Setiap *block* terhubung melalui prinsip *doubly residual stacking* dengan *backcast* ($\widetilde{y_t} - L{:}\, t\ell,$) dan *forecast* ($\widehat{y_t} + 1{:}\, t + H, \ell$) *output* dari *block*. *Multi-rate input pooling*, *hierarchical interpolation* dan *backcast* residual secara bersama-sama menginduksi spesialisasi *forecast additive* dalam berbagai *signal bands*, mengurangi *memory footprint* dan waktu komputasi, meningkatkan parsiman arsitektur dan akurasi prediksi[8].

2.3 Tools yang digunakan

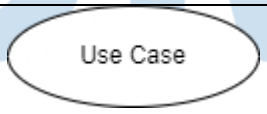
2.3.1 UML

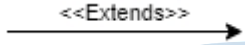
UML atau *Unified Modeling Language* adalah sebuah bahasa atau metode yang sering digunakan ketika mengembangakan sebuah sistem yang memiliki fungsi untuk menggambarkan alur dan cara kerja sebuah sistem, fungsi, tujuan dan mekanisme sistem tersebut[27]. Dalam UML terdapat dua jenis diagram yaitu *Structural Modeling Diagram* dan *Behavioral Modeling Diagram*. Dalam penelitian ini akan digunakkan jenis diagram *Behavioral modeling diagram* yaitu *Use Case Diagram*, *Class Diagram* dan *Activity Diagram* [28].

1. *Use Case Diagram*

   Teknik *Use Case Diagram* berfungsi untuk mengkomunikasikan interaksi manusia atau dapat disebut sebagai *actor* dengan apa yang dapat dilakukan *actor* pada sistem [27]. Terdapat beberapa simbol yang digunakan untuk merepresentasikan fungsi serta peran dalam sebuah *Use Case Diagram*, seperti yang terdapat pada Tabel 2.1 yang berisikan bentuk simbol, notasi dan fungsi yang digunakan dalam sebuah *Use Case Diagram* :

Tabel 2.1 Simbol *Use Case Diagram*

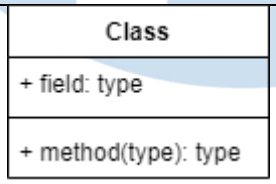
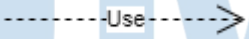
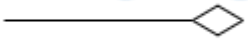
Simbol	Notasi	Fungsi
	*Actor*	Merepresentasikan pengguna yang berinteraksi dengan sistem informasi
	*Use Case*	Menunjukan deskripsi aksi berdasarkan urutan yang terjadi dalam sistem
	*Include*	Tambahan dari sebuah use case yang dimana use case yang ditambahkan memerlukan use case ini untuk berjalan.
	Association	Menunjukan hubungan antara aktor dan *use case*

Simbol	Notasi	Fungsi
	*Extension*	Tambahan dari sebuah *use case* yang dimana *use case* yang ditambahkan dapat berdisi sendiri.
	*Generalization*	Menunjukan hubungan *object parent* dan *child*

2. *Class Diagram*

Teknik *Class Diagram* adalah sebuah model statis yang menampilkan data dan informasi dari keseluruhan sistem. Model ini berfungsi untuk menggambarkan struktur dari sebuah sistem [27]. Terdapat beberapa simbol yang digunakan dalam sebuah *Class Diagram*, Tabel 2.2 berisikan bentuk simbol, notasi dan keterangan dari simbol yang ada pada *Class Diagram* :

Tabel 2. 2 Simbol *Class Diagram*

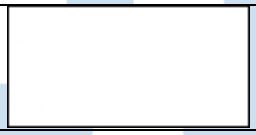
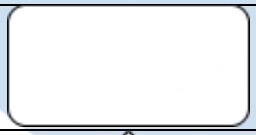
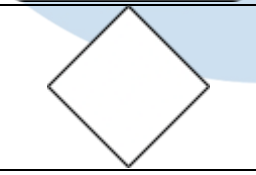
Simbol	Notasi	Fungsi
	*Class*	Komponen utama dari sebuah *Class Diagram* yang terdiri dari 3 bagian.
	Association	Menunjukan relasi antar kelas.
	*Generalization*	Menunjukan relasi antar kelas dengan relasi *parent* dan *child*.
	*Dependency*	Menunjukan relasi antar kelas yang memiliki ketergantungan.
	*Aggregation*	Menunjukan bahwa sebuah kelas ada bagian dari sebuah kelas

3. *Activity Diagram*

Activity Diagram adalah sebuah sebuah pemodelan yang menggambarkan sebuah sistem setelah melakukan pembuatan *Use Case Diagram*. *Activity Diagram* digambarkan dengan alur

kerja secara terstruktur berdasarkan *Use Case Diagram* yang ada, setiap aktivitas yang dilakukan pada *Activity Diagram* akan digambarkan dengan notasi-notasi sesuai dengan fungsinya [29]. Terdapat beberapa simbol yang digunakan dalam pembuatan *Activity Diagram*, tabel 2.3 menunjukan bentuk simbol notasi dan fungsi yang terdapat pada *Activity Diagram* :

Tabel 2. 3 Simbol *Activity Diagram*

Simbol	Notasi	Fungsi
	*Activity*	Berisikan seluruh interaksi yang terdapat pada *Activity Diagram.*
	*Start Point*	Merupakan awal dari sebuah aktifitas
	*Action*	Menandakan sebuah proses yang terjadi dalam A*ctivity Diagram*
	*Decision*	Digunakan ketika terdapat kondisi yang memerlukan sebuah pengambilan keputusan.
	*Line Connector*	Berfungsi untuk menunjukan koneksi antar *object*

2.3.2 Python

Python adalah bahasa pemrograman yang diinterpretasikan, interaktif, dan berorientasi objek. Python menggabungkan modul, pengecualian (*exceptions*), pengetikan dinamis (*dynamic typing*), tipe data dinamis yang *high level*, dan kelas-kelas (*classes*). Python mendukung paradigma pemrograman yang berbeda, tidak hanya berorientasi objek tetapi juga pemrograman prosedural dan fungsional [30]. Bahasa pemrograman Python memiliki kekuatan yang luar biasa dengan sintaks yang sangat jelas, sehingga python banyak digemari oleh banyak *developer*. Python sendiri dapat digunakan untuk *web*

development (Django, flask dan web2py), *GUI development* (tkInter, PySide dan kivy), *Scientific* dan *Numeric* (Pandas, dan Scipy) dan *software development* (Buildbot, Trac dan roundup) [31].

2.3.3 Django

Django adalah sebuah *web framework open source* yang menggunakan bahasa pemrograman Python. *Framework* ini diciptakan untuk membantu para *web developer* dalam memenuhi persyaratan dengan *deadline* waktu yang lebih singkat, sehingga dapat memenuhi batas *deadline* yang ditetapkan.[32]. Django menggunakan pola arsitektur MVC (*Model-View-Controller*) dimana model ini banyak digemari karena memudahkan *developer* untuk melakukan pemeliharaan kode dan membuat kode yang ditulis lebih terorganisir.

2.3.4 SQLite

SQLite adalah sebuah *software library* yang memungkinakan untuk sebuah RDBMS (*Relational Database Management System*) untuk dapat embedded pada dalam aplikasi. Tidak seperti kebanyakan RDBMS, SQLite memiliki sifat *serverless* yang artinya *database engine* berjalan pada proses yang sama dengan aplikasi [33]. SQLite banyak digunakan oleh aplikasi-aplikasi yang membutuhkan penyimpanan data pada *local database* tanpa koneksi internet yang membuat proses pengambilan data menjadi cepat dan efisien. *Web framework* Django, menggunakan SQLite sebagai *database backend default* untuk *project* di dalamnya karena SQLite mudah untuk di *set up* dan digunakan yang membuatnya cocok untuk sebuah *project* berukuran kecil hingga menengah [34].

2.3.5 Visual Studio Code

Visual Studio Code atau yang biasa singkat sebagai VSCode adalah sebuah *code editor* buatan Microsoft yang ringan serta *powerfull* yang berjalan pada komputer *desktop* dan bisa digunakan oleh windows, macOS dan Linux. VSCode memiliki beberapa *built-in support* untuk beberapa bahasa pemrograman seperti JavaScript, TypeScript dan

Node.js serta memiliki ekosistem yang sangat kaya dengan *extensions* yang dapat mendukung bahasa pemrograman lain seperti C++, C#, Java, dan Python[35].

2.3.6 MSE, RMSE dan MAPE

Untuk melakukan evaluasi terhadap model *machine learning* dan *deep learning* yang telah dibuat, digunakan beberapa *metrics* nilai yang berfungsi untuk mempermudah pemahaman terhadap seberapa baik model memprediksi harga jika dibandingkan dengan harga asli. Berikut adalah penjelasan dari masing masing *metrics* :

1. MSE

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(X_i - Y_i)^2 \qquad (2.7)\ [36]$$

MSE atau singkatan dari *Mean Squared Error* adalah *metrics* yang mengukur rata-rata dari kuadrat selisih dari nilai sebenarnya. Semakin kecil nilai MSE maka akan semakin bagus performa model

2. RMSE

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(X_i - Y_i)^2} \qquad (2.8)\ [36]$$

RMSE atau *Root Mean Squared Error* adalah nilai kuadrat dari MSE, nilai ini memberikan ukuran yang lebih mudah dimengerti karena memiliki nilai yang sama dengan variabel yang diukur.

3. MAPE

$$MAPE = \frac{1}{m}\sum_{i=1}^{m}\left|\frac{Y_i - X_i}{Y_i}\right| \qquad (2.9)\ [36]$$

MAPE atau *Mean Absolute Percentage Error* adalah metrics yang mengukur presentasi rata-rata dari kesalahan prediksi terhadap nilai sebenarnya. Semakin rendah nilai MAPE maka semakin baik performa prediksi dari model[36].

2.4 Penelitian Terdahulu

Tabel 2.4 Tabel penelitian terdahulu

No	Nama Jurnal	Jurnal	Tahun	Penulis	Hasil
1	Teknologi Informatika dan Komunikasi (Mantik)	*Bitcoin Price Prediction Using Long Short Term Memory* (LSTM): *Bitcoin Price Prediction Using Long Short Term Memory* (LSTM)[5]	2020	Ihyak Ulumuddin Sunardi Sunardi Abdul Fadlil	Penelitian ini menghasilkan bahwa harga bitcoin dapat diprediksi dengan model LSTM. Dari beberapa jenis *activation* yang digunakan, *activation softmax* memberikan hasil yang paling baik dengan nilai berikut : 1. Tingkat akurasi 97.48% 2. MAPE error 2.52% 3. RMSE 329.15 [5].
2	*International Journal of Industrial Optimization*	*Currency movement forecasting using time series analysis and long short-term memory*[6]	2020	Kristina Sanjaya Putri Siana Halim	LSTM menghasilkan nilai RMSE yang lebih rendah dari ARIMA. LSTM memiliki hasil prediksi yang lebih baik, hal ini dikarenakan LSTM memiliki kemampuan belajar sehingga dapat memanfaatkan ketika diberikan data yang semakin banyak sedangkan ARIMA tidak dapat menggunakannya [6].
3	*Computational and Mathematical Methods in Medicine*	*Forecasting COVID-19 Pandemic Using Prophet, ARIMA, and Hybrid Stacked LSTM-GRU Models in India* [37]	2022	Sweeti Sah B Surendiran R Dhanalakshmi Sachi Nandan Mohanty Fayadh Alenezi Kemal Polat	Dalam penelitian ini, dilakukan komparasi performa algoritma RNN, GRU, LSTM, prophet dan ARIMA untuk memprediksi angka kasus COVID-19 di India. Hasilnya adalah akurasi hasil algoritma LSTM-GRU memiliki nilai RMSE yang paling rendah yaitu 69.92 disusul dengan algoritma-algoritma berikut : 1. GRU : 94.558 2. RNN : 120.35 3. LSTM : 134.505 4. Prophet : 568.58 5. ARIMA : 1260 [37].

No	Nama Jurnal	Jurnal	Tahun	Penulis	Hasil
4	Procedia Computer Science	*Time series analysis and forecasting of coronavirus disease in Indonesia using ARIMA model and PROPHET* [38]	2021	Christophorus Beneditto Aditya Satrio William Darmawan Bellatasya Unrica Nadia Novita Hanafiah	Pada penelitian ini, dilakukan prediksi angka kematian atas kasus COVID-19 di Indonesia dengan menggunakan Algoritma Prophet dan ARIMA. Hasil dari penelitian ini adalah algoritma prophet menghasilkan nilai yang lebih baik yaitu sebesar 91% akurasi, dibandingkan dengan nilai algoritma ARIMA yang tidak sampai 50% [38].
5	JURNAL INFORMATIKA	*Application to predict the new student's score using time series algorithm* [10]	2020	Sinar Nadhif Ilyasa Husni Thamrin	Penelitian ini dilakukan untuk membangun *web application* dengan menggunakan Django untuk membandingkan performa dari algoritma *Auto Regression*, *ARMA* dan *Triple Exponential Smoothing* untuk melakukan prediksi nilai siswa. Hasil dari penelitian ini adalah algoritma *Auto Regression* memiliki hasil yang lebih baik dibandingkan dengan *ARMA* dan *Triple Exponential Smoothing*.
6	Heliyon	*The role of the mass vaccination programme in combating the COVID-19 pandemic: An LSTM-based analysis of COVID-19 confirmed cases* [39]	2023	Seng Hansun Vincent Charles Tatiana Gherman	Penelitian ini melakukan prediksi dengan menggunakan algoritma LSTM untuk memprediksi tren kasus COVID-19 di beberapa negara. Hasil dari penelitian ini adalah model LSTM dapat digunakan untuk memprediksi nilai dari tren kasus COVID-19 dengan akurasi 5.977% dan 10.388%.

No	Nama Jurnal	Jurnal	Tahun	Penulis	Hasil
7	International Journal of New Media Technology	*Predicting the Case of COVID-19 in Indonesia using Neural Prophet Model* [40]	2022	Efraim Yahya Wijaya Alethea Suryadibrata	Pada penelitian ini dilakukan prediksi kasus COVID-19 di Indonesia dengan menggunakan beberapa algoritma prophet dan neural prophet. Hasil dari penelitian ini neural prophet memiliki nilai akurasi yang lebih tinggi daripada algoritma prophet dengan nilai RMSE 5.728 pada eksperimen pertama dan 2.040 pada eksperimen kedua, sedangkan prophet memiliki nilai RMSE 28.798 pada eksperimen pertama dan 2.122 pada eksperimen kedua.

Dari daftar jurnal terdahulu pada tabel 2.4, penelitian sebelumnya telah memberikan beberapa temuan yang relevan. Pertama, ditemukan bahwa penggunaan algoritma LSTM (*Long Short-Term Memory*) dapat memberikan prediksi yang baik untuk harga bitcoin berdasarkan data time series. Temuan ini menunjukkan bahwa LSTM dapat menjadi pilihan yang efektif dalam memprediksi pergerakan harga *cryptocurrency*.

Selanjutnya, hasil penelitian menunjukkan bahwa algoritma Prophet menghasilkan prediksi yang lebih baik daripada model ARIMA (*AutoRegressive Integrated Moving Average*). Selain itu, penelitian terdahulu telah melakukan pembangunan web application yang melibatkan objek dan algoritma yang berbeda. Dalam penelitian ini, kami akan menggunakan algoritma Prophet, LSTM, dan NHits pada data *time series* yang dipilih oleh pengguna. Melalui *web application* yang dikembangkan, pengguna dapat melakukan pemilihan data *time series* yang ingin diprediksi menggunakan ketiga algoritma tersebut.

Dengan demikian, penelitian ini memperluas penelitian terdahulu dengan mengintegrasikan berbagai temuan dan mengembangkan *web application* yang memungkinkan pengguna untuk memprediksi data *time series* cryptocurrency dengan menggunakan algoritma Prophet, LSTM, dan NHits.