BAB II
LANDASAN TEORI

2.1 Tinjauan Teori

Berikut adalah beberapa teori mengenai objek yang digunakan dalam penelitian ini diantaranya:

2.1.1 Saham Transportasi

Saham adalah sebuah bukti berupa tanda kepemilikan seseorang dan lembaga atau organisasi pada sebuah perusahaan [11]. Transportasi adalah sebuah kendaraan atau muatan yang dioperasikan oleh manusia maupun mesin yang berfungsi untuk perpindahan manusia dan barang dari satu tempat ke tempat lain [12]. Tujuan adanya transportasi ini adalah membawa bantuan secara fisik dengan keefektifan dan keamanan dari pergerakan dan perpindahan manusia dari tempat satu ke tempat lain pada suatu wilayah tertentu [13]. Mobilitas manusia yang efektif ditentukan oleh adanya sektor transportasi karena berfungsi sebagai penyedia akses menuju suatu tempat tujuan [14]. Saham transportasi adalah sebuah bukti surat berharga kepemilikan dari perusahaan yang bergerak pada sektor transportasi dan biasanya produk atau jasa yang ditawarkan berupa angkutan transportasi [15].

2.1.2 *Data Mining*

Data Mining merupakan tahap menganalisis data dari berbagai perspektif yang berbeda berdasarkan informasi dari beberapa penemuan korelasi atau pola pada ratusan maupun ribuan data untuk meningkatkan keuntungan dan memperkecil biaya pengeluaran [16]. Teknik analisis ini dibagi menjadi beberapa metode dalam mencari informasi dari ratusan maupun ribuan data. Berikut ini merupakan jenis-jenis metode yang digunakan dalam teknik *data mining* diantaranya [17]:

1. Deskripsi

   Deskripsi adalah jenis teknik *data mining* yang menggambarkan atau menjelaskan korelasi atau pola kecenderungan yang terdapat dalam data.

2. Estimasi

   Estimasi adalah jenis teknik *data mining* yang menggambarkan dan menjelaskan konsep yang mendekati sama dengan klasifikasi, tetapi variabel target estimasi lebih ke arah numerik daripada kategori. Model algoritma yang dibangun menggunakan data yang menyediakan nilai variabel target sebagai nilai prediksi.

3. Prediksi

   Prediksi adalah jenis teknik *data mining* yang menggambarkan dan menjelaskan konsep peramalan sebuah nilai yang belum diketahui sebelumnya dan akan memperkirakan nilai untuk masa mendatang.

4. Klasifikasi

   Klasifikasi adalah jenis teknik *data mining* yang menggambarkan dan menjelaskan konsep penggolongan kelas variabel kategori yang akan dipisahkan dari variabel lainnya dan menjadi target klasifikasi.

5. Pengklasteran

   Pengklasteran adalah jenis teknik *data mining* yang menggambarkan dan menjelaskan konsep pengelompokkan pada *record* data yang nantinya akan dibentuk suatu kelas-kelas objek yang memiliki kesamaan. Pengelompokkan kelas-kelas tersebut nantinya akan ditampilkan ke dalam bentuk visualisasi dari persebaran *data point* dari masing-masing objek.

6. Asosiasi

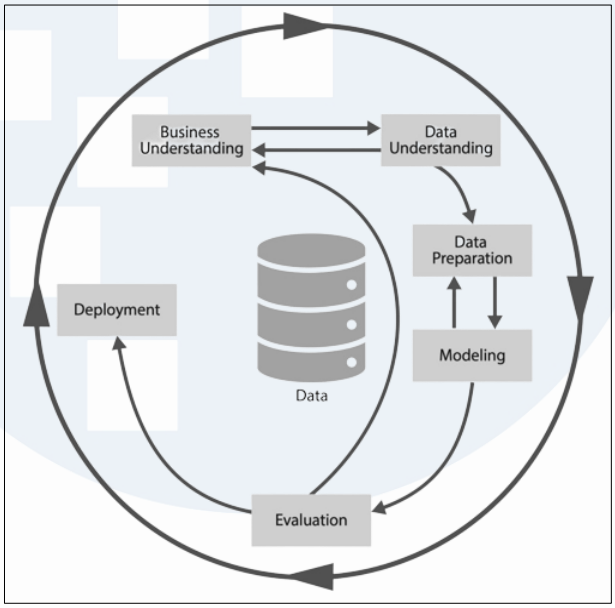
   Asosiasi adalah jenis teknik *data mining* yang menggambarkan dan menjelaskan konsep dari penemuan atribut data yang muncul dalam satu waktu. Teknik ini biasanya dalam dunia bisnis disebut sebagai analisis keranjang belanja *(Market Basket Analysis)*.

2.2 *Framework* yang digunakan

Berikut adalah framework dan model algoritma dalam mengembangkan teknik *data mining* diantaranya:

2.2.1 CRISP-DM *(Cross Industry Standard Process for Data Mining)*

CRISP-DM *(Cross Industry Standard Process for Data Mining)* merupakan *framework* untuk menerjemahkan masalah bisnis menjadi penugasan *data mining* [18]. *Framework* ini telah dikembangkan oleh beberapa perusahaan di Eropa, yaitu Integral Solutions Ltd (ISL), Teradata, dan OHRA [19]. Penugasan *data mining* dengan CRISP-DM bertujuan untuk membuat projek menjadi lebih ekonomis dan mudah dikelola [20].



Gambar 2.1 CRISP-DM *(Cross Industry Standard Process for Data Mining)*
Sumber: [21]

Berdasarkan Gambar 2.1 di atas, CRISP-DM memiliki 6 tahapan diantaranya [22]:

1. *Business Understanding*

   Business Understanding merupakan proses atau tahapan dalam memotivasi suatu tujuan bisnis untuk memaksimalkan secara waktu dan efisiensi dari *machine learning* dan *deep learning* dalam penggunaan analisa prediksi. Tujuan tersebut untuk transformasi ke dalam masalah data mining secara lebih spesifik dengan mengidentifikasi komponen machine learning dan mencari secara detail mengenai komponen tersebut yang memiliki kontribusi yang sangat kecil dalam mencapai tujuan bisnis tertentu.

2. *Data Understanding*

   Data Understanding merupakan proses atau tahapan dalam menemukan hipotesa atau informasi mengenai tujuan projek *data mining* secara lebih spesifik dengan mengidentifikasi komponen *machine learning* dan mencari secara detail mengenai komponen tersebut yang memiliki kontribusi yang sangat kecil dalam mencapai tujuan bisnis tertentu.

3. *Data Preparation*

   Data Preparation merupakan proses atau tahapan dalam mengumpulkan data yang relevan dan mempersiapkannya untuk penugasan *data mining* yang sebenarnya. Hal tersebut meliputi tahap *pre-processing, data reduction, filtering,* dan *feature generation* sesuai dengan tujuan projek *data mining*. Selain itu, data yang dikumpulkan akan diberikan label khusus dalam menentukan perbedaan klasifikasi dari suatu prediksi.

4. *Modeling*

   Modeling merupakan proses atau tahapan dalam *data mining* untuk mencari dan menentukan pengaturan parameter yang dipilih berdasarkan model algoritma, yang nantinya penugasan data mining akan dijalankan pada pemrosesan data. Model algoritma yang digunakan dalam penugasan *data mining* untuk pemrosesan data biasanya adalah klasifikasi dan regresi.

5. *Evaluation*

   Evaluation merupakan proses atau tahapan dalam *data mining* untuk menguji *data training* yang telah ditentukan berdasarkan permasalahan bisnis yang ada. Selain *data training*, terdapat juga *data testing* yang telah dipersiapkan pada tahap *data preparation* dan *modeling* kecuali proses pemberian label pada data.

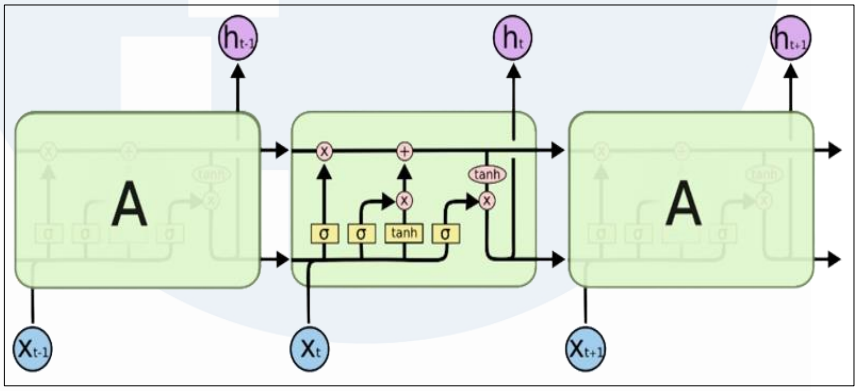
6. *Deployment*

   Deployment merupakan proses atau tahapan dalam *data mining* untuk mentransfer proses analisa prediksi yang telah dilakukan

dalam bentuk pengembangan model algoritma pada sebuah sistem analisa prediksi berbasis *web* yang biasa disebut dengan tahap *production*.

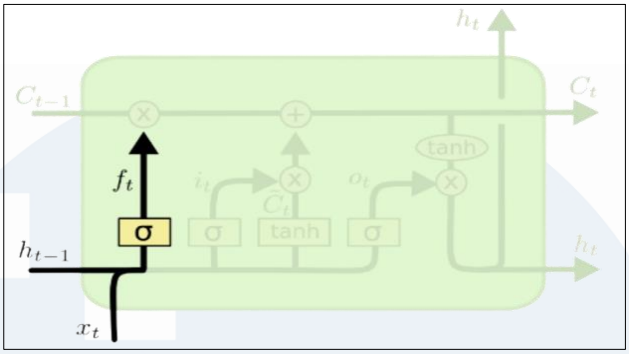
2.2.2 LSTM *(Long Short-Term Memory)*

LSTM *(Long Short-Term Memory)* merupakan salah satu sub-unit dari model algoritma *Reccurent Neural Network (*CNN*)* [23]. Algoritma LSTM ini sangat cocok digunakan untuk mengklasifikasi, pemrosesan, dan pembuatan prediksi berdasarkan deret waktu tunggal [24]. LSTM juga memiliki *memory block* yang nantinya akan menentukan nilai yang telah ditentukan sebagai *output* yang relevan terhadap *input* yang diberikan [25].



Gambar 2.2 Arsitektur LSTM
Sumber: [26]

Berdasarkan Gambar 2.2 di atas adalah arsitektur dari model algoritma LSTM dengan bagian bawah terdapat adanya *cell gates* yang berguna untuk meregulasi beberapa informasi yang akan dikeluarkan ke bagian *cell state* atau unit berikutnya. Pada bagian *cell state* merupakan jalur untuk mengirimkan informasi ke unit selanjutnya [26]. Proses pengiriman informasi tersebut awalnya dari lapisan *forget gate* seperti pada Gambar 2.3.



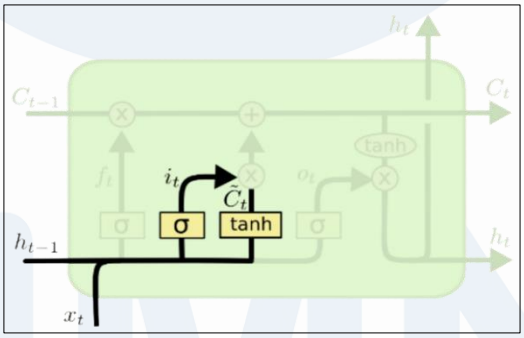
Gambar 2.3 Arsitektur *Forget Gate Layer*
Sumber: [26]

Berdasarkan Gambar 2.3 di atas, *forget gate* merupakan *layer* pertama yang akan menghitung *output* dari informasi yang akan dihapus dan informasi akan diteruskan ke lapisan *cell state* seperti persamaan (2.1) [26].

$$f_t = \sigma\left(Wf[ht - 1, xt] + bf\right) \quad (2.1)$$

Rumus 2.1 Rumus *Forget Gate Layer*

Hasil dari perhitungan pada lapisan *forget gate* akan dikirimkan lapisan berikutnya yaitu *input gate* seperti pada Gambar 2.4.



Gambar 2.4 Arsitektur *Input Gate Layer*
Sumber: [26]

Berdasarkan Gambar 2.4 di atas merupakan lapisan *input gate* yang menampung *output* dari hasil lapisan *forget gate* yang telah dilakukan perhitungan sebelumnya. Pada lapisan *input gate* dilakukan 2 tahap perhitungan yaitu untuk *output* ke dalam lapisan *input gate* pada persamaan (2.2) dan hasilnya akan dihitung kembali untuk diarahkan ke lapisan *cell state* seperti pada persamaan (2.2) [26].

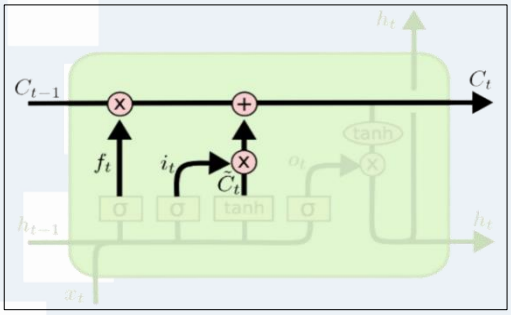
$$it = \sigma.\left(Wi\,[ht - 1, xt] + bi\right) \quad (2.2)$$

Rumus 2.2 Rumus *Input Gate*

$$\hat{C}t = tanh\,(Wc\,.\,[ht - 1, xt] + bc) \quad (2.3)$$

Rumus 2.3 Rumus Hasil *Input Gate* ke *Cell State*

Hasil dari perhitungan pada lapisan *input gate* yang dikembalikan ke lapisan *cell state* akan dikirimkan ke lapisan *update gate* seperti pada Gambar 2.5.



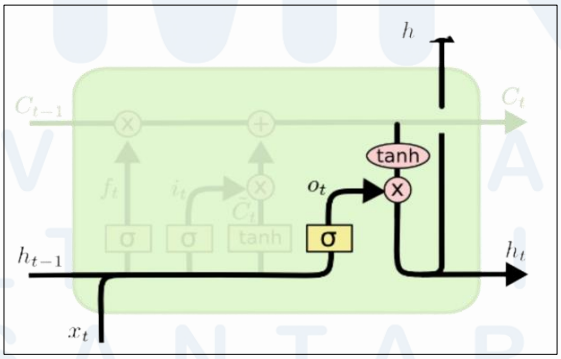
Gambar 2.5 Arsitektur *Update Gate Layer*
Sumber: [26]

Berdasarkan Gambar 2.5 di atas merupakan arsitektur dari lapisan *update gate* yang menampung hasil *output* dari perhitungan pada lapisan *input gate* untuk dihitung kembali dan mengubah hasil perhitungan pada lapisan *cell state* yang didapat dari 2 lapisan sebelumnya dari proses penghapusan maupun penambahan informasi seperti pada persamaan (2.4) [26].

$$Ct = ft\,x\,Ct - 1 + it\,x\,\hat{C}t \quad (2.4)$$

Rumus 2.4 Rumus *Update Gate*

Hasil dari perhitungan pada lapisan *update gate* nantinya akan dikirimkan ke dalam lapisan *output gate* seperti pada Gambar 2.6.



Gambar 2.6 Arsitektur *Output Gate Layer*
Sumber: [26]

Berdasarkan Gambar 2.6 di atas merupakan arsitektur dari lapisan *output gate* yang menampung hasil perhitungan atau pengolahan informasi dari tahap lapisan awal yaitu *forget gate* sampai *update gate*. Pada lapisan ini merupakan hasil keseluruhan perhitungan dengan 2 kali tahap perhitungan untuk mendapatkan hasil secara keseluruhan seperti pada persamaan (2.3) [26].

$$Ot = \sigma\left(Wo\,.[ht - 1, xt] + bo\right) \quad (2.5)$$

Rumus 2.5 Rumus *Forget Gate* sampai *Update Gate*

Arsitektur yang terkandung dalam LSTM memiliki beberapa prosedur perhitungan dalam menyampaikan hasil pada setiap *layer* atau lapisan yang terkandung didalamnya [26]. Berikut Tabel 2.1 merupakan prosedur dari perhitungan penyampaian hasil pada setiap *layer* pada LSTM.

Tabel 2.1 *Pseudocode* LSTM *Layer*

Algorithm 1: *Pseudocode of LSTM Layer*
1: ***Input*:** $x = [x_1, \ldots, x_n], x_t \in \mathbb{R}^m$
2: ***Given parameters*:** $\mathbf{W}_f, \mathbf{U}_f, \boldsymbol{b}_f, \mathbf{W}_{\tilde{c}}, \mathbf{U}_{\tilde{c}}, \boldsymbol{b}_{\tilde{c}}, \mathbf{W}_i, \mathbf{U}_i, \boldsymbol{b}_i, \mathbf{W}_o, \mathbf{U}_0, \boldsymbol{b}_0$
3: ***Initiliaze*** $\boldsymbol{h_0}, \boldsymbol{c_0} = \vec{\mathbf{0}}$ *of length p*
4: ***for*** t=1, ..., n ***do***
5: ***Calculate*** $\boldsymbol{f}_t$ (2.1), $\hat{\mathbf{c}}_t$ (2.3), $\boldsymbol{i}_t$ (2.2)
6: ***Update cell state*** $\boldsymbol{c}_t$ (2.4)
7: ***Calculate*** $\boldsymbol{o}_t$ (2.5)
8: ***end for***
9: ***Output*:** $h = [h_1, \ldots, h_n], h_t \in \mathbb{R}^p$

Sumber: [27]

2.2.2.1 *Activation*

Activation adalah fungsi yang digunakan untuk memutuskan apakah suatu *node* akan diaktifkan atau tidak pada permodelan *neural network* [28]. Fungsi *activation* juga dapat diterapkan pada *output* dari sebuah *neuron* untuk mendapatkan hasil yang lebih spesifik berupa transformasi *non-linear* dari sinyal input [29]. Berikut adalah jenis-jenis *activation* yang ada pada permodelan *neural network* diantaranya:

1. *Linear*

   Linear activation merupakan fungsi yang memiliki nilai gradien berbanding lurus dengan *input* yang dimasukkan pada permodelan *neural network* [30]. Fungsi *Linear* juga

dapat mengembalikan jumlah *weighted* pada masing-masing *input* yang dimasukkan dan memiliki nilai yang setara [31]. Pengembalian jumlah *weighted* pada masing-masing nilai *input* dapat dihitung seperti pada persamaan (2.4) merupakan nilai yang konstan [31].

$$f(x) = x \quad (2.6)$$

Rumus 2.6 Rumus *Linear Activation*

2. *ReLU*

ReLU (Rectified Linear Unit) activation adalah fungsi sederhana yang merupakan fungsi sebagai identitas pada suatu nilai *input* positif dan 0 untuk nilai *input* negatif [32]. Penerapan fungsi *ReLU* ini dapat memecahkan masalah pada fungsi *sigmoid* dan *tanh activation* [33]. Penerapan fungsi *ReLU* pada fungsi *sigmoid* dan *tanh activation* dapat dihitung dengan persamaan (2.7) [33].

$$ReLU(x) = max(x, 0) \quad (2.7)$$

Rumus 2.7 Rumus *ReLU Activation*

3. *Tanh*

Tanh activation merupakan fungsi perhitungan sederhana dari turunan yang digunakan dalam proses *backpropagation* dari permodelan *Reccurent Neural Network (*RNN*)* [34]. Fungsi *tanh* adalah singkatan dari fungsi *hyperbolic tanh* untuk digunakan pada fungsi *piecewise* sebagai penggantian [35]. Penerapan fungsi *tanh* pada fungsi *piecewise* dapat dihitung dengan persamaan (2.8) [35].

$$tanh(x) = 2sigmoid(2x) - 1 \quad (2.8)$$

Rumus 2.8 Rumus *Tanh Activation*

4. *Sigmoid*

   Sigmoid activation adalah fungsi yang dirancang dengan menggunakan beban *non-linear resistive* yang dibutuhkan arus sebagai nilai *input* dan memberikan tegangan pada nilai *output* pada *neuron* [36]. Fungsi *sigmoid* biasanya mengambil nilai *real* sebagai *input* dan *output* dalam rentang 0 sampai 1 [37]. Pengambilan nilai *real* tersebut sebagai nilai *input* dan *output* dapat dihitung dengan persamaan (2.9) [37].

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.9)$$

Rumus 2.9 Rumus *Sigmoid Activation*

2.2.2.2 *Optimizer*

Optimizer adalah suatu target untuk meningkatkan efesiensi dari pemrosesan atau pembelajaran dari model algoritma prediksi berdasarkan peningkatan *accuracy score* pada penugasan prediksi [38]. Pada sebutan lain, optimalisasi dari *optimizer* dapat membentuk yang dimiliki pada *accuracy score* dengan memanfaatkan suatu bobot agar hasilnya bisa memprediksi seakurat mungkin [39]. Adapun jenis-jenis *optimizer* dalam peningkatan akurasi diantaranya:

1. *Adam*

   Adam optimizer merupakan suatu metode algoritma optimasi berbasis *stochastic gradient* yang sangat cocok untuk diimplementasikan secara langsung untuk model apapun dalam hal kumpulan data dan parameter yang besar [40]. Fungsi dari *optimizer adam* dalam implementasi modelnya, memiliki kemampuan yang dapat memperbarui bobot dan *learning rate* secara otomatis [41]. Implementasi dalam memperbarui bobot dan *learning rate* secara otomatis dapat dihitung dengan persamaan (2.10) [41].

$$m_n = E[X^n] \quad (2.10)$$

Rumus 2.10 Rumus *Adam Optimizer*

2. *AdaGrad*

AdaGrad Optimizer merupakan varian optimasi algoritma *Stochastic Gradient Descent* yang mengubah kecepatan *learning rate* untuk parameter yang berbeda [42]. *Optimizer* ini menggunakan *learning rate* yang berbeda setiap saat pada setiap parameter, menggunakan perhitungan pada nilai gradien sebelumnya [43]. Perhitungan pada nilai gradien tersebut pada setiap parameter dapat dihitung dengan persamaan (2.11) [43].

$$g_{t,i} = \nabla_{\theta_t} J(\theta_{t,i}) \quad (2.11)$$

Rumus 2.11 Rumus *AdaGrad Optimizer*

3. *Nadam*

Nadam optimizer merupakan perpanjangan dari algoritma *Adam* dengan menggabungkan momentum *Nesterov* pada *gradient descent* [44]. *Optimizer* ini memiliki kendala yang lebih kuat dalam *learning rate* dan juga memiliki dampak pada pembaruan nilai gradien [45]. Pembaruan pada nilai gradien tersebut dapat dihitung dengan persamaan (2.12) [45].

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{u}_t + \varepsilon}} \left(\beta_1 \hat{a}_t + \frac{(1 - \beta_1) g_t}{1 - \beta_1^t} \right) \quad (2.12)$$

Rumus 2.12 Rumus *Nadam Optimizer*

4. *RMSProp*

RMSProp optimizer merupakan singkatan dari *Root Mean Square Propagation* yang dikembangkan agar hasilnya lebih baik dan tidak mengurangi *learning rate* untuk mencegah terjadinya konvergensi [46]. Algoritma *optimizer* ini juga membagi laju *learning rate* dengan rata-rata nilai gradien

kuadrat yang hilang secara eksponensial [47]. Hasil dari nilai rata-rata gradien tersebut dapat dihitung seperti pada persamaan (2.13) dengan nilai V_{dW} berasal dari persamaan (2.14) [47].

$$V_{dW} = \beta V_{dW} + (1 - \beta)\, dW^2 \quad (2.13)$$

Rumus 2.13 Rumus Laju *Learning Rate* dari Rata-Rata Gradien

$$W = W - a\, \frac{dW}{\sqrt{V_{dW} + \varepsilon}} \quad (2.14)$$

Rumus 2.14 Rumus *RMSProp Optimizer*

5. *AdaDelta*

   AdaDelta optimizer merupakan algoritma optimasi perpanjangan dari *AdaGrad* yang menyelesaikan masalah laju *learning rate* yang hilang, tetapi tidak mengumpulkan nilai gradien kuadrat [48]. *AdaDelta* menampilkan perubahan nilai yang lebih kecil pada parameter yang sering diperbarui dan melakukan perubahan nilai yang lebih besar pada parameter yang jarang diperbarui [49]. Perubahan dari masing-masing nilai tersebut dapat dihitung dengan persamaan (2.15) [49].

$$\theta = \theta - \frac{\sqrt{\beta\, .E\, [\Delta\theta^2]_{t-1} + (1 - \beta)\, .\Delta\theta^2 + \epsilon}}{\sqrt{\beta\, .E[g^2]_{t-1} + (1 - \beta)\, .g^2 + \epsilon}}\, g_t \quad (2.15)$$

Rumus 2.15 Rumus *AdaDelta Optimizer*

6. *SGD*

   SGD optimizer merupakan optimasi algoritma singkatan dari *Stochastic Gradient Descent* yang berfungsi sebagai pengoptimal pilihan untuk berbagai kemajuan terbaru dari *deep learning* pada seluruh domain [50]. *Optimizer* ini juga merupakan varian metode optimasi penurunan gradien pada setiap iterasi yang dapat meminimalkan kesalahan dan mempertimbangkan kesalahan untuk setiap *data point* [51].

Hasil dari penurunan gradien pada setiap iterasi dapat dihitung dengan persamaan (2.16) [51].

$$w_k = w_{k-1} - a_{k-1} \widehat{\nabla} f(w_{k-1}) \quad (2.16)$$

Rumus 2.16 Rumus *SGD Optimizer*

7. *AdaMax*

AdaMax optimizer merupakan varian optimasi algoritma dari *Adam* yang membentuk rentang laju *learning rate* lebih sederhana dan telah digunakan untuk melatih permodelan *Convolutional Neural Network (CNN)* [52]. *AdaMax* memiliki keuntungan terhadap pengurangan sensitifitas pada *hyper-parameter* yaitu *learning rate* dan nilai konstanta yang ada dipilih secara manual saat menggunakan optimasi tersebut [53]. Hasil pengurangan sensitifitas pada *hyper-parameter* terhadap *learning rate* dapat dihitung dengan persamaan (2.17) [53].

$$u_t = max(\beta_2 * v_{t-1}, || g_t ||) \quad (2.17)$$

Rumus 2.17 Rumus *AdaMax Optimizer*

2.2.3 *Mean Imputation*

Mean Imputation merupakan salah satu metode *single imputation* yang paling umum untuk mengganti data yang hilang dengan rata-rata nilai *sample* atau median, modus maupun distribusi data [54]. Proses penggantian data yang hilang ini mengabaikan nilai korelasi dari masing-masing data karena nilai diperhitungkan secara independen dari prediktor apapun terhadap nilai bias dalam varians [55]. Metode ini juga dapat mengabaikan waktu komputasi pada saat penggantian data yang hilang serta secara drastis dapat mengurangi kinerja prediksi permodelan [56].

2.2.4 *MinMaxScaler*

MinMaxScaler merupakan proses teknik untuk normalisasi data dan memastikan semua data memiliki kisaran nilai yang serupa [57]. Proses normalisasi ini merupakan tahap *preprocessing* yang dilakukan untuk mengurangi nilai *sample* dengan nilai *sample* yang terkecil dan dilakukan

proses pembagian dengan nilai *sample* terbesar yang telah dikurangi oleh nilai *sample* terkecil pada *feature selection* [58]. Proses normalisasi tersebut dapat dihitung untuk mendapatkan nilai *sample* terkecil dengan persamaan (2.18) [58].

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (2.18)$$

Rumus 2.18 Rumus *MinMaxScaler*

2.2.5 *Evaluation Model*

Evaluation Model merupakan suatu evaluasi dari tingkat pengukuran terhadap kinerja sistem atau permodelan *machine learning* maupun *deep learning* [59]. Proses evaluasi untuk mengukur kinerja permodelan biasanya untuk memeriksa akurasi hasil eksperimen yang dilakukan yang dibentuk menjadi beberapa skenario pengujian [60]. Skenario pengujian dari hasil eksperimen dibagi menjadi beberapa evaluasi diantaranya:

1. *Mean Absolute Percentage Error* (MAPE)

   Mean Absolute Percentage Error (MAPE) adalah salah satu evaluasi pengujian dengan menunjukkan kesalahan dari nilai persentase pada hasil peramalan terhadap nilai aktual data selama periode tertentu [61]. Evaluasi MAPE dengan nilai yang semakin kecil, maka nilai taksiran semakin mendekati nilai sebenarnya atau metode tersebut akan dipilih menjadi yang terbaik [62]. Berikut nilai taksiran dapat diketahui dari perhitungan pada persamaan (2.19) [63].

   $$MAPE = \left(\frac{1}{n}\sum_{i=1}^{n}\left|\frac{p_i - q_i}{p_i}\right|\right) x\ 100 \quad (2.19)$$

   Rumus 2.19 Rumus *Mean Absolute Percentage Error* (MAPE)

2. *Mean Absolute Error* (MAE)

   Mean Absolute Error (MAE) adalah salah satu evaluasi pengujian yang dapat digunakan jika nilai *outlier* pada masing-masing atribut mewakili bagian data yang kotor [64]. Hasil dari MAE didefinisikan sebagai rata-rata dari perbedaan nilai mutlak antara

prediksi dan pengamatan yang sebenarnya dengan nilai bobot yang sama [65]. Berikut hasil perbedaan nilai mutlak antara prediksi dan observasi dapat dihitung dengan persamaan (2.20) [66].

$$MAE = \frac{\sum_{n=1}^{N} |\hat{r}_n - r_n|}{N} \quad (2.20)$$

Rumus 2.20 Rumus *Mean Absolute Error* (MAE)

3. *Root Mean Squared Error* (RMSE)

   Root Mean Squared Error (RMSE) adalah suatu metode dalam megevaluasi pada teknik peramalan yang dapat digunakan untuk mengukur tingkat akurasi pada hasil peramalan atau perkiraan suatu model algoritma [67]. Evaluasi ini biasa disebut sebagai jarak rata-rata yang diukur secara vertikal dari nilai aktual atau sebenarnya dengan nilai prediksi yang sesuai dengan *fit line* [68]. Berikut jarak rata-rata yang diukur dari nilai aktual dengan prediksi dapat dihitung dengan persamaan (2.21) [69].

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(A_i - \hat{P}_i\right)^2} \quad (2.21)$$

Rumus 2.21 Rumus *Root Mean Squared Error* (RMSE)

4. *Mean Squared Error* (MSE)

   Mean Squared Error (MSE) adalah suatu metode evaluasi dalam mengukur nilai rata – rata deviasi kuadrat dari nilai yang diperkirakan [70]. Hasil evaluasi dari MSE dapat memberikan informasi seberapa dekat garis yang paling cocok ke himpunan *data point* atau semakin dekat hasil MSE dengan nilai 0, maka hasil prediksi semakin akurat [71]. Berikut hasil evaluasi dari MSE dapat dihitung dengan persamaan (2.22) [72].

$$MSE = \sum_{i=1}^{n} \left(y_i - y_i^p\right)^2 \quad (2.22)$$

Rumus 2.22 Rumus *Mean Squared Error* (MSE)

5. *R-Squared*

   R-Squared adalah suatu metode evaluasi untuk mengukur persentase dari total dalam variabel dependen yang diperhitungkan

dengan variabel independen [73]. Evaluasi ini juga digunakan sebagai indeks yang dominan dalam algoritma regresi untuk memverifikasi hasil akurasi dari prediksi [74]. Berikut hasil pengukuran dari hasil *R-Squared* dalam memverifikasi hasil akurasi dari prediksi dapat dihitung dengan persamaan (2.23) [75].

$$R - Squared = \left[\frac{\left[\left[\sum Q_0 \, Q_p \right] - \left[\frac{\sum Q_0 \, Q_p}{N} \right] \right]}{\sqrt{\left[\sum Q_{p^2} - \frac{\left(\sum Q_p \right)^2}{N} \right] \left[\sum {Q_0}^2 - \frac{\left(\sum Q_0 \right)^2}{N} \right]}} \right]^2 \quad (2.23)$$

Rumus 2. 23 Rumus *R-Squared*

6. *Elapsed Time*

   Elapsed Time adalah rata-rata waktu yang dapat diambil sejak *data point* dihasilkan oleh *SenSE* sampai hasil performa diterima [76]. Rata-rata waktu tersebut juga biasa disebut dengan *exeTime* yaitu waktu yang menjalankan *task* atau penugasan permodelan pada perangkat yang diberikan dengan asumsi dapat menjalankan *task* beberapa kali di perangkat dan mengukur performa waktu dari eksekusi [77]. Berikut hasil perhitungan dari *elapsed time* mengenai perintah proses penugasan atau *task* yang dijalankan dapat dihitung dengan persamaan (2.24) [78].

$$E(t) = \frac{||u(t) - \tilde{u}(t)||}{\langle ||u(t)||^2 \rangle^{1/2}} \quad (2.24)$$

Rumus 2.24 Rumus *Elapsed Time*

7. *Shapiro-Wilk Test*

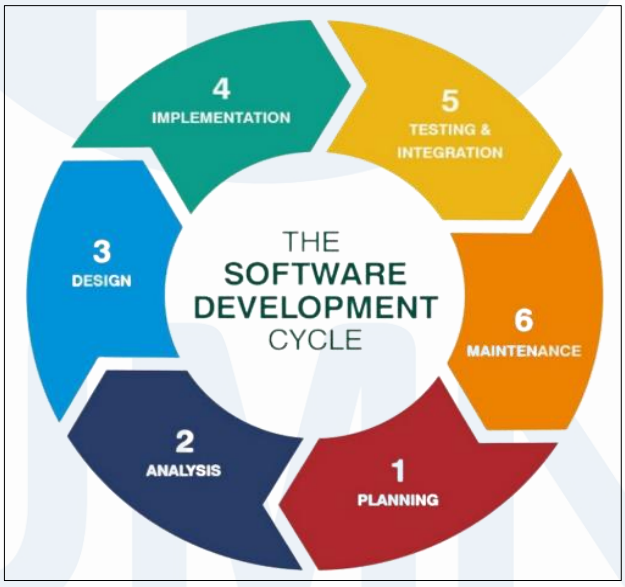
   Shapiro-Wilk Test adalah metode uji hipotesis statistik yang paling banyak digunakan untuk menguji normalitas data [79]. Menurut uji normalitas *Shapiro-Wilk*, data dari masing-masing kelompok memenuhi distribusi normal jika nilai $p > 0.05$ [80]. Uji hipotesis dari *shapiro-wilk* dapat dihitung seperti pada persamaan (2.25) [81].

$$W = \frac{\{\sum_{i-1}^{n} a_i(X_{(n-1+1):n} - X_{i:n})\}^2}{\sum_{i=1}^{n}(X_i - \bar{X})_2} \quad (2.25)$$

Rumus 2.25 Rumus *Shapiro-Wilk Test*

2.2.6 *Agile Software Development*

Agile Software Development adalah salah satu metode pengembangan sistem dengan menerapkan beberapa prinsip dalam mengembangkan suatu sistem dengan menciptakan solusi melalui upaya kolaboratif yang fleksibel antar tim fungsional dan klien [82]. Pada awalnya, metode *agile* dikembangkan karena terdapat banyak proses pengembangan tidak dapat berhasil dengan baik sesuai dengan kebutuhan *user* [83]. Metode *agile* juga biasa disebut dengan *sprint* yang berfokus pada peningkatan dari pengembangan suatu produk atau layanan pada sebuah manajemen proyek [84].



Gambar 2.7 Tahapan *Agile Software Development*
Sumber: [85]

Berdasarkan Gambar 2.7 di atas merupakan tahapan metode pengembangan sistem dari *Agile Software Development*. Berikut adalah penjelasan dari masing-masing tahapan metode tersebut diantaranya [86]:

1. *Planning*

   Planning merupakan tahapan pertama dari metode *agile* yang berfokus dalam pembuatan sistem yang akan dikembangkan dengan cara mengumpulkan data.

2. *Implementation*

   Implementation merupakan tahapan kedua dari metode *agile* yang berfokus dalam mengimplementasi pengembagan sistem sesuai dengan desain yang sudah ada.

3. *Documentation*

   Documentation merupakan tahapan ketiga dari metode *agile* yang berfokus menerapkan dokumentasi modul dan fungsi yang sudah ada sebagai catatan informasi untuk mempermudah proses pengembangan.

4. *Deployment*

   Deployment merupakan tahapan keempat dari metode *agile* yang berfokus menyediakan sistem yang telah dibuat untuk kebutuhan *user*.

5. *Maintenance*

   Maintenance merupakan tahapan terakhir dari metode *agile* yang berfokus memelihara sistem yang telah dibuat kepada *user* bertujuan untuk mencegah *bug* yang terjadi saat pengoperasian sistem tersebut.

2.3 *Tools* yang digunakan

Berikut adalah *tools* dan *software* yang digunakan dalam mengembangkan teknik *data mining* diantaranya:

2.3.1 *Visual Studio Code*

Visual Studio Code (VS Code) merupakan sebuah teks editor yang diterbitkan oleh *Microsoft* yang bersifat *open-source* dalam mengoperasikan sistem *multi-platform* yang tersedia pada versi *Linux, Mac,* dan *Windows* [87]. Teks editor ini secara langsung mendukung bahasa pemrograman, yaitu *JavaScript, TypeScript, Node.js* dan lainnya [88].

Platform *Visual Studio Code* juga menggunakan sistem folder atau *workspace* untuk berinteraksi dengan projek yang sedang dilakukan [89].

Gambar 2.8 *Visual Studio Code*
Sumber: [90]

Berdasarkan Gambar 2.8 di atas merupakan logo dari platform *Visual Studio Code* mendukung bahasa *markup* juga yang dilengkapi tambahan *plug-ins* oleh komunitas dan sudah teraga oleh lisensi perangkat lunak yang bebas untuk digunakan [91]. Berikut pada Tabel 2.2 merupakan beberapa versi dan tahun terbitnya dari *tools Visual Studio Code*.

Tabel 2.2 Versi dan Tahun Terbit *Visual Studio Code*

Versi	Tahun Terbit
1.56-1.63	2021
1.64-1.74	2022
1.75-1.77	2023

Sumber: [92]

2.3.2 *Python*

Python adalah bahasa pemrograman tingkat tinggi yang diciptakan oleh Guido Van Rossum pada tahun 1989 di *National Research Institute*, Belanda [93]. Bahasa pemrograman ini mendukung pemrograman berbasis grafis seperti *GUI Programming* dan *Object Oriented Programming (OOP)* sehingga memiliki kelebihan berupa pengalokasian memori secara dinamis [94]. *Python* juga dapat digunakan sebagai bahasa *prototyping* utama untuk *backend web development* dan lainnya seperti pengembangan *big data* pada *data science* dan *data analysis* [95].

Gambar 2.9 *Python*
Sumber: [96]

Berdasarkan Gambar 2.9 di atas merupakan logo dari bahasa pemrograman *Python* yang memiliki berbagai macam versi. Berikut pada Tabel 2.3 adalah versi dari bahasa pemrograman *Python* dan tahun terbitnya.

Tabel 2.3 Versi dan Tahun Terbit Bahasa Pemrograman *Python*

Versi	Tahun Terbit
3.5.5-3.7.2	2018
3.7.3-3.8.1	2019
3.8.3-3.8.7	2020
3.8.8-3.10.1	2021
3.10.2-3.11.1	2022
3.11.2	2023

Sumber: [97]

2.3.3 *Tensorflow*

Tensorflow merupakan sebuah *software library* atau *framework* yang diciptakan oleh *google* untuk mengimplementasikan konsep *machine learning* dan *deep learning* secara mendalam dengan cara yang termudah [98]. *Framework Tensorflow* banyak digunakan pada berbagai bidang, salah satunya adalah *object detection* yang didalamnya terdapat *API* untuk mempermudah proses *training, constructing,* dan *deployment* pada suatu model [99]. *Tensorflow* juga memiliki arsitektur yang fleksibel sehingga dapat digunakan pada banyak platform yaitu *GPU, CPU, TPU* dan *desktop* hingga ke perangkat sistem [100].

Gambar 2.10 *Tensorflow*
Sumber: [101]

Berdasarkan Gambar 2.10 di atas merupakan logo dari *tools Tensorflow* yang memiliki berbagai macam versi. Berikut pada Tabel 2.4 merupakan versi dan tahun terbit dari *Tensorflow*.

Tabel 2.4 Versi dan Tahun Terbit *Tensorflow*

Versi	Tahun Terbit
1.5.0-1.12.0	2018
1.12.2-2.0.0	2019

Sumber: [102]

Tabel 2.5 Versi dan Tahun Terbit *Tensorflow*

Versi	Tahun Terbit
2.1.0-2.4.0	2020
2.4.1-2.8.0rc0	2021
2.8.0-2.12.0	2023

Sumber: [102]

2.3.4 *Keras*

Keras merupakan suatu *tools library machine learning* yang bersifat *open-source* yang dibentuk oleh bahasa pemrograman *Python* dan dikembangkan oleh seorang *Google Engineer* yang bernama Francois Chollet dalam membangun model *deep learning* dengan cepat hanya dari baris beberapa *code* [103]. *Tools* ini dapat digunakan sebagai *API* untuk *Tensorflow*, perangkat jaringan komputasi seperti *CNTK* dan *Theano* untuk merancang, mengevaluasi, menerapkan, dan memvisualisasikan model *deep learning* [104]. *Keras* juga dapat mendukung hampir semua permodelan *neural network* seperti *convolutional*, *pooling*, *reccurent*, *embedding*, dan sejenisnya [105].

Gambar 2.11 *Keras*
Sumber: [106]

Berdasarkan Gambar 2.11 di atas merupakan logo dari *Keras* yang dapat dijalankan pada *CPU* dan *GPU* pada perangkat keras yang menggunakan ratusan paralel untuk mempercepat pemrosesan *machine learning* [107]. Berikut pada Tabel 2.6 merupakan versi dan tahun terbit dari *tools Keras*.

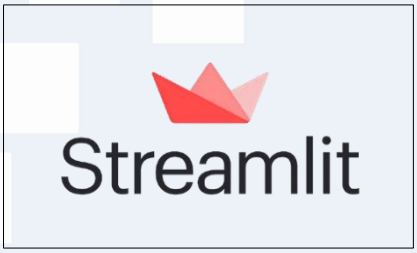
Tabel 2.6 Versi dan Tahun Terbit *Keras*

Versi	Tahun Terbit
2.1.3-2.2.4	2018
2.2.5-2.3.1	2019
2.4.0-2.4.3	2020
2.5.0rc0-2.8.0rc0	2021
2.8.0-2.11.0	2022
2.12.0rc1-2.12.0	2023

Sumber: [108]

2.3.5 *Streamlit*

Streamlit merupakan *tools framework* yang bersifat *open-source* untuk penerapan *machine learning* pada *data science* [109]. *Framework* ini banyak digunakan untuk kerangka kerja berbasis *web* dalam menyebarkan model dan visualisasi dengan tampilan yang minimalis dan cukup baik serta ramah pengguna [110]. Selain itu, *Streamlit* juga mendukung pembuatan presentasi dan kolaborasi serta penerapan berbagai opsi untuk ekspor dokumen ke dalam aplikasi data yang interaktif [111].



Gambar 2.12 *Streamlit*
Sumber: [112]

Berdasarkan Gambar 2.12 di atas merupakan logo dari *tools Streamlit* yang memiliki berbagai macam versi. Berikut pada Tabel 2.7 di bawah ini merupakan versi dan tahun terbit dari *tools Streamlit*.

Tabel 2.7 Versi dan Tahun Terbit *Streamlit*

Versi	Tahun Terbit
0.35.0-0.52.0	2019
0.53.0-1.20.0	2023

Sumber: [113]

2.4 Penelitian Terdahulu

Berikut pada Tabel 2.8 merupakan penelitian terdahulu yang menjadi acuan penelitian ini.

Tabel 2.8 Penelitian Terdahulu

Penulis	Judul	Artikel Jurnal	Metode	Data	Negara	Alat	Hasil	Penelitian Lanjut
• S Banyal • P Goel • D Grover	*Indian StockMarket Prediction Using Stacked LSTM and Multi-Layered Percepron* [9]	*Int. J. Innov. Technol. Explor. Eng.*, vol. 9, no. 3, pp. 1051–1055, 2020	• MLP • LSTM • *Timestamp* • *Activation* • *Optimizer*	*Infosys Limited* (2018 – 2019)	India	• *Intel® core ™ i7-7500U CPU* • *Anaconda* • *Python*	• *Timestamp*: 60 • *Optimizer*: *RMSProp* • *Activation*: *ReLU* • LSTM (MSE): 0.58%	• *Ensemble Model* • *Dynamic Web Scraping API*
• R Syamala • K Suresh	*Stock Price Prediction Using Deep Learning Techniques* [10]	*J. Theor. Appl. Inf. Technol.*, vol. 100, no. 1, pp. 170–183, 2022	• LR • LSTM • *Optimizer*	• *TradingView* • S&P500 & AEX (2016-2021)	Belanda	• *Python* • *Keras* • *Tensorflow*	1. S&P500: • *Optimizer*: *Adam* • LSTM (RMSE): 0.0223 • LSTM (R2): 0.8849 2. AEX • *Optimizer*: *AdaMax* • LSTM (RMSE): 0.0464 • LSTM (R2): 0.6373	• *Stacked LSTM* • *Varied Input Quantiles* • *Hidden Layers*
• S Hansun • J Young	*Predicting* LQ45 *Financial Sector Indices using* RNN-LSTM [114]	*J. Big Data*, vol. 8, no. 1, 2021	• RNN • LSTM	• *Indonesia Stock Exchange* (IDX) • LQ45 *Financial Sector* (2016 – 2019)	Indonesia	• *Intel Core i3-8130U CPU @ 2.20 GHz (4 CPUs) processor* • *Anaconda* • *Python*	• BMRI • RMSE: 739.1828 • MAPE: 18.6135	• *Weighted Exponential Moving Average* • *Double Exponential Smoothing*

Tabel 2.9 Penelitian Terdahulu

Penulis	Judul	Artikel Jurnal	Metode	Data	Negara	Alat	Hasil	Penelitian Lanjut
• S Hansun • A Suryadibrata	*Gold Price Prediction in* COVID-19 *Era* [115]	*Int. J. Comput. Intell. Control*, vol. 13, no. 2, pp. 29–34, 2021	LSTM	• *Yahoo Finance* • *Gold Price* (May 15 2021)	Indonesia	• *Python* • *Keras* • *Tensorflow*	• R-Squared: 97.242% • MAPE: 17.66144 • RMSE: 39.94162	• ANN • SVR • *Bi-directional* LSTM
• NR Shin • DY Yun • CG Hwang	*Artificial neural network algorithm comparison for exchange rate prediction* [116]	*Int. J. Int. Broad. Comm.* vol. 12, no. 3, pp. 125–130, 2020	• LSTM • *Activation* • *Optimizer*	*Bank of Korea's Economic Statistics System* (2000-2020)	Korea	*Keras*	• *Activation*: *Tanh* • *Optimizer*: *Adam* • MAE: 0.009	• *Number of Neurons* • *Dropout* • *Early Stopping* • *Reducing Weights of Bias*
• Z Berradi • M Lazaar • H Omara • O Mahboub	*Effect of Architecture in Reccurent Neural Network Applied on The Prediction of Stock Price* [117]	*IAENG Int. J. Comput. Sci.*, vol. 47, no. 3, pp. 436–441, 2020	• ERNN • LSTM • GRU • *Optimizer*	• *Yahoo Finance* • IAM.PA & ORA.PA (2017-2019)	Maroko	-	1. *Train data*: • *Optimizer*: *Adam* • ERNN (MSE): 0.013381 2. *Test data*: • *Optimizer*: *Adam* • ERNN (MSE): 0.008851	-
• M Mohsin • MI Babar • M Hamza • M Jehanzeb • S Habib • MS Khan	*Industrial Financial Forecasting using Long Short-Term Memory Recurrent Neural Networks* [118]	*Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 4, pp. 88–99, 2019	• LSTM • *Neurons* • *Epochs* • *Learning Rate* • *Activation* • *Optimizer*	• *Pakistan GDP Expenditure* (1970-2016) • *Economic Statistics Branch of the United Nations Statistics*	Pakistan	• *Python* • *Keras*	• *Activation*: *SELU* • *Optimizer*: *Adamax* • *Learning Rate*: 0.002 • *Epochs*: 3000 • *Neurons*: 10 • *Accuracy Score*: 92.356115%	-

Tabel 2.10 Penelitian Terdahulu

Penulis	Judul	Artikel Jurnal	Metode	Data	Negara	Alat	Hasil	Penelitian Lanjut
• SH Faru • A Watitu • L Nderu	*A Hybrid Neural Network Model Based on Transfer Learning for Forecasting Forex Market* [119]	*J. Data Analysis. Info Proc.* no. April 2013, pp. 103–120, 2023	• RNN • LSTM • RNN-LSTM • *Streamlit*	• *Yahoo Finance* • GBP/USD • EUR/USD • USD/JPY • USD/ZAR • AUD/NZD	Kenya	• *Core® i7-7600U CPU@2.80GHz* • *Intel® HD Graphics 620* • *Python* • *Tensorflow*	• EUR/USD - RMSE: 0.003796 - MAE: 0.003054 - MAPE: 0.003117	• *Tuning Hyperparameter* • *Optimization Algorithms* • *Loss Functions*
• J John • S Sonawne • S Mankar • D Wasu • P Rachchawar • G Bhoyar	*Online Web Scraping to Anticipate The Stock Market Prediction* [120]	*Int. J. Comp. Sci. Tech.* vol. 10, no. 2, pp. 13–18, 2022	• LSTM • ARIMA • *Streamlit*	• *Yahoo Finance* • Tesla Inc. (2010-2019)	India	• *Python* • *Keras* • *Tensorflow* • *Pandas* • *Numpy* • *Pandas Datareader* • *Matplotlib*	*Accuracy Score*: >80%	• *Cryptocurrency Trading App* • *Sentiment Analysis*
• F Kamalov • L Smail • I Gurrib	*Stock Price Forecast With Deep Learning* [121]	*2020 Int. Conf. Decis. Aid Sci. Appl. DASA 2020*, pp. 1098–1102, 2020	• LSTM • RNN • CNN • *Optimizer* • *Batch Size*	S&P500 Index	Dubai	• Keras • Google Colaboratory	• *Optimizer* : *RMSProp* • *Batch Size*: 32 • *Validation* (MAE): 0.0178 • *Test* (MAE): 0.0260	*Automated Stock Trading Platforms*

Tabel 2.11 Penelitian Terdahulu

Penulis	Judul	Artikel Jurnal	Metode	Data	Negara	Alat	Hasil	Penelitian Lanjut
• J Lv • C Wang • W Gao • G Zhao	*An Economic Forecasting Method Based on the LightGBM-Optimized LSTM and Time-Series Model* [122]	*Comput. Intell. Neurosci.*, vol. 2021, 2021	• LightGBM-LSTM • RNN • GRU • *Activation* • *Optimizer*	CSI 300 Index (2005-2020)	China	• *Keras* • *Tensorflow*	1. *Training Error*: • *Activation*: *SoftSign* • *Optimizer*: *RMSProp* • RMSE: 0.01108 • MAE: 0.00770 2. *Test Error*: • *Activation*: *SoftSign* • *Optimizer*: *RMSProp* • RMSE: 0.01076 • MAE: 0.00651	• GRU • *Training Speed*
• A Fazeli • S Houghten	*Deep Learning for The Prediction of Stock Market Trends* [123]	*Proc. – 2019 IEEE Int. Conf. Big Data, Big Data 2019*, pp. 5513– 5521, 2019	• LSTM • *Dropout* • *Optimizer*	• *Yahoo Finance* • S&P500 & Apple.Inc (2014-2019)	Kanada	• *Keras* • *Python* • *Tensorflow* • *CNTK* • *Theano* • *Pandas* • *Numpy* • *Scikit-Learn*	• *Optimizer: Nadam* • *Dropout*: 0.1 • MSE: 0.004845492 • *Huberloss*: 0.004993705	• *Effect of different input data* • *Sentiment Analysis*
• Kerchev • T George	*Stocktrading with machine learning models Authors : Gil Moes & Mirza Muharemovic* [124]	*Universite du Luxembourg*, 2021	• RNN • LSTM • *Activation* • *Layers* • *Learning Rate* • *Batch Size* • *Epochs* • *Seq Len*	• *Yahoo Finance* • AMD (2019)	Perancis	• *Keras* • *Python* • *Tensorflow* • *Pandas* • *Numpy* • *Scikit-Learn*	• *Seq Len*: 20 • *Neurons*: 200 • *Layers*: 2 • *Learning Rate*: 0.001 • *Batch Size*: 50 • *Epochs*: 100 • *Activation*: *ReLU* • *Fg (20,21)*	-

Tabel 2.12 Penelitian Terdahulu

Penulis	Judul	Artikel Jurnal	Metode	Data	Negara	Alat	Hasil	Penelitian Lanjut
• M Rana • MM Uddin • MM Hoque	*Effects of Activation Functions and Optimizers on Stock Price Prediction using LSTM Reccurent Neural Network* [8]	*ACM Int. Conf. Proceeding Ser.*, pp. 354–358	• LR • SVR • LSTM • *Activation* • *Optimizer*	Saham Acciona (2008-2018)	China	-	1. *Activation*: • *Linear* • *Tanh* 2. *Optimizer* : • *AdaMax* • *Adam* 3. LSTM (RMSE): 0.0151	• GRU • CNN
A Olof	*A comparative study between algorithms for time series forecasting on customer prediction* [125]	p. 58, 2019	• TCN • ARIMA • RNN • LSTM • HMM • CRISP-DM	• *Google Trends* • SMHI & Lantmet (2010-2017)	Jerman	• *Python* • *Keras* • *Statsmodels*	• TCN • MAE: 1.229,5 • RMSE: 1.489,5 • MAPE: 42%	• *Hyperparameter*: - *Grid Search* - *Bayesian Search*
M Mashtura	*Stock Price Prediction Using Time Series Model* [126]	*Brac Univ.*, vol. 1(1), no. August, pp. 1–51, 2019	• ARIMA • PROPHET • LSTM • CRISP-DM	• *Yahoo Finance* • Saham Perbankan (1991-2003)	Irlandia	• *R* • *Keras*	• LSTM • RMSE: 3.184659373	• *Seasonality* • *Traditional Statistical* • *back-testing technique*

Berdasarkan Tabel 2.8 di atas merupakan 16 penelitian terdahulu yang menjadi acuan dalam penelitian ini. Beberapa penelitian terdahulu yang telah dilakukan tentunya memiliki keterkaitan dengan penelitian yang akan dilakukan. Pada penelitian [8-124] akan menjadi acuan dalam memilih permodelan algoritma *Long Short-Term Memory* dengan menggunakan *hyperparameter activation* yaitu *linear, relu, sigmoid,* dan *tanh* serta *optimizer adam, adagrad, nadam, rmsprop, adadelta, sgd,* dan *adamax*. Selain itu, pada penelitian [125-126] akan menjadi acuan dalam memilih *framework data mining* yaitu CRISP-DM (*Cross Industry Standard Process for Data Mining*) dalam penelitian. Penggunaan *streamlit* akan menjadi acuan dalam penelitian ini dalam mengembangkan permodelan algoritma LSTM pada sistem informasi berbasis *website* seperti pada peneltian [119-120]. Berdasarkan beberapa penelitian yang telah dilakukan, permodelan algoritma *Long Short-Term Memory* (LSTM) hanya menggunakan beberapa *hyperparameter activation* seperti *linear, relu, sigmoid,* atau *tanh* serta *optimizer adam, adagrad, nadam, rmsprop, adadelta, sgd,* atau *adamax* saja. Selain itu, dalam permodelan hanya menggunakan beberapa evaluasi metrik seperti *Mean Absolute Error* (MAE), *Mean Squared Error* (MSE), R2, *Root Mean Squared Error* (RMSE), atau *Mean Absolute Percentage Error* (MAPE) serta penggunaan *streamlit* juga tanpa berdasarkan permodelan algoritma LSTM dengan *hyperparameter activation* dan *optimizer*. Dataset yang digunakan juga adalah saham perbankan dan indeks saham dari luar negeri seperti S&P500, *AEX, Tesla .Inc,* CSI 300, dan *Apple .Inc*.

Hal ini maka penelitian yang akan dilakukan memiliki perbedaan yaitu menggunakan 6 dataset transportasi di Indonesia. Metode atau *framework data mining* yang digunakan adalah CRISP-DM dalam mengimplementasikan permodelan algoritma *Long Short-Term Memory* (LSTM) konvensional dengan model LSTM dari komparasi perbandingan *hyperparameter* seluruh *activation* yaitu *linear, relu, sigmoid,* dan *tanh*, serta seluruh *optimizer* yaitu *adam, adagrad, nadam, rmsprop, adadelta, sgd,* dan *adamax*. Penggunaan penjabaran evaluasi metrik yang dipilih adalah MAPE, MAE, RMSE, MSE, R2, dan *elapsed time* dalam mengukur *training model* serta *statistical test* dari *shapiro-wilk test*.