Prediksi Poin Fantasy Premier League Berdasarkan Posisi Pemain Berbasis Deep Learning

Anas Satria Lombu
209170337

Tesis diajukan sebagai syarat untuk meraih gelar Magister Komputer
Konsentrasi Sains Data
Program Studi Informatika Program Magister
Fakultas Teknologi Industri
Universitas Islam Indonesia
2024

Lembar Pengesahan Pembimbing

Prediksi Poin Fantasy Premier League Berdasarkan Posisi Pemain Berbasis Deep Learning

Anas Satria Lombu
20917037

Yogyakarta, 21 Maret 2024

Pembimbing	Pembimbing
Irving Vitra Paputungan, S.T., M.Sc., Ph.D	Chandra Kusuma Dewa, S.Kom., M.Cs., Ph.D

Lembar Pengesahan Penguji

Prediksi Poin Fantasy Premier League Berdasarkan Posisi Berbasis Deep Learning

Anas Satria Lombu
20917037

Yogyakarta, Juni 2024

Tim Penguji,

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

Chandra Kusuma Dewa, S.Kom., M.Cs., Ph.D
Anggota I

Dhomas Hatta Fadholi, S.T., M.Eng., Ph.D
Anggota II

Mengetahui,
Ketua Program Studi Informatika Program Magister
Universitas Islam Indonesia

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

Abstrak

Prediksi Poin Fantasy Premier League Berdasarkan Posisi Berbasis Deep Learning

Fantasy Premier League (FPL) merupakan game berbasis olahraga fantasi yang berfokus pada sepakbola khususnya Liga Primer Inggris. Setiap manajer dalam permainan ini diberikan kesempatan untuk membangun tim virtual selama satu musim. Sebuah tim virtual terdiri dari berbagai posisi pemain yang akan mendapatkan poin FPL berdasarkan performa mereka dalam dunia nyata. Penelitian ini bertujuan untuk mengimplementasikan algoritma deep learning untuk memprediksi total poin FPL yang dihasilkan oleh pemain berdasarkan performa mereka pada 5 pertandingan terakhir. Model prediksi dirancang dengan menggunakan algoritma Convolutional Neural Network yang terdiri dari lapisan konvolusi satu dimensi, maxpooling, dan dense. Selain itu, Algoritma Long Short-Term Memory (LSTM) dengan lapisan LSTM dan Dense berjumlah 64 unit ditambahkan sebagai perbandingan model untuk mengetahui performa model deep learning terbaik pada penelitian ini. Pada skenario pertama, rasio data 70:30 diperoleh nilai rata-rata dari 4 posisi pemain dengan Mean Squared Error (MSE) menggunakan CNN ialah 0.0052 dan 0.0027 untuk LSTM. Sementara itu, pada skenario kedua dengan rasio data 80:20, diperoleh hasil evaluasi adalah 0.0027 untuk CNN dan LSTM sebesar 0.0025. Hasil evaluasi model menunjukkan bahwa algoritma LSTM, dengan memanfaatkan tiga pintu pada arsitektur model, lebih unggul dalam mengenali urutan data historis dibandingkan dengan algoritma CNN.

Kata kunci

fantasy premier league, convolutional neural network, long short term memory

Abstract

Predicting Fantasy Premier League based on Position Using Deep Learning

Fantasy Premier League is a fantasy sports-based game focused on football, particularly the English Premier League. Each manager in this game is given the opportunity to build a virtual team for single season. The virtual team consists of various player positions that will earn FPL points based on their real-world performance. This research aims to implement a deep learning algorithm to predict FPL points generated by players based on their performance in the last 5 matches. The prediction model is designed using a Convolutional Neural Network (CNN) algorithm consisting of one dimensional covolution layers, maxpooling, and dense layers. Additionally, a Long Short-Term Memory (LSTM) algorithm with LSTM and dense layers totaling 64 units is added as a comparison model to determine the best-performing deep learning model in this study. In the first scenario, with a 70:30 data ratio, the average value of 4 player positions with Mean Squared Error (MSE) using CNN is 0.0052 and 0.0027 for LSTM. Meanwhile, in the second scenario with an 80:20 data ratio, the evaluation results are 0.0027 for CNN and 0.0022 for LSTM. Model evaluation results indicate that the LSTM algorithm, leveraging three gates in the model architecture, is superior in recognizing historical data sequences compared to the CNN algorithm.

Keywords

fantasy premier league, convolutional neural network, long short-term memory

Pernyataan Keaslian Tulisan

Dengan ini saya menyatakan bahwa tesis ini merupakan tulisan asli dari penulis, dan tidak berisi material yang telah diterbitkan sebelumnya atau tulisan dari penulis lain terkecuali referensi atas material tersebut telah disebutkan dalam tesis. Apabila ada kontribusi dari penulis lain dalam tesis ini, maka penulis lain tersebut secara eksplisit telah disebutkan dalam tesis ini.

Dengan ini saya juga menyatakan bahwa segala kontribusi dari pihak lain terhadap tesis ini, termasuk bantuan analisis statistik, desain survei, analisis data, prosedur teknis yang bersifat signifikan, dan segala bentuk aktivitas penelitian yang dipergunakan atau dilaporkan dalam tesis ini telah secara eksplisit disebutkan dalam tesis ini.

Segala bentuk hak cipta yang terdapat dalam material dokumen tesis ini berada dalam kepemilikan pemilik hak cipta masing-masing. Apabila dibutuhkan, penulis juga telah mendapatkan izin dari pemilik hak cipta untuk menggunakan ulang materialnya dalam tesis ini.

Yogyakarta, Februari 2024

Anas Satria Lombu, S.Kom

Daftar Publikasi

Publikasi berikut menjadi bagian dari Bab 3

Sitasi publikasi 1

Kontributor	Jenis Kontribusi
Anas Satria Lombu	Melakukan komputasi dan analisis model Menulis Paper
Irving Vitra Paputungan	Memberikan ide dan gagasan Melakukan pengecekan dan memberikan masukan pada paper
Chandra Kusuma Dewa	Melakukan perancangan model Melakukan pengecekan dan memberikan masukan pada paper

Halaman Kontribusi

Saya ingin menyampaikan terima kasih kepada dosen pembimbing I dan II yang telah memberikan kontribusi besar dalam penelitian dan penulisan tesis ini.

Halaman Persembahan

Dengan mengucapkan syukur Alhamdulillah, penulis mempersembahkan Tesis ini kepada :

1. Bapak Ahmat Lombu dan Ibunda Fajar Wahyuni atas pengorbanan mereka selama ini dan senantiasa memanjatkan doanya yang tak henti-hentinya sehingga penulis dapat menyelesaikan Tesis ini.
2. Adik Azka dan Dimas yang selalu memberikan dukungan agar Tesis ini dapat terselesaikan.

Kata Pengantar

Puji syukur penulis ucapkan kehadirat Allah Subhanhu WaTa'ala yang telah melimpahkan rahmat, hidayah, dan karunia-Nya, sehingga penulis dapat menyelesaikan Tesis yang berjudul "Prediksi Fantasy Premier League Points Berdasarkan Posisi Pemain Berbasis Deep Learnin". Tak lupa shalawat dan salam kami haturkan kepada junjungan kita Nabi Muhammad Sallalahu 'Alaihi Wassallam, yang telah membawa kita dari zaman jahiliyah menuju zaman terang benderang.

Tesis ini dibuat sebagai salah satu syarat yang harus dipenuhi untuk memperoleh gelar Magister Informatika pada Konsentrasi Sains Data di Program Studi Informatika Program Magister, Universitas Islam Indonesia. Selain itu, Tesis ini juga penulis dapat mengimplementasikan ilmu dan pengetahuan yang telah diperoleh selama berada di bangku perkuliahan.

Selama proses pengerjaan tesis ini, penulis menyadari mengalami kesulitan yang telah dilalui dan adanya bantuan, dukungan, bimbingan serta doa dari berbagai pihak. Untuk itu dengan segala kerendahan hati izinkanlah penulis menyampaikan rasa terima kasih dan penghargaan tersebut kepada:

1. Kedua Orang Tua Bapak Ahmat Lombu dan Ibu Fajar Wahyuni juga Adik Dimas Rifa'i Lombu dan Azka Faris Lombu, serta keluarga penulis yang telah mendoakan dan memberikan dukungan selama penulis menempuh studi di Universitas Islam Indonesia.
2. Bapak Fathul Wahid, S.T., M.Sc., Ph.D. selaku Rektor Universitas Islam Indonesia.
3. Bapak Prof. Dr. Ir. Hari Purnomo, M.T. selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Bapak Irving Vitra Paputungan, S.T., M.Sv., Ph.D. selaku Ketua Program Studi Informatika Program Magister dan Pembimbing Utama penulis, yang telah meluangkan waktunya dan dengan penuh perhatian memberikan bimbingan serta motivasi yang positif sehingga tesis ini dapat terselesaikan.
5. Bapak Chandra Kusuma Dewa, S.Kom., M.Cs., Ph.D. selaku Pembimbing Kedua penulis, yang juga telah meluangkan waktunya dan dengan penuh perhatian memberikan bimbingan serta motivasi yang positif sehingga tesis ini dapat terselesaikan.

6. Seluruh dosen di Program Studi Informatika Program Magister Fakultas Teknologi Industri Universitas Islam Indonesia.
7. Rekan-rekan Angkatan di Program Studi Informatika Program Magister konsentrasi Sains Data.
8. Semua pihak yang tidak dapat penulis sebutkan satu persatu yang telah membantu dalam penyelesaian tesis ini.

Semoga segala bimbingan, motivasi, dan dukungannya mendapat imbalan dari Allah SWT. Penulis menyadari bahwa laporan tesis ini masih banyak kekurangan, oleh karena itu, penulis mengharapkan masukan yang membangun dan saran serta kritik yang positif sehingga laporan ini menjadi lebih bermanfaat untuk semua pihak.

Daftar Isi

Daftar Tabel

Daftar Gambar

Glosarium

FPL	- Fantasy Premier League
EPL	- English Premier League
CNN	- Convolutional Neural Network
LSTM	- Long Short-Term Memory
MSE	- Mean Squared Error

BAB 1
Pendahuluan

1.1 Latar Belakang

Olahraga memiliki peran penting dalam budaya manusia di seluruh dunia. beberapa jenis olahraga sangat digemari, seperti sepakbola yang memiliki daya tarik terhadap lebih dari 40% penduduk di dunia menjadikannya sebagai olahraga paling popular setelah basket (Nugroho, 2021). Liga Primer Inggris, lebih dikenal dengan istilah *English Primer League* merupakan salah satu kompetisi sepakbola kasta tertinggi di dunia yang memiliki jumlah penonton melalui kanal televisi mencapai 4,7 juta orang (Murni et al., 2023). Federasi sepakbola liga inggris, selaku pihak penyelenggara kompetisi menghadirkan inovasi terbaru yang memanfaatkan teknologi pada sepakbola.

Fantasy Sport merupakan sebuah fenomena baru berupa aktivitas daring dalam perkembangan mengalami peningkatan signifikan selama beberapa dekade terakhir. Pada tahun 1950, fantasy sport mulai diperkenalkan oleh *Wilfred Winkenbach* pada olahraga golf (Kristiansen et al., 2018). Sebuah tim berisi pemain golf profesional yang dipilih oleh peserta, kemudian peserta yang memiliki poin terbaik dinobatkan sebagai juara pada kompetisi. perkembangan *fantasy sport* telah meliputi olahraga lain seperti sepakbola, basket, hockey, baseball, dan rugby. dalam riset yang dilakukan pada tahun 2023 oleh *Fantasy Sport & Gaming Association*, menunjukkan bahwa *fantasy sport* bertema sepakbola mencapai 79% partisipan (Association, 2023). Jumlah partisipan terbanyak diantara *fantasy sport* lainnya berdasarkan hasil survei.

Fantasy Premier League (FPL) merupakan salah satu *fantasy sport* yang berfokus pada sepakbola, dimana mengizinkan para penggemar *English Primer League* (EPL) untuk menjadi seorang manajer yang dapat mengelola sebuah tim secara virtual. FPL berada di bawah naungan Federasi Sepakbola Inggris, melibatkan 20 klub dan pemain-pemain profesional yang berasal dari penjuru dunia. semenjak FPL rilis pada musim 2002/2003, pertumbuhan partisipan menunjukkan tren positif tiap tahunnya. Selama 20 tahun terakhir, tercatat lebih dari 36 juta orang di seluruh dunia merasakan atmosfer permainan FPL untuk menguji kemampuan setiap individu dalam manajemen tim (Masters, 2022a). Para manajer dapat bersaing secara kompetitif dengan manajer lainnya secara publik tanpa batasan wilayah. Terbukti bahwa 79% manajer berada di luar wilayah UK. Meskipun saling

berjauhan, FPL membawa penggemar lebih dekat dengan olahraga favorit mereka, memungkinkan interaksi yang lebih intensif antara penggemar dari berbagai belahan bumi. Selain itu, komunitas baru tercipta melalui liga pribadi yang dimainkan bersama teman, anggota keluarga, dan kolega kantor sebagai hiburan yang dapat menyeimbangkan kesehatan mental dari kegiatan atau aktivitas sepanjang hari.

Pencapaian tertinggi diraih pada musim 2022/2023, jumlah manajer mencapai 11,4 juta terdaftar dalam satu musim. Pertumbuhan mengalami peningkatan sebesar 25% dari musim sebelumnya yang hanya mencapai 9,1 juta manajer (Masters, 2022b). Analisis data melalui *Drone Emprit Academic* berupa *Social Network Analisis* menggunakan *Artificial Intelligence* dan *Natural Processing Language* terkait FPL pada media sosial X mengemukakan bahwa cuitan terkait FPL paling banyak ditemui di benua Asia, diikuti Afrika, Eropa Amerika Utara, Amerika Selatan dan Oseania (Kartiko, 2020). Indonesia menjadi salah satu negara pengirim cuitan terbanyak di benua Asia, terutama pada hari Sabtu, Minggu dan Senin. Hal ini dikarenakan pertandingan FPL dimainkan pada akhir pekan. Produktivitas dimulai pada pukul 18.30 WIB hingga pukul 00.00 WIB yang menjadi puncak tertinggi karena seluruh pertandingan telah terlaksana sehingga para manajer melakukan evaluasi tim FPL secara pribadi.

Secara teknis permainan, seorang manajer akan menerima anggaran secara virtual untuk memilih pemain yang dibatasi berdasarkan aturan permainan. Manajer juga akan menerima poin fantasi dari performa pemain profesional dalam dunia nyata setiap minggunya. Sebagai contoh untuk menghasilkan poin, seperti cara mencetak gol, memberikan umpan yang menghasilkan gol, tanpa kebobolan, dan durasi pemain dalam pertandingan. secara objektif, tidak hanya dapat menghasilkan poin tetapi perlu meningkatkan produktivitas poin setiap minggunya secara regular dari tim virtual yang terdiri dari 15 pemain meliputi 11 pemain utama dan 4 pemain cadangan (Khamsan & Maskat, 2019).

Fantasy Premier League telah memfasilitasi manajer dengan menyediakan sekumpulan data statistik yang telah disajikan pada sistem. Data statistik dapat dijadikan sebagai informasi penting dalam pengambilan keputusan untuk memilih pemain berdasarkan performa yang dapat berkontribusi sebagai poin fantasi. Beragam sudut pandang yang tercipta dari hasil pemikiran setiap individu manajer akan menimbulkan kompetisi antar manajer. Namun, tidak sedikit para manajer hanya mengandalkan sentimen dan perasaan pribadi berupa kecintaannya pada salah satu klub atau pemain favorit yang dapat mempengaruhi pemilihan pemain yang terdaftar pada tim virtual.

Kecerdasan Buatan *(Artificial Intelligence)* adalah kapasitas suatu komputer atau robot yang dioperasikan oleh komputer untuk melakukan aktivitas yang melibatkan kecerdasan dan kearifan manusia. Seiring kemajuan teknologi, penerapan salah satu cabang kecerdasan buatan yaitu *deep learning*, dapat menjadi solusi dalam perancangan model prediksi untuk menyelesaikan permasalahan manusia. Algoritma *Long Short-Term Memory* (LSTM) menghasilkan nilai *test-loss* lebih rendah dibandingkan dengan algoritma *Support Vector Machine* dan *Multi-Layer Perceptron* untuk memprediksi performa pemain pada pertandingan selanjutnya di *English Premier League*. model arsitektur LSTM memiliki 3 buah pintu dinilai dapat mempertahankan dan mengenali urutan data dengan baik (Lindberg & Söderberg, 2020). Pemanfaatan *deep learning* lainnya menggunakan algoritma *Convolutional Neural Network* (CNN) pada permainan FPL, memanfaatkan konvolusi 1 dimensi yang dapat memahami data berdasarkan waktu. *Ramdas* perlu mengubah informasi statistik ke dalam sebuah matriks agar dapat dilakukan ekstrasi fitur seperti yang dilakukan pada pemrosesan gambar dan video (Ramdas, 2022).

Berdasarkan uraian diatas, peneliti akan merancang dan membandingkan algoritma CNN dan LSTM pada model *deep learning* yang digunakan untuk memprediksi poin game Fantasy Premier League. Perancangan model dibedakan berdasarkan posisi pemain dan penentuan deret waktu. Model akan menggunakan data historis 5 pertandingan sebelumnya yang dijadikan sebagai *timesteps*. Penggunaan epoch sebanyak 50 diimplementasikan pada lapisan LSTM, Conv1D dan dense yang memiliki 64 *unit* setiap lapisan. Model regresi *Lasso* bertujuan untuk melakukan seleksi fitur pada dataset agar mengurangi dimensi data. Diharapkan penerapan *deep learning* dapat membantu manajer memperoleh poin maksimal di setiap pertandingan pada Fantasy Premier League.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan diatas, maka diperoleh beberapa rumusan masalah sebagai berikut:

1. Bagaimana cara merancang model *deep learning* menggunakan algoritma *Convolutional Neural Network* pada game Fantasy Premier League?
2. Bagaimana cara merancang model *deep learning* menggunakan algoritma Long Short-Term Memory pada game Fantasy Premier League?
3. Algoritma *deep learning* manakah yang memiliki performa model terbaik untuk memprediksi poin berdasarkan 5 data historis pertandingan sebelumnya pada game Fantasy Premier League?

1.3 Batasan Masalah

Batasan masalah diperlukan untuk menghindari meluasnya ruang lingkup yang akan dibahas dalam penelitian sehingga tidak menyimpang dari pokok permasalahan. Batasan masalah dalam penelitian ini sebagai berikut:

1. Data diperoleh dari repository github yang berisi data statistik pemain FPL pada musim 2021/2022.
2. Cedera pemain, kondisi cuaca, *budget* FPL dan kompetisi selain EPL merupakan faktor eksternal penelitian.
3. Implementasi *deep learning* menggunakan algoritma *Convolutional Neural Network* dan *Long Short-Term Memory*.
4. Penggunaan 5 data historis pertandingan sebelumnya untuk memprediksi total poin pada pertandingan selanjutnya yaitu *gameweek* ke-6.
5. Hasil penelitian berupa evaluasi nilai *test-loss* yang merepresentasikan performa model.

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah melakukan komparasi model prediksi menggunakan algoritma *deep learning* berdasarkan posisi pemain untuk mengetahui algoritma terbaik pada studi kasus Fantasy Premier League.

1.5 Manfaat Penelitian

Manfaat penelitian ini sebagai berikut:

1. Memberikan solusi dalam permasalahan pengolahan informasi data statistik pemain pada game Fantasy Premier League berbasis *deep learning*.
2. Menghasilkan model prediksi pemain, dimana pada game Fantasy Premier League pemilihan pemain dibedakan menjadi 4 bagian berdasarkan posisi pemain.
3. Memberikan wawasan terhadap penerapan algoritma *deep learning* pada studi kasus game Fantasy Premier League yang dapat dijadikan sebagai referensi untuk penelitian lainnya.

BAB 2
Tinjauan Pustaka

2.1 Penelitian Terdahulu

Diperlukan pengetahuan mengenai hasil penelitian sebelumnya agar dapat mencegah duplikasi. Beberapa penelitian sebelumnya telah menggunakan berbagai metode untuk menangani studi kasus pada game Fantasy Premier League dapat dilihat pada tabel 2.1

Akhil Gupta (2017) melakukan penelitian *Hybrid Predicted Point* antara *Autoregressive Integrated Moving Average* (ARIMA) dan *Long Short-Term Memory* (LSTM) untuk memprediksi *time series* (Gupta, 2017). Penggabungan metode bertujuan untuk menghasilkan model prediksi yang memiliki tingkat akurasi yang optimal. Kedua metode berbeda ini dinilai dapat saling menutupi kekurangan pada masing-masing metode, dimana ARIMA memiliki Batasan nilai dalam mengasumsikan nilai linear dan LSTM dapat mempertahankan nilai non-linear dalam jumlah data yang besar. Sumber data diperoleh dari website resmi FPL. hasil evaluasi yang dilakukan menggunakan *Root Mean Square Error* (RMSE) terbaik dihasilkan oleh pemain bernama Vardy ialah 2.808 pada proporsi optimal 40% ARIMA dan 60% LSTM.

Menurut Khamsam et al (2019), Perubahan nilai harga pemain yang fluktuatif mempengaruhi anggaran para manajer dalam memilih pemain yang diinginkan ke dalam tim virtual. Setiap pemain yang mengalami kenaikan nilai harga didukung oleh performa bagus ditampilkan pada pertandingan-pertandingan sebelumnya. Aktivitas transfer pemain yang dilakukan oleh seluruh manajer pada permainan juga memberikan pengaruh terhadap perubahan nilai harga secara virtual. Implementasi teknik *Synthetic Minority Over-sampling Technique* (SMOTE) menggunakan algoritma *K-Nearest Neighbour* (K-NN) berfungsi untuk menyeimbangkan distribusi kelas label yang lebih dominan di salah satu kelas. kelas “Price Fall” dan “Price Rise” sebagai kelas minoritas akan memiliki jumlah yang setara dengan kelas “Price Remain Unchanged” yang lebih dominan, menggunakan pendekatan data buatan (Khamsan & Maskat, 2019). Model prediksi yang dibangun memberikan dapat membantu manajer FPL dalam melakukan transaksi yang diperlukan seperti memperoleh kenaikan harga dan menjual calon pemain virtual, sebelum perubahan harga sebenarnya terjadi. Nilai standard deviasi terbaik diperoleh menggunakan kombinasi pembagian data *train test stratified* dan metode SMOTE upsampling skenario kedua pada penelitian sebesar 5.7873.

Menurut Bonello et al (2019), memanfaatkan data statistik yang bersumber dari FPL API telah menghasilkan total 2314 poin dengan rata-rata 36 poin per minggu pertandingan. Penggunaan data historis statistik lebih unggul dibandingkan data bursa taruhan yang didapatkan dengan cara scraping data, hanya menghasilkan 1994 poin dengan rata-rata 52 poin untuk musim 2018/2019 menggunakan algoritma *Gradient Boosting Machine* (Bonello et al., 2019). Pemain berposisi sebagai penjaga gawang memiliki nilai evaluasi model sebesar 82%, lebih rendah rendah dibandingkan model berposisi penyerang yakni 88%, karena cenderung sangat konsisten dan bergantung kepada performa keseluruhan pemain. terdapat 5 fitur penting pada penjaga gawang meliputi *influence, ict_index, minutes, value,* dan *transfer balance.*

Daily Fantasy Sport (DFS) merupakan salah satu game *fantasy sport* berfokus pada olahraga rugby. Amerika Serikat merupakan negara popular olahraga rugby menjadikan negara partisipan terbanyak pada game ini. pendekatan antara pembelajaran mesin dan manusia dilakukan untuk mendapatkan poin fantasi yang maksimal. sebuah tim virtual akan menerima poin melalui pemain-pemain yang berdasarkan posisi pemain. Tujuan penelitian dalam menyusun tim virtual terbaik memperoleh bahwa pemain pada posisi *Tigh-end* (TE), *Kickers* dan *Defence* (DEF) menghasilkan nilai *Root Mean Square Error* (RMSE) sebesar 4,10, 3,85, dan 3,5 (Beal et al., 2020). Nilai evaluasi yang lebih rendah jika dibandingkan dengan posisi *Quarterback* (QB) dan *Running Back* (RB) sebesar 6,89 dan 7,09. Berdasarkan hasil pengujian menggunakan data *National Football League* (NFL) musim 2014-2017, poin fantasi mengalami peningkatan sebesar 8,13% di setiap minggu disebabkan oleh pendekatan mesin.

Darmastyo Bagas Prabowo (2020), melakukan pembagian data menjadi beberapa skenario diantaranya 75% data latih dan 25% data uji untuk skenario pertama, 80% data latih dan 20% data uji untuk skenario kedua, sedangkan skenario ketiga berisi 90% data latih dan 10% data uji (Prabowo, 2020). Penelitian ini juga melakukan komparasi model *machine learning K-Nearest Neighbors* dan *Naïve Bayes* untuk memprediksi hasil pertandingan sepakbola pada *English Premier League*. Diperoleh hasil terbaik dari komparasi model dengan menggunakan *Naïve Bayes* pada skenario ke-3 dengan nilai akurasi sebesar 63,59%.

Lindberg and D. Soderberg (2020), penelitian yang dilakukan menyediakan model regresi dan klasifikasi berdasarkan lima pertandingan sebelumnya untuk menyajikan analisis perbandingan dari ketiga metode berbeda diantaranya *Support Vector Machine* (SVM), *Multi-Layer Perceptron* (MLP) dan *Long Short-Term Memory* (LSTM) (Lindberg

& Söderberg, 2020). Algoritma LSTM akan menggunakan data *time series* dengan jumlah *timesteps* sebanyak 1, 3 dan 5. Hubungan antar fitur dapat dikenali melalui matrik korelasi. Teknik pembagian data terbagi menjadi 3 bagian yaitu 60% data latih, 20% data validasi dan 20% data uji. Model regresi bertujuan untuk memprediksi besaran poin yang dihasilkan setiap pemain sedangkan klasifikasi terhadap pemain yang bermain secara optimal di setiap pertandingan. MLP dan LSTM regresi menghasilkan nilai rata-rata poin yaitu 294 dan 293 dari total poin prediksi pada 5 pertandingan sebelumnya. perbandingan model regresi diukur menggunakan metrik *Mean Squared Error* diperoleh nilai rata-rata berdasarkan posisi pemain ialah 8,36 untuk LSTM dan 9,03 untuk MLP.

Delano Ramdas (2022), melakukan implementasi algoritma *Convolutional Neural Network* yang secara umum menggunakan data berupa gambar dan video pada studi kasus berbeda. Dalam penelitian ini CNN diterapkan untuk menangani kasus *time series forecasting* untuk memprediksi perfoma pemain sepakbola pada Fantasy Premier League (Ramdas, 2022). Data masukan diubah ke dalam bentuk matriks dan proses konvolusi sedikit berbeda dimana dimulai dari arah atas menuju ke bawah. *Hyperparameter tuning* juga diterapkan pada jumlah *input window size*, *kernel size* dan jumlah *filter*. Sehingga diperoleh rata-rata hasil evaluasi M*ean Squared Error* (MSE) terendah yaitu 1,35.

Perbandingan metode *machine learning* lainnya menggunakan metode Linear Regression, Decision Tree, Random Forest untuk memprediksi poin pada permainan Fantasy Premier League pada tahun 2022. Fitur penting yang terlibat diantaranya durasi waktu pemain dalam tiga pertandingan terakhir, total poin yang dihasilkan pemain dalam lima pertandingan terakhir, jumlah gol, mengahalau bola, dan melakukan *tackle* dijadikan sebagai data masukan sebuah model. Untuk mengetahui performa model, metrik *Root Mean Squared* Error (RMSE) menghasilkan nilai 2.172 dan 2.159 untuk data latih dan data uji. sedangkan penggunaan metrik evaluasi lainnya yaitu *Mean Absolute Error* (MAE) memperoleh 1.289 pada data latih dan 1.264 di data uji (Bangdiwala et al., 2022).

Tabel 2.1 Tinjauan Pustaka

Judul	**Penulis**	**Tahun**	**Dataset**	**Metode**	**Akurasi**
Time Series Modeling for Dream Team in Fantasy Premier	Akhil Gupta	2017	Situs resmi Fantasy Premier	ARIMA dan LSTM	Diperoleh proporsi optimal antara kedua metode ialah 40% ARIMA dan

League			League		60% LSTM menghasilkan nilai RMSE (vardy) ialah 2.808
Handling Highly Imbalanced Output Class Label : A Case Study On Fantasy Premier League (FPL) Virtual Player Price Changes Prediction Using Machine Learning	Muhammad Muhaimin and Ruhaila Maskat	2019	Data FPL musim 2016/2017 pada repository github Vaastav Anand	K-Nearest Neighbors (KNN)	Nilai standar deviasi terbaik diperoleh menggunakan metode SMOTE Upsampling kedua pada pembagian data train test stratified sebesar 5.7873 dengan durasi runtime selama 101 detik.
Multi-stream Data Analytics for Enhanced Performance Prediction in Fantasy Football	Nicolas Bonello, Joeran Beel, Seamus Lawless, Jeremy Debattista	2019	FPL API, Odds API, dan Twitter API	Gradien Boosting Machine	Model menghasilkan total 2314 poin dengan rata-rata 36 poin per pertandingan dengan menggunakan sumber data FPL API.
Optimising Daily Fantasy Sport Teams with Artificial Intelligence	Ryan Beal et al	2020	FanDuel 2014-2017	Linear Regression, LSTM, Radial Basis Function (RBF) dan Random	Dari 6 posisi yang tersedia pada dataset, diperoleh bahwa posisi tight-ends(TE), kickers (K), dan defence (DEF) memiliki nilai evaluasi lebih

				Forest (RF)	baik yaitu 4.10, 3.5, dan 3.85.
Prediksi Hasil Pertandingan Sepakbola English Premier League Dengan Menggunakan Algoritma K-Nearest Neighbors dan Naïve Bayes Classifier	Darmastyo Bagas Prabowo	2020	Bersumber dari situs www.football-data.co.uk	K-Nearest Neighbors dan Naïve Bayes	Hasil komparasi model terbaik diperoleh menggunakan algoritma Naïve Bayes dengan jumlah data latih sebesar 90% dan data uji sebesar 10%
Comparison of Machine Learning Approaches Applied to Predicting Football Players Performance	Adrian Lindberg dan David Sodeberg	2020	Repository github milik vaastav Anand, website sofascore, website understat	Support Vector Machine (SVM), Multi-Layer Perceptron (MLP) dan Long Short-Term Memory (LSTM)	Hasil terbaik diperoleh oleh algoritma MLP diikuti LSTM dengan total poin prediksi yaitu 294 dan 293. Nilai rata-rata MSE regresi terbaik diperoleh sebesar 8.36 menggunakan LSTM.
Using Convolutional Neural Network to Predict the Performance of Footballers in the Fantasy	Delano Ramdas	2022	Data FPL musim 2020/2021 pada repository github Vaastav	Convolutional Neural Network (CNN)	Hasil rata-rata evaluasi model menggunakan metrik Mean Squared Error (MSE) adalah 1.35

Premier League			Anand		
Using ML Models to Predict Points in Fantasy Premier League	Malhar Bangdiwala et al	2022	Data FPL musim 2016/2017 pada repository github Vaastav Anand	Linear Regression, Decision Tree, Random Forest	Berdasarkan pengujian pada ketiga model machine learning, Linear Regression dinilai tidak overffiting dengan nilai evaluasi metrik RMSE pada train dan test sebesar 2.172 dan 2.159 sedangkan metrik MAE sebesar 1.289 dan 1.264.

Berdasarkan penelitian terdahulu yang berkaitan dengan model prediksi pada game Fantasy Premier League menggunakan algoritma kecerdasan buatan, diperoleh kesimpulan sebagai berikut:

1. Kecerdasan buatan menjadi salah satu solusi dalam menangani permasalahan pada game Fantasy Premier League dengan menghadirkan model prediksi yang dibangun dari algoritma machine learning dan deep learning.
2. Berdasarkan tahun penelitian, implementasi deep learning lebih sering digunakan dibandingkan machine learning dalam beberapa tahun terakhir, algoritma deep learning dapat mengenali pola dan memahami data historis dalam jumlah besar serta menghasilkan nilai error yang lebih kecil.
3. Sumber data, pembagian data dan implementasi hyperparameter tuning yang bervariasi menghasilkan perbedaan signifikan pada suatu model sehingga dapat menjadi wawasan baru dalam suatu penelitian

Penelitian terdahulu dapat dijadikan landasan untuk merancang suatu model prediksi menggunakan algortima *deep learning* pada game Fantasy Premier League. Perbandingan algoritma CNN dan LSTM menggunakan skenario pembagian data pada sumber data yang paling banyak digunakan dalam penelitian sebelumnya akan dirancang

pada penelitian ini. diharapkan perancangan model mampu mengatasi variasi data dalam jumlah besar melalui kombinasi tahapan pemodelan sehingga menghasilkan model yang memiliki nilai error yang rendah.

2.2 Fantasy Premier League

Fantasy Premier League merupakan game virtual berbasis *fantasy sport*, yang memberikan kesempatan bagi para penggemar atau pecinta *English Premier League* (EPL) untuk merasakan pengalaman menjadi seorang manajer tim. Seorang manajer diberikan anggaran untuk memilih pemain-pemain favoritnya untuk berkompetisi dalam setiap pertandingan, mengikuti aturan yang telah ditetapkan. Setiap tim terdiri dari 11 pemain inti dan 4 pemain cadangan, dengan posisi pemain yang ditentukan, termasuk penjaga gawang, pemain bertahan, pemain tengah, dan pemain depan. Pemain meraih poin berdasarkan penampilan nyata mereka dalam kompetisi EPL sesuai regulasi penilaian poin FPL.

Aturan penilaian poin dibagi menjadi dua kategori, yaitu penambahan poin dan pengurangan poin. Penambahan poin diberikan kepada pemain yang berhasil mencetak gol *(goal)*, memberikan operan kepada pemain yang mencetak gol *(assist)*, melakukan penyelamatan untuk tim nya *(saves)*, dan menjaga gawang agar tidak kebobolan selama 90 menit *(clean sheet)*. Di sisi lain, pengurangan poin diberikan kepada pemain yang gagal dalam mengeksekusi tendangan penalti *(penalty missed)*, mencetak gol ke gawang sendiri *(own goal)*, mendapatkan kartu kuning *(yellow card)* atau merah *(red card)* dalam satu pertandingan. Besaran poin juga dipengaruhi oleh jumlah menit bermain tanpa tambahan waktu *(minutes)*, di mana pemain akan mendapatkan poin lebih tinggi setelah bermain selama 60 menit. Pemain bertahan dan penjaga gawang menerima poin lebih tinggi daripada pemain depan, terutama jika mereka berhasil mencetak gol. Selain itu, penjaga gawang yang mampu menggagalkan tendangan penalti juga mendapatkan poin yang signifikan *(penalty saved)*. Rincian mengenai penentuan poin ini dapat ditemukan di situs resmi FPL, *fantasy.premierleague.com* atau dapat dilihat pada tabel 2.2.

Manajer akan mendapatkan poin dari para pemain yang telah dipilih ke dalam tim impian. Perhitungan poin dimulai dari awal hingga akhir musim sepanjang 38 pertandingan. Penilaian poin terfokus pada pemain inti, sementara poin dari pemain cadangan dihitung jika pemain inti tidak bermain, sesuai dengan urutan pemain cadangan. Pemilihan kapten tim dapat dilakukan oleh manajer kepada salah satu pemain yang memiliki peluang besar mendapatkan poin tertinggi karena pemain akan menerima *double point*.

Tabel 2.2 Penilaian poin

No	**Ketentuan**	**Nilai Poin**
1.	Pemain yang bermain selama 60 menit	1
2.	Pemain yang bermain diatas 60 menit (tanpa tambahan waktu)	2
3.	Pemain bertahan dan penjaga gawang berhasil mencetak gol	6
4.	Pemain tengah berhasil mencetak gol	5
5.	Pemain depan berhasil mencetak gol	4
6.	Pemain yang memberi umpan sukses berbuah gol *(assist)*	3
7.	Penjaga gawang dan pemain bertahan tanpa kebobolan *(clean sheet)*	4
8.	Pemain tengah berhasil *clean sheet*	1
9.	Penjaga gawang yang berhasil melakukan penyelamatan sebanyak 3x	1
10.	Penjaga gawang yang berhasil mengagalkan tendangan pinalti	5
11.	Poin tambahan untuk pemain terbaik di setiap pertandingan	1-3
12.	Pemain yang gagal mengeksekusi tendangan pinalti	-2
13.	Penjaga gawang dan pemain bertahan yang kebobolan 2 gol	-1
14.	Pemain yang mendapatkan kartu kuning	-1
15.	Pemain yang mendapatkan kartu merah	-3
16.	Pemain yang melakukan gol bunuh diri *(own goal)*	-2

2.3 Deep Learning

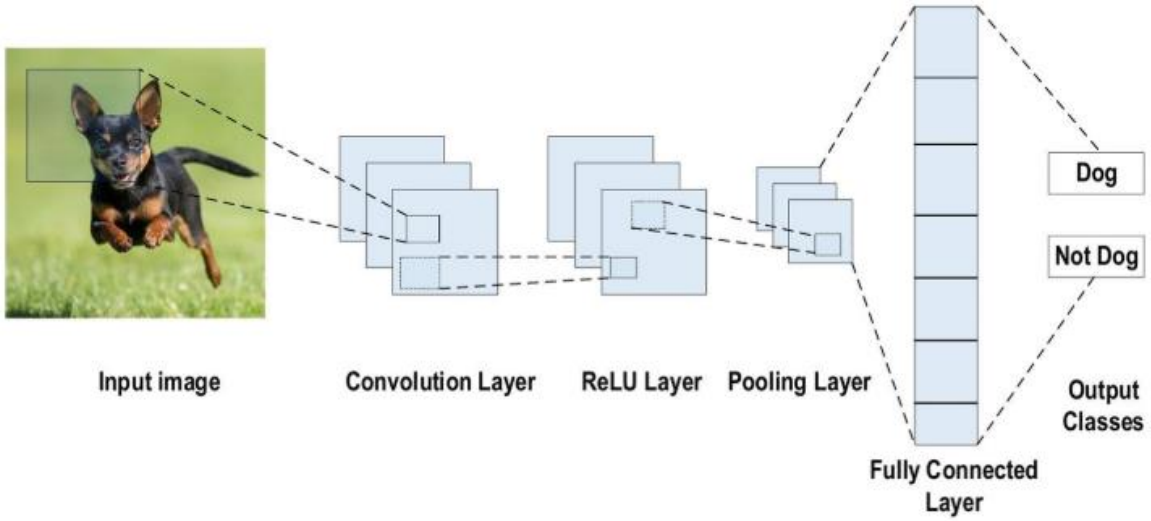
Deep Learning merupakan salah salah satu cabang dari kecerdasan buatan *(Artificial Intelligence)* yang memanfaatkan jaringan saraf tiruan *(neural network)* yang mendalam untuk memodelkan dan memahami data. Implementasi jaringan saraf tiruan memiliki kemiripan pada struktur dan fungsi otak manusia, yang melibatkan penggunaan *neural network* dengan banyak lapisan *(layers)* atau dikenal sebagai *deep neural network* . Jumlah *layers* dapat membantu model untuk secara otomatis mengekstrasi fitur-fitur yang semakin abstrak dari data yang diberikan. Sehingga *deep learning* dapat belajar memahami pola dari data yang berjumlah banyak dan kompleks (Alzubaidi et al., 2021).

Deep learning juga dinilai mampu menghadirkan solusi pada permasalahan dalam dunia nyata seperti peramalan deret waktu, pengenalan gambar, deteksi objek dan pemrosesan bahasa alami (Lim & Zohren, 2021). *Time series forecasting* memanfaatkan

suatu rangkaian waktu yang berisi informasi data historis untuk melakukan prediksi nilai di masa yang akan datang. Pada penelitian ini akan berkaitan dengan *time series forecasting*.

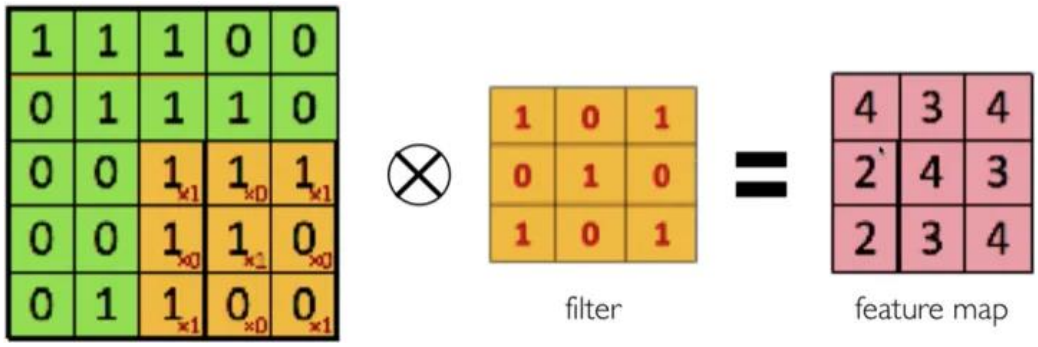
2.3.1 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) merupakan salah satu jenis *deep learning* yang terdiri dari beberapa lapisan jaringan saraf tiruan untuk mengindentifikasi informasi yang berasal dari sebuah citra. Konsep awal CNN terinspirasi dari virtual korteks pada hewan mamalia yaitu kucing. Kemampuan CNN dalam menganalisis data visual berupa gambar dan video, dapat digunakan sebagai pengenalan, mendeteksi dan klasifikasi sebuah objek (Marfu'ah & Kurniawardhani, 2020). Komponen arsitektur CNN meliputi *Convolution layer*, *Relu layer*, *Pooling layer* dan *Fully Connected layer* yang diilustrasikan melalui Gambar 2.1.



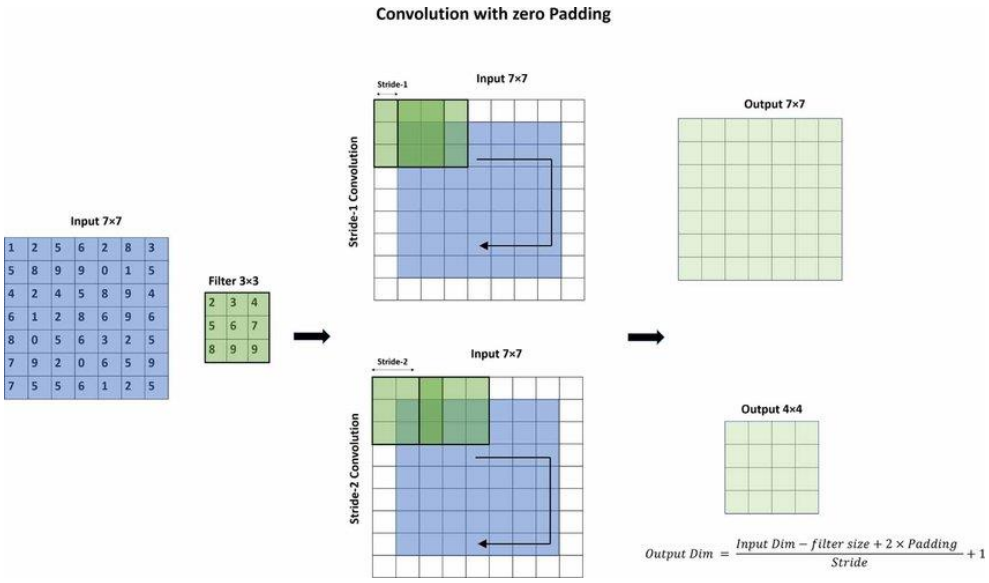
Gambar 2.1 Arsitektur model CNN

Pada *convolution layer* dilakukan proses *encoding* untuk mengubah data yang semula berbentuk gambar menjadi bentuk tiga dimensi. Ukuran dimensi memiliki panjang, lebar, dan kedalaman yang dijadikan sebagai data masukan. Semakin dalam dimensi data maka gambar bersifat RGB. Dalam layer ini, terdapat sebuah kernel yang berfungsi untuk melakukan *filter* terhadap data masukan. Kernel bergerak melintasi seluruh permukaan mulai dari awal hingga akhir dimensi data berdasarkan jumlah *stride*, biasa dikenal sebagai proses konvolusi seperti ditunjukkan oleh Gambar 2.2.



Gambar 2.2 Proses konvolusi CNN

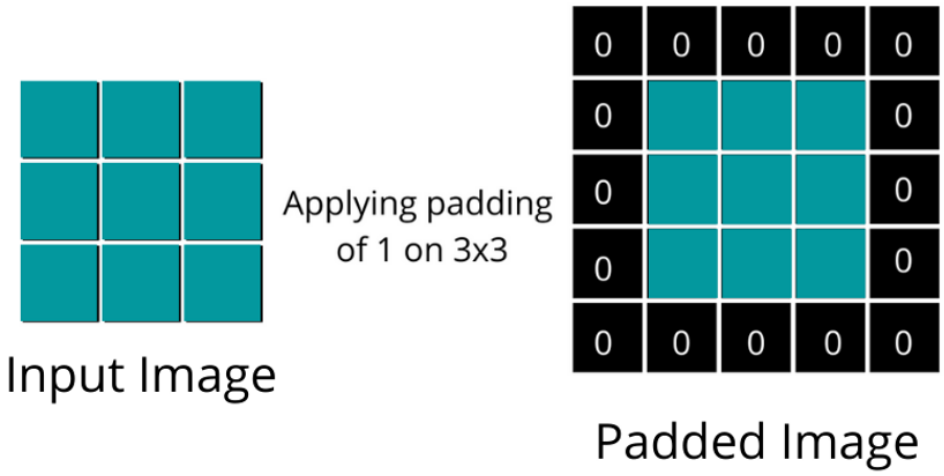
Stride merupakan sebuah parameter untuk mengatur jumlah pergeseran ketika proses konvolusi. Sebagai contoh, jika nilai *stride* adalah 2 maka *filter* kernel akan bergeser sebanyak 2 piksel secara horizontal dan vertikal seperti yang ditunjukkan pada Gambar 2.3. Semakin kecil nilai *stride* maka informasi yang diperoleh semakin detail. Namun, proses komputasi membutuhkan waktu yang lebih jika menggunakan *stride* kecil. Sehingga pemilihan nilai *stride* perlu dipertimbangkan untuk mendapatkan performa bagus.



Gambar 2.3 Stride CNN

Setiap pergeseran akan menghasilkan *feature map*, yang berisi fitur-fitur penting dari sebuah gambar. *Feature map* dihasilkan melalui proses perkalian dan penjumlahan selama konvolusi berlangsung. Selain itu, parameter lain yang digunakan untuk menambahkan ukuran piksel di area sekeliling data masukan adalah *Padding*. Pada Gambar 2.4 merupakan implementasi *padding* pada data masukan. Tujuan *padding* untuk

memanipulasi dimensi data masukan untuk menghasilkan *feature map* yang tidak terlalu kecil. *Feature map* akan selalu berukuran lebih kecil dari data masukan pada layer ini. kehilangan informasi yang semakin banyak dapat dicegah dengan penggunaan *padding* agar tetap menjaga ukuran dari *feature map* yang dihasilkan agar berhasil melakukan ekstraksi dengan jumlah fitur yang banyak.



Gambar 2.4 Implementasi padding

Implementasi fungsi aktivasi ditambahkan pada keluaran yang dihasilkan dari *convolution layer*. Terdapat 3 jenis fungsi aktivasi yaitu *Rectified Linear Unit (ReLu), Tanh, dan Sigmoid*. Fungsi aktivasi yang digunakan dalam arsitektur CNN adalah ReLu karena dapat mengubah nilai negatif yang dihasilkan pada proses sebelumnya menjadi nilai 0. Sedangkan nilai positif akan tetap sama sesuai nilai yang dihasilkan sehingga beban komputasi akan semakin ringan. Fungsi aktivasi relu secara matematis ditunjukkan melalui persamaan 2.1.

$$f(x)_{ReLu} = \max(0, x) \quad (2.1)$$

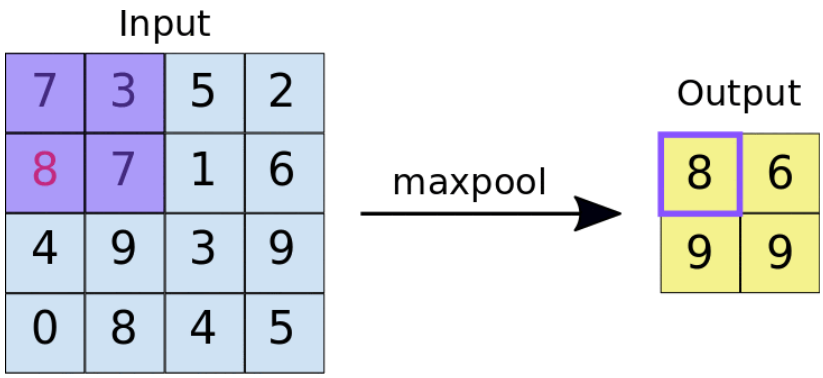
Dimana:

$f(x)_{ReLu}$ = fungsi aktivasi ReLu

x = nilai sebenarnya

Model arsitektur CNN selanjutnya akan melewati *pooling layer*. Proses *pooling* dilakukan untuk mengurangi dimensi data dan mempertahankan informasi penting pada *feature map*. proses pooling sedikit menyerupai proses konvolusi, yang membedakannya

ialah melakukan penyederhanaan tanpa mengurangi informasi penting yang terkandung memanfaatkan dua jenis *pooling* yaitu *average* dan *maxpooling*. *Average pooling* mengambil nilai rata-rata dan *Max pooling* mengambil nilai maksimum untuk mendapatkan hasil keluaran dalam proses *pooling layer* sehingga meminimalkan terjadinya *overfitting*. Gambar 2.5 merupakan proses *pooling layer* yang mengambil nilai maksimum sebagai keluaran yang siap diteruskan ke lapisan berikutnya.



Gambar 2.5 Pooling layer

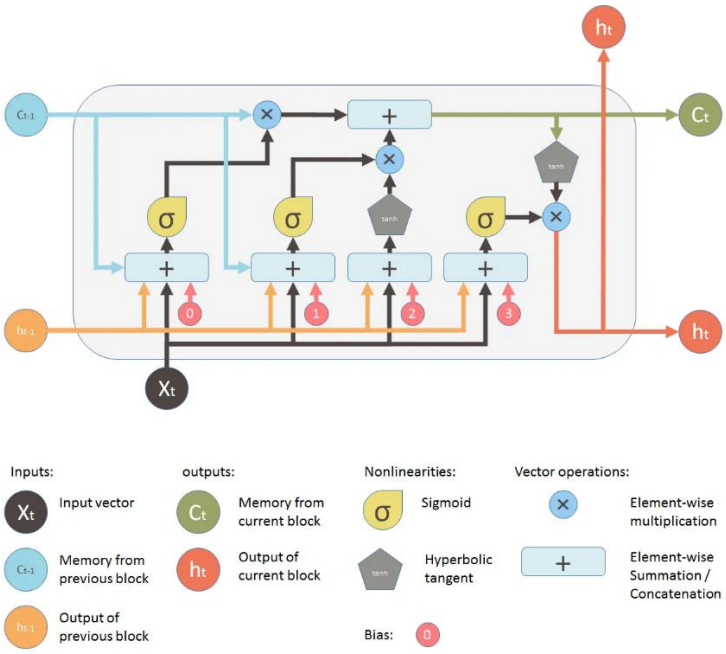
Seperti jaringan saraf tiruan, semua neuron terhubung satu sama lain dari lapisan sebelumnya ke lapisan selanjutnya. Pada *Fully Connected Layer*, *multidimensional array* yang dihasikan dari proses sebelumnya dilakukan *flatten*, yang bertujuan untuk mengubah bentuk dimensi menjadi satu kesatuan berbentuk *vector*. *Fully Connected Layer* berada pada lapisan terakhir pada arsitektur CNN sehingga menghasilkan sebuah keluaran dari model CNN.

2.3.2 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) merupakan salah satu pengembangan dari arsitektur *Recurrent Neural Network* (RNN) yang memanfaatkan salah satu jenis arsitektur *Artificial Neural Network* (ANN) untuk mempertahankan informasi dari waktu ke waktu. RNN akan menangkap sinyal dari langkah sebelumnya dan membawanya ke langkah berikutnya sebagai masukan untuk diproses secara rekursif, menghasilkan keluaran pada saat ini. Namun, RNN memiliki keterbatasan dalam mempertahankan informasi dari data yang memiliki jarak waktu yang panjang. Hal ini mengakibatkan munculnya permasalahan *vanishing gradient*, dimana nilai bobot berubah menjadi kecil selama proses pelatihan.

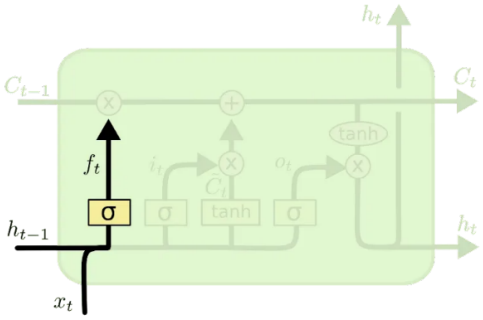
Sepp Hocreiter dan *Jurgen Schmidhuber* memperkenalkan model arsitektur *Long Short-Term Memory* pada tahun 1997 dinilai menjadi solusi untuk menangani

permasalahan *vanishing gradient* pada RNN. Berbeda dengan RNN yang memanfaatkan *neuron* pada lapisan tersembunyi, LSTM memiliki kumpulan sel memori *(cell state)* yang diperbarui melalui komponen utama berupa struktur gerbang untuk mempertahankan informasi yang diperoleh. Setiap sel memori berisi struktur gerbang yaitu *input gate*, *forget gate* dan *output gate.* Seluruh rangkaian proses yang terjadi dalam sel memori digambarkan oleh Gambar 2.6.



Gambar 2.6 Arsitektur model LSTM

Forget gate dalam sebuah sel memori memiliki fungsi untuk menentukan informasi mana saja yang akan dipertahankan dan dibuang. Pada gambar 2.7 menunjukkan bahwa sel memori menerima informasi berupa *hidden state* yang dihasilkan sebagai keluaran dari cell sebelumnya (h_{t-1}) dan informasi baru yang diterima saat ini (x_t). Selanjutnya, kedua informasi tersebut akan digabung menjadi satu ke dalam sebuah vektor menggunakan operasi konkatenasi. Setelah penggabungan informasi berhasil maka proses selanjutnya menggunakan fungsi sigmoid. fungsi aktivasi sigmoid akan menghasilkan nilai 0 dan 1, yang berarti informasi dapat diteruskan apabila nilai yang dihasilkan mendekati nilai 1. Sebaliknya, jika nilai mendekati angka 0 maka informasi akan dibuang. Formula matematika pada *forget gate* direpresentasikan pada persamaan 2.2.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Gambar 2.7 Forget gate

$$f_t = \; \sigma\left(W_f . [h_{t-1}, x_t] + \; b_f\right) \tag{2.2}$$

Dimana :

f_t = Forget gate

σ = Fungsi aktivasi sigmoid

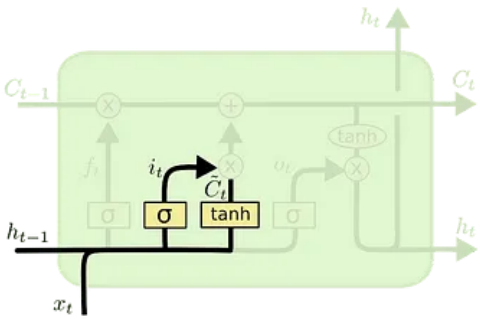
W_f = Nilai weight pada forget gate

h_{t-1} = Nilai output dari cell state sebelumnya

x_t = Nilai input saat ini

b_f = Nilai bias pada forget gate

Pada struktur gerbang lainnya yaitu *input gate*, dapat mengontrol informasi yang berasal dari *cell state* sebelumnya dan menerima informasi baru yang diterima saat ini. Gambar 2.9 menunjukkan *input gate* berkolaborasi bersama dua fungsi aktivasi berbeda. Bersama fungsi aktivasi sigmoid akan menentukan update nilai vektor pada sel memori. Kemudian, pembaharuan nilai vektor dihasilkan dari persamaan 2.3. Sedangkan penggunaan fungsi aktivasi lainnya menggunakan tanh, untuk mengontrol informasi baru yang ditambahkan ke cell state. Nilai vektor yang akan dijadikan sebagai kandidat baru diproses melalui persamaan 2.4.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Gambar 2.8 Input gate

$$i_t = \sigma\,(W_i.[h_{t-1}, x_t] + b_i) \quad (2.3)$$

Dimana :

i_t = Input gate

σ = Fungsi aktivasi sigmoid

W_i = Nilai weight pada input gate

h_{t-1} = Nilai output dari cell state sebelumnya

x_t = Nilai input saat ini

b_i = Nilai bias pada input gate

$$\widehat{C}_t = \tanh\,(W_c.[h_{t-1}, x_t] + b_c) \quad (2.4)$$

Dimana :

$\widehat{C}_t$ = Nilai vektor baru yang ditambahkan ke cell state

Tanh = Fungsi aktivasi tang

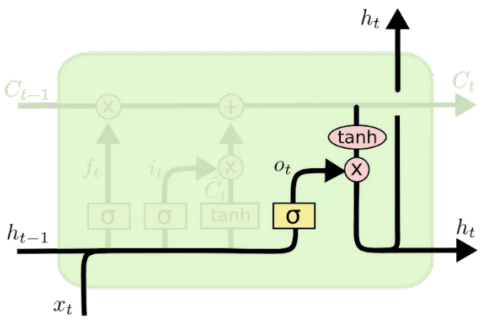
W_c = Nilai weight pada cell state

h_{t-1} = Nilai output dari cell state sebelumnya

x_t = Nilai input saat ini

b_c = Nilai bias pada cell gate

Output gate merupakan gerbang terakhir yang harus dilalui sebelum informasi dikirim ke *cell state* selanjutnya. *hidden state* yang berasal dari *cell* sebelumnya akan diproses menggunakan fungsi sigmoid melalui persamaan 2.5. Kemudian, *cell state* baru diolah dengan fungsi aktivasi *tanh* dapat dilihat pada persamaan 2.6. Hasil dari fungsi tanh dikalikan dengan hasil dari fungsi sigmoid dan disimpan di *hidden state* baru. Jika cell state dan *hidden state* sudah terbentuk, maka siap diteruskan ke *cell* selanjutnya. Pada gambar 2.9 merupakan keberadaan *output gate* pada sel memori.



$$o_t = \sigma\,(W_o\,[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh\,(C_t)$$

Gambar 2.9 Output gate

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

$$h_t = o_t * \tanh\,(C_t) \quad (2.6)$$

Dimana :

o_t = Output gate

σ = Fungsi aktivasi sigmoid

W_o = Nilai weight pada output gate

h_{t-1} = Nilai output dari cell state sebelumnya

x_t = Nilai input saat ini

b_o = Nilai bias pada output gate

h_t = Nilai output cell state ke-

C_t = Nilai bias pada output gate

2.3.3 Least Absolute Shrinkage and Selection Operator (LASSO)

Least Absolute Shrinkage and Selection Operator (LASSO) merupakan salah satu bagian dari linear regression yang dapat menemukan hubungan antara variabel (fitur) terhadap variabel target untuk menghasilkan suatu model prediksi dengan data yang telah diketahui variabel targetnya *(supervised learning)*. Istilah *"shrinkage"* digunakan pada LASSO karena dapat mengalami penyusutan nilai koefisien pada model regresi. Penyusutan nilai koefisien terjadi untuk mencegah model mengalami *overfitting*. dalam dunia *machine learning*, model yang mengalami *overfitting* tidak dapat diandalkan sebagai hasil keluaran. Dimana model prediksi mengalami kompleksitas data pada data latih sehingga ketika dilakukan pengujian terhadap data baru model tidak mampu melakukan prediksi secara optimal.

Pencegahan yang dapat dilakukan untuk menangani *overfitting* model dapat dilakukan dengan menerapkan teknik regularisasi. Teknik ini bekerja dengan menambahkan penalti pada formula model regresi. Terdapat beberapa jenis regularisasi yang dibedakan berdasarkan nilai penalti yang ditambahkan, sedangkan LASSO menggunakan regularisasi L1 menambahkan nilai absolut fitur dari koefisien fitur sebagai penalti untuk meminimalkan nilai koefisien regresi hingga nilai 0. Peningkatan nilai λ yang digunakan sebagai parameter non negatif akan mempengaruhi perubahan nilai koefisien menuju nilai 0 (Ardiansyah et al., 2020).

Perubahan nilai koefisien dapat menggugurkan fitur-fitur yang tidak relevan dan tidak memiliki pengaruh signifikan terhadap target. nilai koefisien fitur bernilai 0 menandakan bahwa fitur tersebut tidak memiliki korelasi yang kuat sehingga jika dihilangkan tidak mempengaruhi model prediksi. Hal ini sangat penting dilakukan dalam membangun model prediksi, khususnya dalam proses seleksi fitur. Pemilihan fitur-fitur yang tepat dapat mencegah model untuk mempelajari data secara kompleks sehingga dimensi data lebih sederhana. Selain itu, estimasi waktu pembelajaran model lebih efisien. Regresi LASSO ditemukan oleh seorang peneliti bernama *Thibsirani* pada tahun 1996 (Kim et al., 2019). Formula regresi LASSO yang dikonsepkan secara matematis ditunjukkan pada persamaan 2.7.

$$min_{\beta} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} x_{ij} \{\beta_j \; x \; M_j\} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \qquad (2.7)$$

Dimana:

n = total observasi

p = total peubah fitur

i = urutan observasi

j = urutan peubah fitur

y_i = respon peubah terhadap observasi ke-*i*

x_{ij} = nilai peubah fitur j pada observasi ke-*i*

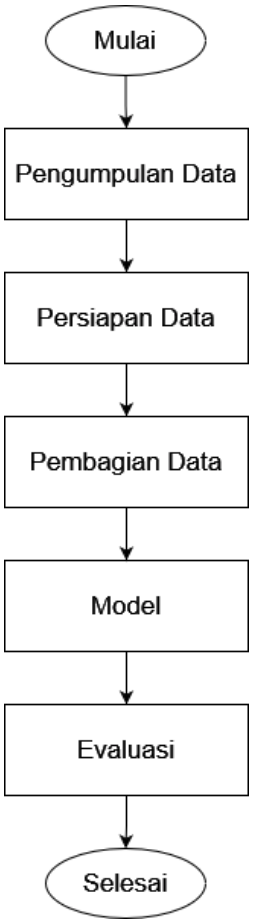
β_j = nilai koefisien parameter pada fitur j

$\lambda =$ nilai parameter non negatif

BAB 3
Metodologi

3.1 Alur Penelitian

Pada bagian ini berisi tentang alur penelitian yang telah dilakukan oleh penulis secara terstuktur. Alur penelitian memuat langkah-langkah penelitian yang dikerjakan mulai dari pengumpulan data, persiapan data, pembagian data, model, dan evaluasi. Alur penelitian dapat dilihat pada bagan alur pada gambar 3.1.



Gambar 3.1 Alur penelitian

3.2 Pengumpulan Data

Langkah awal penelitian ini dimulai melalui proses pengumpulan data. Data memiliki peranan utama dalam perancangan model prediksi. Secara umum, terdapat beberapa metode dalam melakukan pengumpulan data seperti observasi, wawancara, dokumentasi yang dinilai sebagai data kualitatif (Ardiansyah et al., 2023). Sedangkan data kuantitatif

dapat diperoleh melalui kuesioner dan observasi terstruktur untuk menemukan nilai pada variabel-variabel yang telah ditentukan sebelumnya. pengumpulan data secara modern juga dapat dilakukan melalui alat pendeteksi (sensor). Sensor akan mengirimkan informasi berupa data pada perangkat yang telah terhubung sebelumnya. Selain itu, beberapa instansi atau organisasi saat ini telah menyediakan data yang dapat diakses secara publik sehingga dapat digunakan oleh berbagai pihak untuk kepentingan lain. Data publik secara umum dipublish pada website resmi atau melalui *repository* yang dapat dilakukan pembaharuan secara berkala sehingga data yang disediakan tidak kadaluarsa.

Dalam penelitian ini, proses pengumpulan data menggunakan data publik yang telah disediakan oleh salah satu akun *repository github* bernama *Vaastav Anand*. Kumpulan data memuat data statistic para pemain pada game FPL di mulai dari musim 2016/2017 hingga 2023/2024. pemilihan data statistik dalam penelitian ini hanya satu musim saja, yaitu musim 2021/2022 yang bertujuan untuk menghindari bias penelitian karena setiap pemain dapat berpindah klub ketika periode bursa transfer aktif (Yang, 2019). Setiap klub juga tidak dapat menjamin para pemain akan selalu bersama klub untuk musim selanjutnya, terutama klub yang gemar mengganti pelatih ketika target dan tujuan tidak tercapai pada musim tersebut.

Dataset memuat parameter penilaian (variabel) pada pemain sepakbola yang disajikan secara kuantitatif. Parameter penilaian nantinya akan menjadi prediktor pada pemodelan. jumlah parameter yang dalam penelitian sebanyak 36, dapat dilihat pada tabel 3.1. Terdapat beberapa nilai variabel yang bergantung pada posisi pemain. sebagai contoh, *penalties_missed* yang nilai nya akan meningkat pada posisi penjaga gawang sedangkan *assist* dan *goal* memiliki kesempatan yang sangat minim. Berbanding terbalik dengan posisi pemain tengah atau penyerang yang dapat menghasilkan nilai tinggi pada variabel tersebut.

Tabel 3.1 Dataset Penelitian

No	**Variabel**	**Penjelasan**
1.	name	Nama pemain
2.	position	Posisi pemain
3.	team	Klub pemain saat ini
4.	xP	Penilaian performa dalam pertandingan sebelumnya
5.	assists	Passing yang menghasilkan gol

6.	bonus	Pemberian bonus poin pada pemain
7.	bps	Penilaian sistem dalam pemberian bonus poin
8.	clean_sheets	Tanpa kebobolan dalam pertandingan
9.	creativity	Penilaian kreatifitas dalam membuat peluang
10.	element	Kode unik pemain
11.	fixture	Id setiap pertandingan
12.	goals_conceded	Total kebobolan
13.	goals_score	Total mencetak gol
14.	ict_index	Kombinasi perhitungan metrik influence, creativity dan threat
15.	influence	Penilaian terhadap pengaruh pemain dalam pertandingan
16.	kickcoff_datetime	Waktu pertandingan dilaksanakan
17.	minutes	Durasi waktu pemain dalam pertandingan
18.	opponent_team	Id unik untuk klub away
19.	own_goals	Total goal bunuh diri
20.	penalties_missed	Total gagal mengeksekusi pinalti
21.	penalties_saved	Total penyelamatan pinalti
22.	red_cards	Total kartu merah
23.	yellow_cards	Total kartu kuning
24.	round	Ronde pertandingan ke-
25.	saves	Penyelamatan yang dilakukan pemain
26.	selected	Total pemain terpilih oleh manajer lain
27.	team_a_score	Total gol pada klub away
28.	team_h_score	Total gol untuk klub home
29.	threat	Besaran ancaman yang dibuat oleh pemain
30.	transfer_balance	Nilai perbandingan antara transfer_in dan transfer_out
31.	transfer_in	Total pergantian pemain yang dilakukan oleh manajer lain untuk dimasukkan ke dalam tim impian
32.	transfer_out	Total pergantian pemain yang dilakukan oleh manajer untuk dikeluarkan dari tim impian
33.	talue	Harga pemain
34.	was_home	Status pemain dalam pertandingan

35.	gw	Pertandingan minggu ke-
36.	total_points	Total poin yang dihasilkan pemain

3.3 Persiapan Data

Persiapan data merupakan salah satu rangkaian penting yang terlibat dalam penelitain ini yang bertujuan untuk mempersiapkan data dari proses pengumpulan data agar dapat diterima oleh model. Menurut *Carnegie*, persiapan data akan menghasilkan data baru yang memiliki kualitas bagus sehingga dapat meningkatkan performa model (Carnegie, 2023). Persiapan data menghabiskan waktu lebih lama dibandingkan tahapan lain dalam perancangan model. hal ini dikarenakan terdapat beberapa langkah-langkah yang diperlukan untuk mengolah data agar menghasilkan model prediksi yang akurat. Persiapan data akan yang dilakukan meliputi pemrosesan data, seleksi fitur, normalisasi, dan transformasi data. Berikut penjelasan terkait langkah-langkah pada persiapan data.

3.3.1 Pemrosesan Data

Tidak semua data yang disediakan secara publik dalam kondisi siap digunakan secara langsung. Sebagian besar data yang tersedia perlu diproses sesuai dengan kebutuhannya. data yang didapatkan dari proses sebelumnya kemudian perlu dilakukan pemrosesan data. dalam tahap pemrosesan bertujuan untuk melakukan pengecekan data yang tidak memiliki nilai *(missing value)*, duplikasi data (duplicate data) dan kelengkapan data *(completeness of data)*. Keberadaan data yang tidak memiliki nilai pada variabel-variabel tertentu atau ditemukan duplikasi data pada suatu data dapat dikatakan sebagai *noisy data* (Sabar Sautomo & Hilman Ferdinandus Pardede, 2021). Hal ini akan menganggu hasil dari model prediksi yang dibangun.

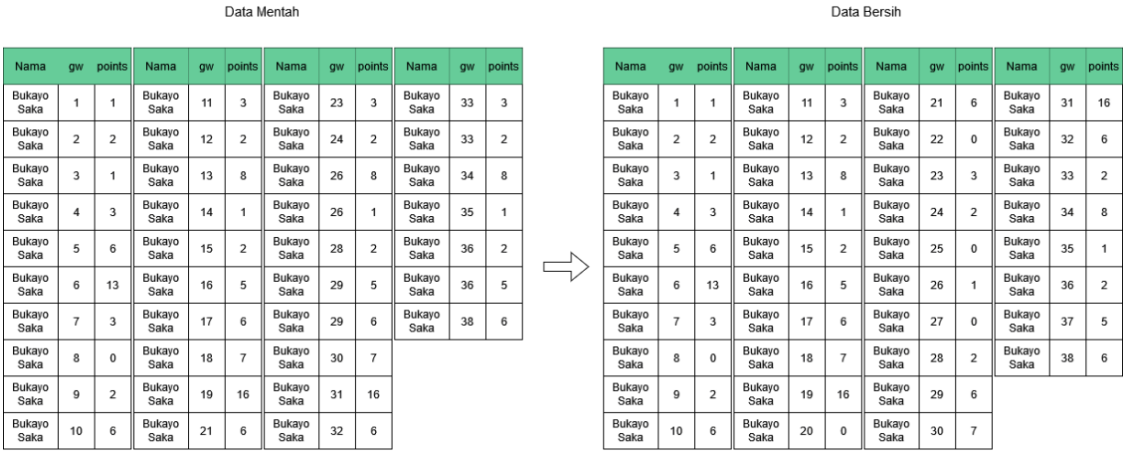
Untuk menangani kasus nilai kosong pada suatu variabel dan duplikasi data dapat dilakukan dengan cara mengganti nilai yang kosong atau menghapus data (Purbolaksono et al., 2021). Namun, perlu dilakukan pengecekan terlebih dahulu terhadap dua kondisi tersebut sebelum dilakukan penghapusan data. Penghapusan data akan mengurangi dimensi data sehingga jumlah data perlu disesuaikan dengan kebutuhan. Mengingat dalam konsep *deep learning*, untuk menghasilkan akurasi yang makimal diperlukan data dalam jumlah besar.

Kelengkapan data bertujuan untuk menjaga format data agar tetap konsisten sesuai dengan aturan permainan FPL. Dimana jumlah pertandingan yang berlangsung dalam satu musim yaitu sebanyak 38 pertandingan. Setiap pemain yang telah terdaftar pada pada klub

FPL semestinya memiliki jumlah pertandingan yang sama. Faktanya, pemain FPL yang terdaftar memiliki jumlah pertandingan yang bervariasi sehingga perlu adanya kesetaraan jumlah pertandingan antar pemain.

Inkonsistensi data tidak dapat diterima oleh model *time series*. Dimana jumlah pertandingan akan digunakan sebagai deret waktu yang berperan sebagai urutan data yang akan dikenali oleh model nantinya. agar data memiliki deret waktu yang sesuai di setiap individu pemain maka perlu modifikasi data. Pemain yang telah memiliki minimal 30 pertandingan namun tidak melampaui batas maksimum pertandingan dalam semusim yaitu sebanyak 38 pertandingan akan dilakukan imputasi data. sedangkan para pemain yang tidak mencapai total pertandingan sebanyak 30 pertandingan akan dihilangkan karena dapat menyebabkan redudansi data. imputasi data secara manual dalam jumlah besar dapat mengganggu integritas data sehingga tindakan penghapusan data dinilai tepat untuk menangani kondisi tersebut. imputasi data dinilai dapat menjadi solusi terhadap inkonsistensi data yang menjadi permasalahan pada penelitian ini karena dapat menambahkan nilai pada variabel-variabel di setiap pemain. imputasi data yang dilakukan dengan menambahkan nilai 0 (bukan null) pada seluruh variabel pada pemain agar tercapai konsistensi pada data deret waktu (Prasetya et al., 2023). Asumsi tersebut diperoleh atas dasar pemahaman penulis terhadap konsep model *time series* yang disesuaikan dengan kondisi data yang tersedia.

Pemrosesan data dilakukan untuk mencapai kesetaraan jumlah deret waktu antar pemain yang diperlukan untuk memudahkan model mengenali pola *time series*. jumlah deret waktu yang dimiliki oleh seorang pemain berbeda-beda, sedangkan fitur yang merepresentasikan deret waktu ialah fitur gw. Sebagai contoh, pemrosesan data dilakukan pada salah satu pemain Arsenal bernama *Bukayo Saka* seperti yang ditunjukkan pada Gambar 3.2. Awalnya, jumlah deret waktu miliki *Bukayo Saka* tidak mencapai jumlah sebanyak 38. Urutan deret waktu yang semula berjalan baik dari gw ke-1 hingga gw ke-19, kemudian mengalami kehilangan data untuk data selanjutnya pada gw ke-20, 22, 25, 27, dan 37. Selain itu, urutan deret waktu juga mengalami duplikasi data pada gw ke-26, 29, 33, dan 36. data duplikasi yang berada di awal akan dilakukan penghapusan karena data terakhir merupakan data terbaru dari nilai gw yang mengalami duplikasi data. Pemrosesan data dilanjutkan untuk mencapai kesetaraan deret waktu dengan cara menambahkan nilai baru pada fitur gw yang mengalami kehilangan atau kekosongan data agar dapat diterima oleh model time series dalam mempelajari pola data.



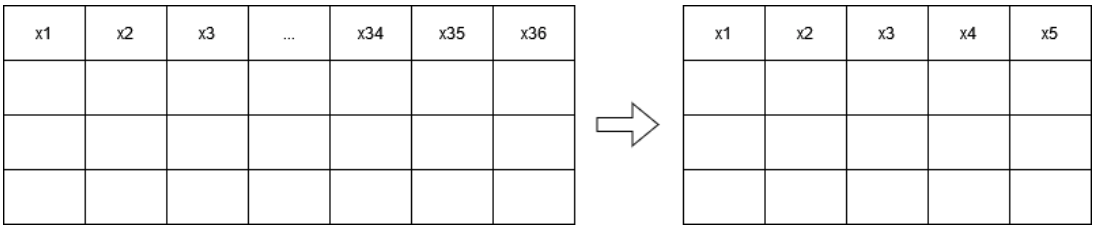
Gambar 3.2 Pembersihan data pemain

3.3.2 Seleksi Fitur

Seleksi fitur merupakan suatu proses pemilihan fitur atau variabel yang dinilai relevan sehingga dapat meningkatkan performa model. dalam proses seleksi akan terpilih fitur terbaik dari beberapa fitur yang tersaji pada data (Zulaikhah Hariyanti Rukmana et al., 2022). fitur yang dinilai dapat mempengaruhi kinerja model akan dipertahankan sedangkan fitur yang bersifat redudansi dan tidak memiliki hubungan terhadap model akan dihilangkan dalam seleksi fitur. Penyebab fitur dihilangkan karena fitur-fitur tersebut tidak dapat memberikan kontribusi signifikan terhadap pembelajaran atau informasi yang terkandung memiliki kemiripan dengan informasi pada fitur lainnya.

Terdapat beberapa metode seleksi fitur diantaranya ialah metode *filter*, *wrapper* dan *embedded* (Rosihan et al., 2023). Metode *embedded* melakukan kombinasi antara metode *filter* dan *wrapper*, dimana memanfaatkan perhitungan statistik *(metode filter)* kemudian mengimplementasikan model pembelajaran lain *(metode wrapper)* untuk menemukan fitur terbaik. Pada penelitian ini menggunakan metode *Least Absolute Shrinkage and Selection Operator* (LASSO) yang tergolong dalam metode *embedded* untuk seleksi fitur. Penggunaan regularisasi dan penyusutan nilai koefisien dapat mereduksi data yang bersifat kompleks (Guenther & Sawodny, 2019). Reduksi data mengubah dimensi data lebih sederhana sehingga dapat meningkatkan efisiensi waktu pada pembelajaran.

Konsep seleksi fitur telah divisualisasikan pada Gambar 3.3 dimana hasil seleksi fitur terpilih menjadi lebih sedikit dibandingkan jumlah fitur sebelumnya. Data yang awalnya memiliki fitur atau variabel sebanyak 36 akan menghasilkan 5 variabel setelah dilakukan seleksi fitur. Kelima fitur terpilih dinilai merupakan fitur terbaik yang diperoleh dari fitur-fitur lainnya. implementasi metode LASSO dalam pembuatan model *time series* lainnya untuk memprediksi total penjualan menghasilkan dimensi data yang lebih sederhana seperti fitur *Store*, *Dept*, dan *Size* dari 14 fitur yang tersedia (Faiqoh, 2023). Nilai metrik evaluasi menunjukkan bahwa model yang menerapkan seleksi fitur, menghasilkan nilai evaluasi lebih rendah dibandingkan dengan model tanpa melakukan seleksi fitur. Hal ini dikarenakan data yang tidak relevan diikutsertakan dalam pembelajaran model.



Gambar 3.3 Proses seleksi fitur

3.3.3 Normalisasi

Data yang telah melalui pemrosesan dan seleksi fitur memiliki rentang nilai yang berbeda untuk setiap fitur atau variabel. Kesesuaian rentang nilai antara fitur terpilih yang satu dengan fitur terpilih lainnya diperlukan untuk mencapai model prediksi yang optimal (Dicoding, 2020). Normalisasi bertujuan untuk mengubah skala data yang memiliki rentang nilai yang berbeda-beda ke dalam skala yang sama. Skala data akan diubah menjadi bentuk yang lebih sederhana pada rentang nilai 0 hingga 1. Berdasarkan perbandingan metode normalisasi yang telah dilakukan menggunakan algoritma *machine learning* untuk mengidentifikasi tanaman, diperoleh normalisasi menggunakan *Min-Max* memiliki nilai akurasi lebih tinggi di setiap pembelajaran dibandingkan normalisasi *Zero-mean* (Ambarwari et al., 2020).

Penggunaan normalisasi *Min-Max* akan mengubah nilai pada setiap fitur yang berada antara nilai 0 sampai 1. Rentang nilai pada fitur x1 hingga x4 lebih kecil jika dibandingkan fitur x5, yang memiliki rentang nilai maksimum 90 seperti yang ditunjukkan pada Gambar 3.4. Implementasi normalisasi dapat menyetarakan nilai kedua fitur sehingga

proses pembelajaran model semakin cepat (Ambarwari et al., 2020). Hasil perubahan rentang nilai pada normalisasi Min-Max diperoleh dari persamaan 3.1

x1	x2	x3	x4	x5
1.1	0	0	0	0
0.02	0	5	2.5	29
4.7	0	30	16.1	90

x1	x2	x3	x4	x5
0.132653	0	0	0	0
0.086735	0	0.145833	0.079365	0.322222
0.290816	0	0.152778	0.200000	1.000000

Gambar 3.4 Normalisasi Min-Max

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.1}$$

Dimana:

X_{new} = hasil dari normalisasi data

X = nilai pada dataset

X_{min} = nilai minimum pada dataset

X_{max} = nilai maximum pada dataset

3.3.4 Transformasi Data

Transformasi data merupakan bagian dari serangkaian persiapan data yang menghasilkan data ke dalam format yang dapat diterima oleh algoritma pemodelan (Tarigan et al., 2023). Setiap algoritma berusaha mempelajari data berdasarkan karakteristik masing-masing. Transformasi data akan menghasilkan bentuk data baru yang dapat memudahkan model untuk memahami karakteristik pola pada data masukan. Dalam kasus data *time series*, diperlukan representasi waktu yang akan dijadikan sebagai urutan data. Secara umum, jenis waktu dibedakan menjadi jam, hari, minggu, bulan atau tahun untuk melakukan peramalan model *time series* (Zhang & Thorburn, 2022).

Model prediksi untuk mengetahui nilai total poin pada pemain FPL di pertandingan (GW) ke-6 menggunakan data pemain FPL mulai dari pertandingan ke-1 hingga pertandingan ke-5 yang dimainkan dalam satu musim. Penentuan 5 pertandingan yang dilaksanakan pada awal musim untuk melihat apakah pemain memiliki performa baik di awal musim. Mengingat, terdapat pemain yang memerlukan adaptasi terhadap klub jika pemain tersebut baru bergabung bersama klub. Konsistensi pemain yang telah berhasil memiliki performa bagus akan selalu tetap terjaga hingga pertandingan-pertandingan

selanjutnya dapat dipelajari oleh model berdasarkan data historis pertandingan sebelumnya. Jumlah pertandingan sebelumnya yang digunakan dalam penelitian sebanyak 5 pertandingan.

Dalam upaya menyempurnakan pemahaman terkait model *time series*, setiap pemain akan memiliki jumlah data yang sama dan urutan data direpresentasikan pada kolom ‘gw’ yang digambarkan pada Gambar 3.5. Bentuk data yang terlihat seperti tabel berisi baris dan kolom kemudian akan diubah ke dalam sebuah *array*. Setiap matriks dalam *array* mewakili satu pemain FPL yang berisi nilai-nilai dari berbagai fitur. Kolom pertama akan merepresentasikan fitur pertama hingga fitur terakhir yang mengikuti hasil seleksi fitur. Sedangkan baris akan merepresentasikan *gameweek* yang digunakan sebagai urutan data. informasi pada pemain selanjutnya didapatkan pada matriks lainnya secara menurun hingga maksimum pemain yang didaftarkan. Kumpulan matriks berbentuk sebuah *array* telah siap dijadikan sebagai data masukan pada model *time series*.

name	gw	xp	bps	value
Bukayo Saka	1	3.3	4	12.3
Bukayo Saka	2	1.5	8	16.5
Bukayo Saka	3	0.8	0	0.8
Bukayo Saka	4	3.0	18	63
Bukayo Saka	5	4.0	16	63
Bruno	1	4.4	61	120
Bruno	2	11.0	7	121
Bruno	3	8.2	4	121
Bruno	4	4.7	33	120
Bruno	5	3.2	17	120

$$\begin{bmatrix} \begin{bmatrix} 3.3 & 4 & 12.3 \\ 1.5 & 8 & 16.5 \\ 0.8 & 0 & 0.8 \\ 3.0 & 18 & 63 \\ 4.0 & 16 & 63 \end{bmatrix} \\ \begin{bmatrix} 4.4 & 61 & 120 \\ 11.0 & 7 & 121 \\ 8.2 & 4 & 121 \\ 4.7 & 33 & 120 \\ 3.2 & 17 & 120 \end{bmatrix} \end{bmatrix}$$

Gambar 3.5 Proses transformasi data

3.4 Pembagian Data

Pembagian data dilakukan untuk membagi data menjadi dua bagian yaitu data latih *(training data)* dan data uji *(test data).* Dimana data latih memiliki fungsi untuk menyediakan data yang dapat digunakan oleh model dalam proses pelatihan atau pembelajaran. Sedangkan data uji akan digunakan oleh model untuk melakukan evaluasi

model terhadap data yang belum dikenali sebelumnya. Dalam berbagai riset penelitian telah banyak skenario pembagian data yang dapat digunakan, diantaranya menggunakan rasio pembagian data latih dan data uji sebesar 70:30 dan 80:20 (Alonso & Babac, 2022; Van Eetvelde et al., 2021).

Kuantitas data pada data latih akan selalu lebih besar dibandingkan data uji agar model dapat memahami data lebih banyak sehingga menghasilkan model yang optimal. Pembagian data juga akan disesuaikan berdasarkan posisi pemain sehingga kuantitas data pada masing-masing posisi akan berbeda, mengikuti skenario pembagian data yang digunakan. Penelitian ini akan memiliki 4 *subset* data yang terdiri dari data penjaga gawang, pemain bertahan, pemain tengah dan penyerang. Dimensi data pada masing-masing *subset* data memiliki jumlah dan karakteristik berbeda setelah mengikuti seluruh tahapan pada persiapan data. Untuk menambah wawasan terkait penelitian, pembagian data dibedakan menjadi dua skenario. Skenario pertama, data latih berjumlah 70% dari jumlah data dan data uji sebesar 30%. Sedangkan untuk skenario kedua, data latih lebih besar daripada skenario sebelumnya yaitu 80% dan 20% untuk data uji.

3.5 Model

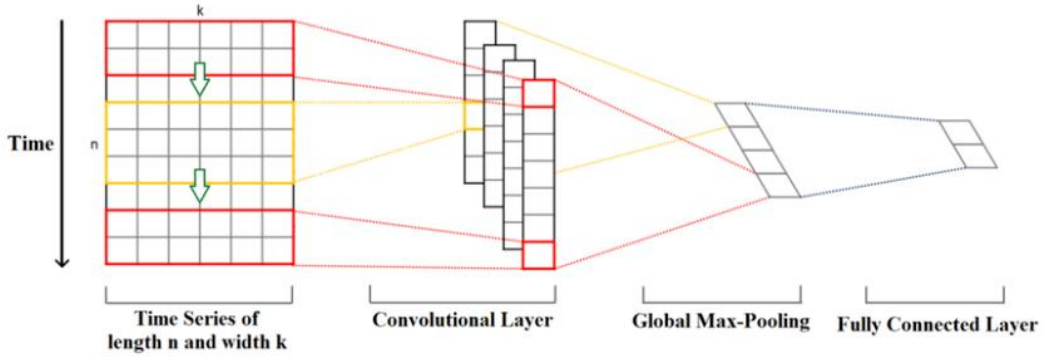
Perancangan model pada penelitian ini menggunakan model *deep learning time series* pada algoritma *Convolutional Neural Network* dan *Long Short-Term Memory*. Kedua algoritma ini digunakan untuk melakukan prediksi poin pada game Fantasy Premier League. model *time series* ini akan memanfaatkan data historis waktu untuk melakukan prediksi. Setiap algoritma akan menghasilkan 4 model yang telah dipisahkan berdasarkan posisi pemain. hal ini bertujuan untuk mencapai objektivitas pada game FPL, dimana pemilihan pemain akan dibedakan berdasarkan posisi pemain. Kedua model arsitektur yang dirancang menggunakan jumlah unit pada masing-masing lapisan dan fungsi aktivasi yang sama. Selain itu, lapisan terakhir dari kedua model hanya akan memuat 1 unit yang digunakan sebagai *output* untuk melakukan prediksi dalam rangkaian waktu.

3.5.1 Convolutional Neural Network (CNN)

Model arsitektur yang dirancang menggunakan algoritma CNN memanfaatkan konvolusi 1 dimensi untuk menangani kasus *time series*. *input data* akan direpresentasikan ke dalam sebuah matriks yang memiliki panjang dan lebar, diasumsikan sebagai n x m. nilai n merupakan representasi waktu dan m sebagai fitur. Dalam penelitian ini, terdapat 36

fitur atau variabel, sehingga dikenal dengan istilah *multivariate time series*. Selanjutnya, *input data* akan melalui proses konvolusi dengan kernel pada *convolutional layer*. Lebar konvolusi *kernel* akan selalu sama mengikuti jumlah fitur, sedangkan panjang lebih bervariasi. Proses konvolusi akan menghasilkan vektor baru yang diperoleh dari hasil perkalian antara *input data* dengan *kernel*. (Granat, 2019). Konvolusi yang membedakan algoritma CNN dengan algoritma lain, kemampuan *kernel* dalam melakukan ekstrasi fitur. Perpindahan konvolusi dimulai dari arah atas menuju ke mengikuti waktu. Fungsi aktivasi non linear seperti *Rectified Linear Unit* (ReLu) juga ditambahkan, untuk mengembalikan nilai asli jika bernilai positif dan jika bernilai negatif akan diubah menjadi 0 (Lorente et al., 2021).

Hasil vektor baru akan diteruskan pada *Pooling layer*, dalam penelitian ini menggunakan *Max Pooling*. Ekstrasi fitur dilakukan dengan menyeleksi nilai vektor maksimum yang dilalui oleh *pooling filter*. kemudian, nilai vektor maksimum akan diteruskan ke *Fully Connected Layer* untuk dijadikan sebagai hasil pemodelan. Gambar 3.6 merupakan ilustrasi model arsitektur algoritma CNN *time series*.



Gambar 3.6 Model arsitektur CNN

Model arsitektur yang dirancang pada algoritma CNN memanfaatkan Conv1D sebagai *input model*. Data akan diubah ke dalam bentuk 1 dimensi, berisi 64 *unit*, *kernel size* dan fungsi aktivasi. *Filter* konvolusi yang digunakan ialah *MaxPooling1D*. Selanjutnya data akan melalui lapisan *flatten* untuk dapat diteruskan pada *fully connected layer*. Terdapat 2 lapisan *dense* yang berisi 64 *unit* dan 1 *unit* untuk lapisan terakhir yang akan menghasilkan *output model*.

3.5.2 Long Short-Term Memory (LSTM)

Implementasi *Long Short-Term Memory* dilakukan menggunakan *library* keras pada bahasa pemrograman *python*. Setelah data telah siap untuk dilanjutkan ke tahap pemodelan, kemudian data akan melewati lapisan LSTM dan dense secara terurut. Lapisan LSTM berisi 64 unit menerima data sesuai dengan format time series. Kemudian, diteruskan ke lapisan dense sebanyak 64 unit serta lapisan yang berfungsi sebagai keluaran sebanyak 1 unit. Setiap lapisan juga memuat fungsi ReLu yang bertujuan mengubah nilai negatif menjadi 0 dan mempertahankan nilai positif, kecuali pada lapisan terakhir karena hasil akan dijadikan sebagai keluaran model. Terlihat bahwa model arsitektur LSTM lebih sederhana dikarena LSTM memiliki ketiga pintu yang dapat melakukan penyaringan informasi yang cukup baik dan mengenali data sesuai dengan urutan.

3.6 Evaluasi

Tahapan akhir pada penelitian ini adalah Evaluasi. Untuk mengetahui performa suatu model dilakukan proses evaluasi. Proses ini biasanya sangat berkaitan dengan data uji, dimana model akan diberikan data baru untuk mengetahui kemampuan dari suatu model yang dibangun. Salah satu metrik evaluasi yang dapat digunakan dalam kasus *time series forecasting* menggunakan *Mean Square Error* (MSE) (Sabar Sautomo & Hilman Ferdinandus Pardede, 2021). Metrik MSE akan menghitung kuadrat dari selisih antara setiap nilai prediksi dengan nilai sebenarnya, kemudian mengambil rata-rata dari keseluruhan kuadrat kesalahan. Formula MSE dapat dilihat pada persamaan 3.2.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2 \tag{3.2}$$

Dimana :

MSE = hasil nilai mean square error

n = jumlah sampel dalam data

y = nilai sebenarnya dari sampel ke-i

$\hat{y}$ = nilai hasil prediksi untuk sampel ke-i

BAB 4
Hasil dan Pembahasan

4.1 Dataset

Dataset yang digunakan dalam penelitian bersumber dari salah satu repository github yang bersifat *open source*. data yang berasal dari open source dipindahkan ke komputer lokal dengan cara *cloning* melalui alamat url https://github.com/vaastav/Fantasy-Premier-League. Sumber data ini telah banyak dimanfaatkan dalam penelitian lainnya yang berkaitan dengan data pemain FPL. Pada sumber data yang sama, data pemain di musim 2019/2020 digunakan dalam perbandingan model *machine learning* untuk memprediksi performa pemain (Lindberg & Söderberg, 2020). data musim selanjutnya yaitu 2020/2021, penelitian lainnya dilakukan oleh *Ramdas* untuk mengetahui performa pemain FPL menggunakan pendekatan yang biasanya digunakan dalam mengolah gambar (Ramdas, 2022). data musim 2021/2022 merupakan data terbaru pada saat penulis mengunjungi laman sumber dan belum digunakan pada penelitian lainnya sehingga dapat menjadi *state of the art* pada penelitian ini.

Setelah data musim terpilih, didapatkan data statistik pemain untuk setiap pertandingan *(gameweek). Import* data dilakukan menggunakan *library pandas* pada bahasa pemrograman *python*. File yang semula memiliki format .csv *(command-separate values)* akan diubah ke dalam bentuk *dataframe*. Dimensi data berisi 24.565 baris dan 36 kolom seperti yang ditunjukkan pada Gambar 4.1. Baris merepresentasikan jumlah data pemain FPL dan kolom menunjukkan fitur atau variabel. Data tercatat mulai dari pertandingan pertama yang berlangsung pada tanggal 14 Agustus 2021 hingga pertandingan terakhir yang dilaksanakan secara serentak pada tanggal 22 Mei 2022 yang telah didaftarkan oleh 20 klub pada kompetisi EPL. Fitur ‘gw’ memiliki peran penting yang digunakan sebagai penentu waktu dalam perancangan model.

	name	position	team	xP	assists	bonus	bps	clean_sheets	creativity	element	...	team_h_score	threat	total_points	transfers_balance	transfers_in	transfers_out	value	was_home	yellow_cards	GW
0	Eric Bailly	DEF	Man Utd	0.0	0	0	0	0	0.0	286	...	5	0.0	0	0	0	0	50	True	0	1
1	Keinan Davis	FWD	Aston Villa	0.4	0	0	0	0	0.0	49	...	3	0.0	0	0	0	0	45	False	0	1
2	Ayotomiwa Dele-Bashiru	MID	Watford	0.0	0	0	0	0	0.0	394	...	3	0.0	0	0	0	0	45	True	0	1
3	James Ward-Prowse	MID	Southampton	2.3	0	0	20	0	30.5	341	...	3	0.0	2	0	0	0	65	False	0	1
4	Bruno Miguel Borges Fernandes	MID	Man Utd	4.4	0	3	61	0	35.9	277	...	5	59.0	20	0	0	0	120	True	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
24560	Wilfred Ndidi	MID	Leicester	0.0	0	0	0	0	0.0	216	...	4	0.0	0	-202	22	224	48	True	0	38
24561	Matt Ritchie	DEF	Newcastle	0.9	0	0	3	0	0.0	292	...	1	0.0	1	143	396	253	49	False	0	38
24562	Nathan Redmond	MID	Southampton	3.4	0	0	5	0	0.0	336	...	4	0.0	2	455	683	228	59	False	0	38
24563	Mathew Ryan	GK	Brighton	0.0	0	0	0	0	0.0	65	...	3	0.0	0	-2	0	2	45	True	0	38
24564	Ryan Fredericks	DEF	West Ham	0.2	0	0	0	0	0.0	415	...	3	0.0	0	36	103	67	44	False	0	38

24565 rows × 36 columns

Gambar 4.1 Dataset

4.2 Hasil Persiapan data

Pada proses ini akan dilakukan pengolahan data mentah menjadi data bersih. Dalam melakukan pemrosesan data, masih menggunakan *library pandas* yang berperan untuk memanipulasi data. Dataset akan melalui proses pemrosesan data, seleksi fitur, normalisasi dan transformasi data untuk diteruskan ke tahapan pemodelan.

4.2.1 Hasil Pemrosesan Data

Data yang diterima belum semestinya sudah siap dijadikan sebagai masukan pada model sehingga perlu dilakukan pengecekan *missing values* dan *duplicated*. beberapa fungsi dalam *library pandas* seperti *isnull()* digunakan untuk memastikan data memiliki nilai kosong *(null)* atau tidak. Pengecekan *missing value* dilakukan terhadap keseluruhan fitur yang dimiliki oleh data. setiap fitur menghasilkan nilai 0 yang berarti tidak ditemukan nilai *null* pada fitur seperti yang digambarkan pada Gambar 4.2.

```
# check missing values

df.isnull().sum()
```

✓ 0.0s Python

```
name              0
position          0
team              0
xP                0
assists           0
bonus             0
bps               0
clean_sheets      0
creativity        0
element           0
fixture           0
goals_conceded    0
goals_scored      0
ict_index         0
influence         0
kickoff_time      0
minutes           0
opponent_team     0
own_goals         0
penalties_missed  0
penalties_saved   0
red_cards         0
round             0
saves             0
selected          0
...
value             0
was_home          0
yellow_cards      0
GW                0
```

Gambar 4.2 Hasil fungsi isnull()

Penggunaan fungsi *pandas* lainnya untuk mengetahui apakah suatu data memiliki kemiripan atau kesamaan dengan data lainnya yaitu *duplicated().* Jika fungsi *duplicate()* dijalankan maka hanya menghasilkan nilai *true* dan *false* saja. Untuk mengetahui jumlah duplikasi data yang tersedia, perlu menambahkan *.sum()* pada akhir fungsi. Hal ini bertujuan untuk menghitung berapa banyak nilai *true* yang dihasilkan sehingga memperoleh jumlah duplikasi data yang sebenarnya. Hasil dari fungsi yang dijalankan

bernilai 0, menandakan bahwa tidak ditemukan duplikasi data seperti yang diilustrasikan pada Gambar 4.3.

Gambar 4.3 Hasil fungsi duplicated()

Pemrosesan data berikutnya yaitu Pemisahan pemain berdasarkan team masing-masing. Setiap pemain bergabung bersama pemain lainnya jika memiliki nilai yang sama pada fitur ‘team’. Pemisahan data berdasarkan team bertujuan untuk memudahkan dalam proses identifikasi kelengkapan data pemain berdasarkan waktu. Mengingat, dalam pemrosesan model *time series* sangat diperlukan informasi waktu untuk menunjukkan urutan data. Ditemukan anomali data pada salah satu pemain bernama *Ben Davies*, dimana pemain terdaftar pada dua team berbeda seperti yang ditunjukkan pada Gambar 4.4, Fitur ‘team’ terisi oleh Liverpool dan Spurs. Melalui analisis data yang dilakukan terhadap kedua team, diperoleh bahwa data *Ben Davies* pada *team* Spurs lebih bervariasi dibandingkan di team Liverpool. Sedangkan data pada team Liverpool, didominasi nilai 0 pada beberapa fitur dan tidak menghasilkan nilai pada target variabel. Keberagaman variasi data disajikan oleh *team* Spurs menunjukkan representasi data sebenarnya. Kondisi seperti ini diyakini bahwa *Ben Davies* berada pada klub Spurs.

name	position	team	xP	assists	bonus	bps	clean_sheets	creativity	element	...	team_h_score	threat	total_points	transfers_balance	transfers_in	transfers_out	value	was_home	yellow_cards	GW
Ben Davies	DEF	Liverpool	2.0	0	0	0	0	0.0	248	...	0	0.0	0	0	0	0	40	False	0	1
Ben Davies	DEF	Liverpool	0.0	0	0	0	0	0.0	248	...	2	0.0	0	-6199	1080	7279	40	True	0	2
Ben Davies	DEF	Liverpool	0.0	0	0	0	0	0.0	248	...	1	0.0	0	-6159	0	6159	40	True	0	3
Ben Davies	DEF	Liverpool	0.0	0	0	0	0	0.0	248	...	0	0.0	0	-3889	0	3889	40	False	0	4
Ben Davies	DEF	Liverpool	0.0	0	0	0	0	0.0	248	...	3	0.0	0	-1729	0	1729	40	True	0	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Ben Davies	DEF	Spurs	6.5	0	0	25	1	18.5	364	...	0	6.0	6	30972	43432	12460	44	False	0	34
Ben Davies	DEF	Spurs	5.1	0	0	11	0	11.9	364	...	3	18.0	1	32775	42817	10042	45	True	1	35
Ben Davies	DEF	Spurs	5.5	0	0	8	0	0.7	364	...	1	4.0	1	902	17292	16390	45	False	1	36
Ben Davies	DEF	Spurs	5.5	0	0	21	1	13.8	364	...	3	0.0	5	902	17292	16390	45	True	1	36
Ben Davies	DEF	Spurs	5.2	0	0	25	1	0.0	364	...	0	0.0	6	11679	21196	9517	45	False	0	38

Gambar 4.4 Pemisahan data berdasarkan klub

Dilakukan penghapusan data milik *Ben Davies* di *team* Liverpool karena minimnya informasi yang terkandung dan setiap pemain hanya dapat bermain untuk satu *team*. Dalam penelitian ini, variabel ‘GW berperan sebagai penunjuk waktu per pertandingan untuk setiap pemain. Dalam semusim pertandingan FPL dimainkan sebanyak 38 pertandingan. Sehingga nilai pada variabel ‘GW’ semestinya berurut dimulai dari 1 hingga 38. Untuk

mencapai kelengkapan data yang sesuai dengan kebutuhan model, hanya pemain yang memenuhi syarat diteruskan ke proses selanjutnya. Akan tetapi, masih banyak pemain yang memiliki urutan data *time series* tidak berurutan. Pada fitur 'GW' yang ditampilkan pada gambar 4.5 merupakan hasil pemrosesan data sebelumnya, terdapat permasalahan format *time series*. pada fitur yang sama ditemukan bahwa seluruh data pada nilai ke 13, 16, 17 dan 27 tidak tersedia dan nilai ke 26, 30 dan 38 memiliki kesamaan nilai pada beberapa fitur lainnya. Tercapainya format data *time series* dapat dilakukan dengan cara imputasi nilai pada fitur 'GW' yang tidak tersedia. Dalam pembelajaran mesin, nilai dapat diganti menggunakan nilai *modus* atau *mean* (Sudrajat & Cholid, 2023). Namun dalam kasus FPL, nilai fitur 'GW' yang tidak tersedia diartikan sebagai pemain tidak dimainkan dalam suatu pertandingan oleh pelatih karena beberapa faktor. Faktor yang dapat mempengaruhi adalah kondisi kebugaran pemain, pemain mengalami cedera, pemain sedang tidak dalam performa terbaik dan penerapan strategi. Jika pemain tidak dimainkan berarti pemain tidak memiliki kontribusi pada suatu pertandingan maka nilai 0 diberikan kepada seluruh fitur berdasarkan fitur 'GW' yang tidak tersedia untuk melengkapi urutan data *time series*.

name	position	team	xP	assists	bonus	bps	clean_sheets	creativity	element	...	team_h_score	threat	total_points	transfers_balance	transfers_in	transfers_out	value	was_home	yellow_cards	GW
Ben Davies	DEF	Spurs	1.2	0	0	0	0	0.0	364	...	1	0.0	0	0	0	0	45	True	0	1
Ben Davies	DEF	Spurs	0.5	0	0	0	0	0.0	364	...	0	0.0	0	-1804	1094	2898	45	False	0	2
Ben Davies	DEF	Spurs	1.0	0	0	0	0	0.0	364	...	1	0.0	0	-2494	788	3282	45	True	0	3
Ben Davies	DEF	Spurs	0.5	0	0	-1	0	0.1	364	...	3	0.0	0	-1479	1846	3325	44	False	0	4
Ben Davies	DEF	Spurs	0.0	0	0	0	0	0.0	364	...	0	0.0	0	-818	1431	2249	44	True	0	5
Ben Davies	DEF	Spurs	-0.5	0	0	0	0	0.0	364	...	3	0.0	0	-1004	761	1765	44	False	0	6
Ben Davies	DEF	Spurs	0.0	0	0	0	0	0.0	364	...	2	0.0	0	-1104	306	1410	44	True	0	7
Ben Davies	DEF	Spurs	0.4	0	0	0	0	0.0	364	...	2	0.0	0	-2125	34	2159	44	False	0	8
Ben Davies	DEF	Spurs	-0.5	0	0	0	0	0.0	364	...	1	0.0	0	-462	164	626	44	False	0	9
Ben Davies	DEF	Spurs	-0.5	0	0	0	0	3.1	364	...	0	9.0	0	-290	333	623	44	True	1	10
Ben Davies	DEF	Spurs	1.0	0	0	25	1	12.5	364	...	0	3.0	6	83	569	486	44	False	0	11
Ben Davies	DEF	Spurs	2.5	0	0	6	0	0.8	364	...	2	19.0	2	3212	3857	645	44	True	0	12
Ben Davies	DEF	Spurs	5.2	0	0	20	1	1.1	364	...	2	19.0	6	1407	1927	520	44	True	0	14
Ben Davies	DEF	Spurs	8.7	2	3	43	1	27.0	364	...	3	11.0	15	7952	8574	622	44	True	0	15
Ben Davies	DEF	Spurs	5.3	0	0	3	0	0.7	364	...	2	4.0	0	18529	25089	6560	44	True	1	18
Ben Davies	DEF	Spurs	5.7	0	0	0	0	0.0	364	...	3	0.0	0	44968	54381	9413	45	True	0	19
Ben Davies	DEF	Spurs	5.1	0	0	18	0	17.8	364	...	1	8.0	2	-9601	8463	18064	45	False	0	20
Ben Davies	DEF	Spurs	3.0	0	0	27	1	26.3	364	...	0	20.0	6	-4005	12228	16233	45	False	0	21
Ben Davies	DEF	Spurs	2.7	0	0	13	0	13.9	364	...	2	2.0	1	-6562	11761	18323	45	False	0	22
Ben Davies	DEF	Spurs	0.5	0	0	6	0	0.8	364	...	2	0.0	1	-10907	2509	13416	45	False	0	23
Ben Davies	DEF	Spurs	1.5	0	0	23	0	0.4	364	...	2	8.0	1	-712	11548	12260	45	True	0	24
Ben Davies	DEF	Spurs	1.0	0	0	11	0	24.3	364	...	0	4.0	1	-5110	6114	11224	45	True	0	25
Ben Davies	DEF	Spurs	2.9	0	0	15	0	0.6	364	...	2	0.0	1	-12959	4774	17733	45	False	0	26
Ben Davies	DEF	Spurs	2.9	0	0	17	0	3.2	364	...	1	19.0	2	-12959	4774	17733	45	False	0	26
Ben Davies	DEF	Spurs	2.5	0	0	20	1	0.5	364	...	0	33.0	5	-2631	4479	7110	45	False	1	27
Ben Davies	DEF	Spurs	3.2	0	0	26	1	3.0	364	...	5	0.0	6	-10263	9637	19900	44	True	0	28
Ben Davies	DEF	Spurs	7.5	0	0	11	0	2.8	364	...	3	0.0	1	15483	24038	8555	44	False	0	29
Ben Davies	DEF	Spurs	7.5	0	0	28	1	2.9	364	...	0	8.0	6	15483	24038	8555	44	False	0	29
Ben Davies	DEF	Spurs	3.3	0	0	10	0	2.2	364	...	3	0.0	2	5568	9443	3875	44	True	0	30
Ben Davies	DEF	Spurs	4.7	0	0	23	0	2.7	364	...	5	33.0	8	-7055	4209	11264	44	True	0	31
Ben Davies	DEF	Spurs	6.0	0	0	22	1	0.9	364	...	0	2.0	6	9481	13436	3955	44	False	0	32
Ben Davies	DEF	Spurs	5.8	0	0	12	0	1.0	364	...	0	0.0	2	32445	38347	5902	44	True	0	33
Ben Davies	DEF	Spurs	6.5	0	0	25	1	18.5	364	...	0	6.0	6	30972	43432	12460	44	False	0	34
Ben Davies	DEF	Spurs	5.1	0	0	11	0	11.9	364	...	3	18.0	1	32775	42817	10042	45	True	1	35
Ben Davies	DEF	Spurs	5.5	0	0	8	0	0.7	364	...	1	4.0	1	902	17292	16390	45	False	1	36
Ben Davies	DEF	Spurs	5.5	0	0	21	1	13.8	364	...	3	0.0	5	902	17292	16390	45	True	1	36
Ben Davies	DEF	Spurs	5.2	0	0	25	1	0.0	364	...	0	0.0	6	11679	21196	9517	45	False	0	38

Gambar 4.5 Data kotor ben davies

Untuk menangani kesamaan nilai pada fitur ‘GW’, nilai pertama pada kondisi ini akan dipertahankan dan nilai selanjutnya akan dibuang. Nilai kemunculan pertama menandakan bahwa sistem telah mencatat data pemain terlebih dahulu pada gameweek tersebut. jika penghapusan data dilakukan pada kedua nilai yang sama maka akan menghilangkan urutan data *time series* serta mengurangi jumlah data, sehingga dalam penelitian ini hanya membuang salah satu nilai dari kondisi tersebut. hasil pemrosesan berisi data setiap pemain yang memiliki format *time series* secara berurutan sesuai jumlah pertandingan dalam semusim seperti ditunjukkan pada Gambar 4.6 diwakilkan oleh *Ben Davies.*

	name	position	team	xP	assists	bonus	bps	clean_sheets	creativity	element	...	team_h_score	threat	total_points	transfers_balance	transfers_in	transfers_out	value	was_home	yellow_cards	GW
0	Ben Davies	DEF	Spurs	1.2	0	0	0	0	0.0	364	...	1	0.0	0	0	0	0	45	1	0	1
1	Ben Davies	DEF	Spurs	0.5	0	0	0	0	0.0	364	...	0	0.0	0	-1804	1094	2898	45	0	0	2
2	Ben Davies	DEF	Spurs	1.0	0	0	0	0	0.0	364	...	1	0.0	0	-2494	788	3282	45	1	0	3
3	Ben Davies	DEF	Spurs	0.5	0	0	-1	0	0.1	364	...	3	0.0	0	-1479	1846	3325	44	0	0	4
4	Ben Davies	DEF	Spurs	0.0	0	0	0	0	0.0	364	...	0	0.0	0	-818	1431	2249	44	1	0	5
5	Ben Davies	DEF	Spurs	-0.5	0	0	0	0	0.0	364	...	3	0.0	0	-1004	761	1765	44	0	0	6
6	Ben Davies	DEF	Spurs	0.0	0	0	0	0	0.0	364	...	2	0.0	0	-1104	306	1410	44	1	0	7
7	Ben Davies	DEF	Spurs	0.4	0	0	0	0	0.0	364	...	2	0.0	0	-2125	34	2159	44	0	0	8
8	Ben Davies	DEF	Spurs	-0.5	0	0	0	0	0.0	364	...	1	0.0	0	-462	164	626	44	0	0	9
9	Ben Davies	DEF	Spurs	-0.5	0	0	0	0	3.1	364	...	0	9.0	0	-290	333	623	44	1	1	10
10	Ben Davies	DEF	Spurs	1.0	0	0	25	1	12.5	364	...	0	3.0	6	83	569	486	44	0	0	11
11	Ben Davies	DEF	Spurs	2.5	0	0	6	0	0.8	364	...	2	19.0	2	3212	3857	645	44	1	0	12
12	Ben Davies	DEF	Spurs	0.0	0	0	0	0	0.0	0	...	0	0.0	0	0	0	0	0	0	0	13
13	Ben Davies	DEF	Spurs	5.2	0	0	20	1	1.1	364	...	2	19.0	6	1407	1927	520	44	1	0	14
14	Ben Davies	DEF	Spurs	8.7	2	3	43	1	27.0	364	...	3	11.0	15	7952	8574	622	44	1	0	15
15	Ben Davies	DEF	Spurs	0.0	0	0	0	0	0.0	0	...	0	0.0	0	0	0	0	0	0	0	16
16	Ben Davies	DEF	Spurs	0.0	0	0	0	0	0.0	0	...	0	0.0	0	0	0	0	0	0	0	17
17	Ben Davies	DEF	Spurs	5.3	0	0	3	0	0.7	364	...	2	4.0	0	18529	25089	6560	44	1	1	18
18	Ben Davies	DEF	Spurs	5.7	0	0	0	0	0.0	364	...	3	0.0	0	44968	54381	9413	45	1	0	19
19	Ben Davies	DEF	Spurs	5.1	0	0	18	0	17.8	364	...	1	8.0	2	-9601	8463	18064	45	0	0	20
20	Ben Davies	DEF	Spurs	3.0	0	0	27	1	26.3	364	...	0	20.0	6	-4005	12228	16233	45	0	0	21
21	Ben Davies	DEF	Spurs	2.7	0	0	13	0	13.9	364	...	2	2.0	1	-6562	11761	18323	45	0	0	22
22	Ben Davies	DEF	Spurs	0.5	0	0	6	0	0.8	364	...	2	0.0	1	-10907	2509	13416	45	0	0	23
23	Ben Davies	DEF	Spurs	1.5	0	0	23	0	0.4	364	...	2	8.0	1	-712	11548	12260	45	1	0	24
24	Ben Davies	DEF	Spurs	1.0	0	0	11	0	24.3	364	...	0	4.0	1	-5110	6114	11224	45	1	0	25
25	Ben Davies	DEF	Spurs	2.9	0	0	17	0	3.2	364	...	1	19.0	2	-12959	4774	17733	45	0	0	26
26	Ben Davies	DEF	Spurs	2.5	0	0	20	1	0.5	364	...	0	33.0	5	-2631	4479	7110	45	0	1	27
27	Ben Davies	DEF	Spurs	3.2	0	0	26	1	3.0	364	...	5	0.0	6	-10263	9637	19900	44	1	0	28
28	Ben Davies	DEF	Spurs	7.5	0	0	11	0	2.8	364	...	3	0.0	1	15483	24038	8555	44	0	0	29
29	Ben Davies	DEF	Spurs	3.3	0	0	10	0	2.2	364	...	3	0.0	2	5568	9443	3875	44	1	0	30
30	Ben Davies	DEF	Spurs	4.7	0	0	23	0	2.7	364	...	5	33.0	8	-7055	4209	11264	44	1	0	31
31	Ben Davies	DEF	Spurs	6.0	0	0	22	1	0.9	364	...	0	2.0	6	9481	13436	3955	44	0	0	32
32	Ben Davies	DEF	Spurs	5.8	0	0	12	0	1.0	364	...	0	0.0	2	32445	38347	5902	44	1	0	33
33	Ben Davies	DEF	Spurs	6.5	0	0	25	1	18.5	364	...	0	6.0	6	30972	43432	12460	44	0	0	34
34	Ben Davies	DEF	Spurs	5.1	0	0	11	0	11.9	364	...	3	18.0	1	32775	42817	10042	45	1	1	35
35	Ben Davies	DEF	Spurs	5.5	0	0	8	0	0.7	364	...	1	4.0	1	902	17292	16390	45	0	1	36
36	Ben Davies	DEF	Spurs	0.0	0	0	0	0	0.0	0	...	0	0.0	0	0	0	0	0	0	0	37
37	Ben Davies	DEF	Spurs	5.2	0	0	25	1	0.0	364	...	0	0.0	6	11679	21196	9517	45	0	0	38

Gambar 4.6 Hasil pemrosesan data

4.2.2 Hasil Seleksi Fitur

Tahapan selanjutnya pada pembersihan data adalah seleksi fitur. Diketahui jumlah kolom (fitur) yang terkandung dalam dataset berjumlah 36. Jumlah yang dinilai cukup besar sehingga dapat berpengaruh pada waktu komputasi dan performa model. Dalam penelitian ini, seleksi fitur dicapai menggunakan metode lasso. Masing-masing posisi pemain akan menghasilkan jumlah dan fitur berbeda. Langkah awal ialah import *library scikit learn* pada bahasa pemrograman *python* yang biasa digunakan dalam pembuatan model. Gambar

4.7 memuat fungsi lasso, *train_test_split*, *GridSearchCV*, dan *KFold* yang dibutuhkan untuk seleksi fitur.

```
# For Feature Selection
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.linear_model import Lasso
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV, KFold
```

Gambar 4.7 Import library

kolom *total_points* yang digunakan sebagai target variabel akan dipisahkan dengan kolom lainnya. dalam prakteknya, target variabel akan disimpan pada variabel y. sedangkan kolom atau fitur lainnya disimpan pada variabel x memanfaatkan fungsi *.drop()* untuk menghapus target variabel dalam data. Pada Gambar 4.8 merupakan contoh *define* variabel x dan y pada python berdasarkan jenis variable yang diwakili oleh data pemain belakang *(defender)*.

```
# Define x and y in DEF data

X_def = deff.drop('total_points', axis=1).values
y_def = deff['total_points'].values
```

Python

Gambar 4.8 Define variabel

Implementasi *train_test_split* berfungsi untuk membagi data menjadi data latih dan uji. fungsi pada *python* yang ditunjukkan pada Gambar 4.9 berisi variabel x dan y yang telah dipisahkan berdasarkan jenis variabelnya. Jumlah data uji diatur menggunakan parameter *test_size* sebesar 0.2, berarti 20% dari data akan digunakan sebagai pengujian. Data acak akan selalu tetap ketika nilai *random state* adalah 42.

```
# Train Test Split

X_def_train, X_def_test, y_def_train, y_def_test = train_test_split(X_def, y_def, test_size=0.2, random_state=42)
```

Gambar 4.9 Train_test_split pada seleksi fitur

Metode lasso memiliki parameter yang diperlukan untuk menghasilkan fitur-fitur terbaik dari data masukan. optimasi parameter dalam penelitian dilakukan menggunakan *GridSearchCV* dari *library scikit learn*. Inisialisasi *GridSearchCV* berisi beberapa nilai parameter alpha serta *kfold* sebagai evaluasi kinerja. Nilai alpha yang digunakan

diantaranya ialah 0.001, 0.01, dan 01. selain itu, *kfold* bertugas membagi data ke dalam 5 *subset* dengan *random_state* bernilai 42 seperti yang ditunjukkan pada Gambar 4.10. semakin besar nilai alpha maka dampak penalti regularisasi L1 yang diberikan semakin besar.

```python
# set parameters to be tested on GridSearchCV
params = {'alpha':np.arange(0.001, 0.01, 0.1)}

# Number of folds and adding the random state for replication
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Initializing the model
lasso = Lasso()

# GridSearchCV with the model, params and folds.
lasso_cv_def = GridSearchCV(lasso, param_grid=params, cv=kf)
lasso_cv_def.fit(X_def, y_def)
```

Python

Gambar 4.10 Inisialisasi GridSearchCV

Diperoleh bahwa nilai alpha terbaik yang dihasilkan oleh *GridSearchCV* bernilai 0,001. Nilai alpha terbaik dijadikan sebagai inputan parameter regresi lasso pada data latih dan data uji menggunakan fungsi .fit() seperti pada Gambar 4.11. Hasilnya berupa nilai koefisien yang diubah ke dalam bentuk absolut, nilai koefisien negatif akan diubah menjadi nilai positif menggunakan fungsi .abs pada *library numpy*.

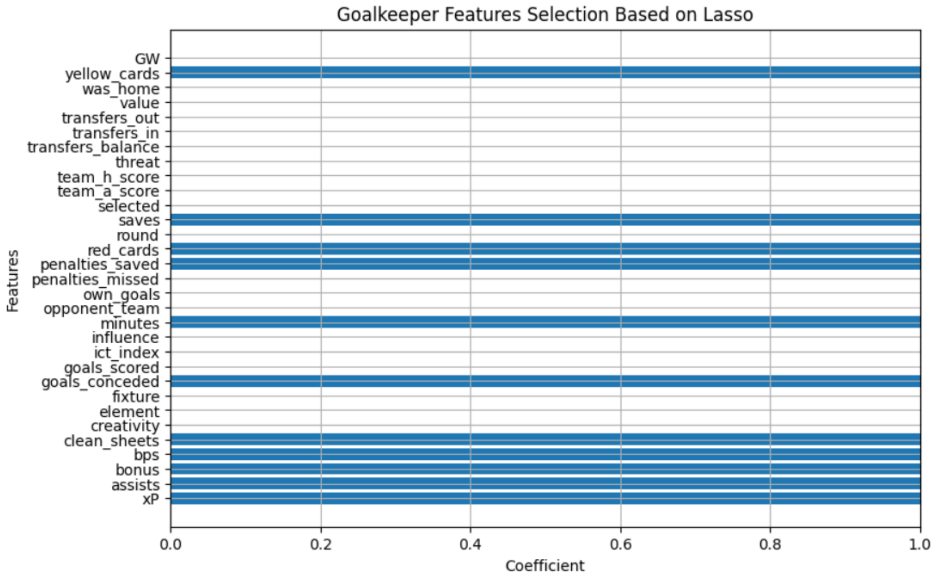
```python
# calling the model with the best parameter
lasso_def = Lasso(alpha=0.001)
lasso_def.fit(X_def_train, y_def_train)

# Using np.abs() to make coefficients positive
lasso_def_coef = np.abs(lasso_def.coef_)
```

Python

Gambar 4.11 Model lasso fit

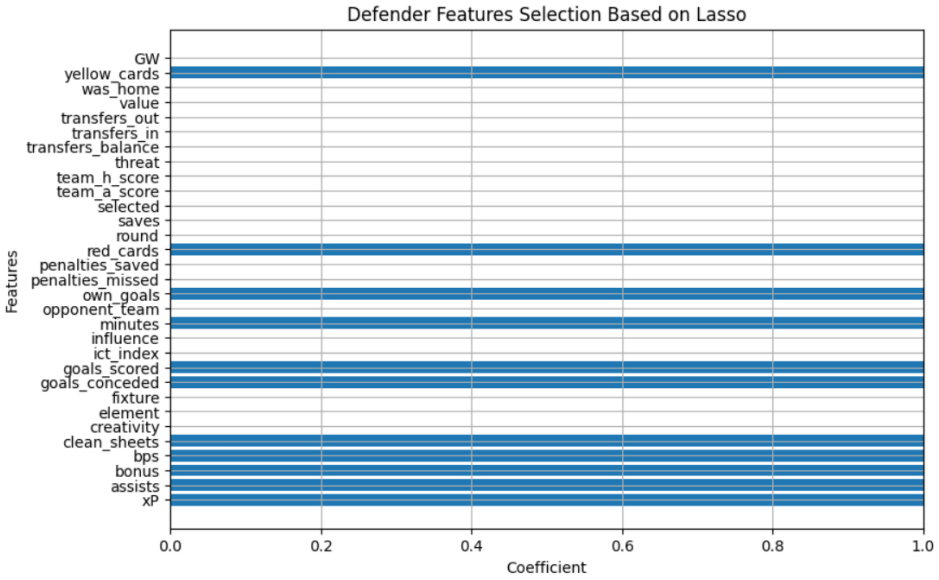
Secara teori telah dideskripsikan pada subbab 2.3.3, bahwa metode lasso akan memanfaatkan regularisasi untuk menyusutkan nilai koefien mendekati nilai 0. Jika nilai koefisien yang dihasilkan setiap fitur berada diatas 0, fitur memiliki pengaruh signifikan dan perlu dipertahankan. Akan tetapi dalam studi kasus ini, *threshold* yang digunakan adalah 0,01. Penggunaan *threshold* secara *default*, diperoleh masih banyak fitur yang berhasil lolos dalam proses ini sehingga peningkatan *threshold* dilakukan agar lebih selektif dalam mendapatkan fitur signifikan terhadap model di setiap posisi. Kemudian, nilai koefisien seluruh fitur akan divisualisasikan ke dalam bentuk *bar chart horizontal* pada masing-masing posisi.

Diperoleh hasil seleksi fitur pada posisi *goalkeeper* berisi fitur-fitur seperti *xP*, *assists*, *bonus*, *bps*, *clean_sheets*, *goals_conceded*, *minutes*, *penalties_saved*, *red_cards*, *saves*, dan *yellow_cards*. *Saves* dan *penalties_saved* merupakan dua satu fitur terpilih yang hanya dimiliki oleh posisi ini. *goalkeeper* merupakan pemain terakhir yang dapat mencegah terjadinya gol dalam suatu klub agar terhindar dari kekalahan. pada fitur lainnya seperti *penalties_saved* dinilai sangat penting apabila seorang *goalkeeper* dapat mengagalkan tendangan pinalti oleh lawan. Secara sistem poin pada game FPL, *penalties_saved* menghasilkan 5 poin sedangkan *saves* menghasilkan 1 poin terhitung apabila goalkeeper telah melakukan penyelamatan sebanyak 3x. Pada Gambar 4.12 disajikan hasil seleksi fitur pada posisi *goalkeeper* berjumlah 11 fitur.



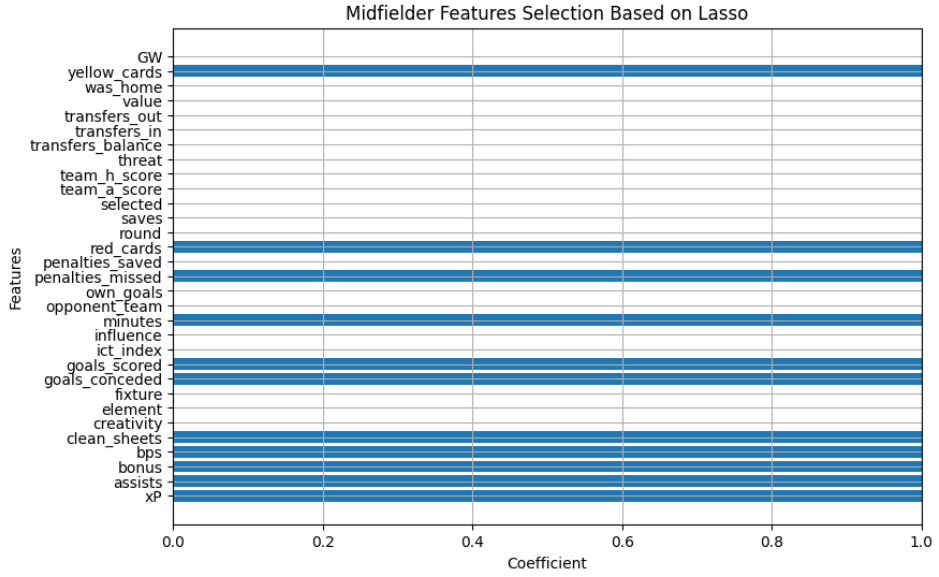
Gambar 4.12 Hasil seleksi fitur goalkeeper

Dilihat dari segi fungsionalitas, posisi *defender* memiliki tugas dan tanggung jawab sebagai barisan pertama untuk mencegah serangan oleh pihak lawan. fitur terpilih akan berfokus pada aspek pertahanan seperti *goal_conceded* dan *clean_sheets*, dimana sangat kuat untuk merepresentasikan posisi ini. Akan tetapi, posisi *defender* juga tidak menutup kemungkinan untuk dapat berpartisipasi ke dalam proses penyerangan bahkan menghasilkan gol pada *goals_scored*. Model lasso menilai bahwa faktor-faktor yang berpotensi merugikan klub seperti *yellow cards*, *red_cards* dan *own_goals* juga dapat mempengaruhi model. secara keseluruhan fitur-fitur terpilih pada posisi *defender* ditunjukkan pada Gambar 4.13.



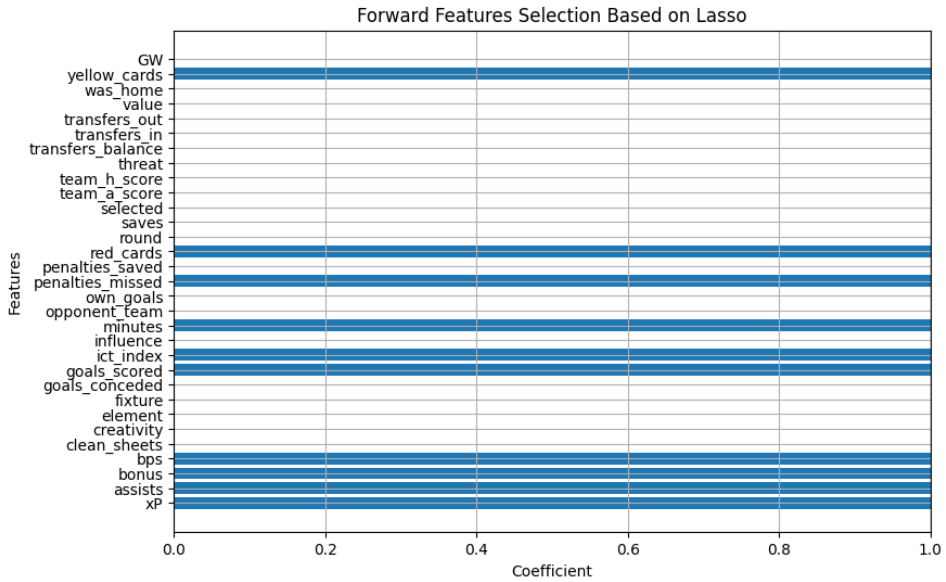
Gambar 4.13 Hasil seleksi fitur defender

Hasil seleksi fitur menggunakan data pemain berposisi *midfielder* diantaranya adalah *xP, assists, bonus, bps, goals_scored, ict_index, minutes, penalties_missed, red_cards, dan yellow_cards*. nilai koefisien fitur yang mencapai *threshold* pada metode lasso digambarkkan dengan *bar chart horizontal* berwarna biru pada Gambar 4.14. nama-nama fitur pada sumbu y, yang tidak mencapai *threshold* akan dibuang karena dinilai tidak memiliki kontribusi dalam pengembangan model.



Gambar 4.14 Hasil seleksi fitur midfielder

Pemain berposisi *forward* memiliki area permainan paling depan dalam sebuah tim. *forward* sebagai pemain paling depan memiliki tugas untuk menciptakan gol agar sebuah tim meraih kemenangan. Hal ini selaras dengan hasil seleksi fitur yang dilakukan menggunakan metode lasso, dimana fitur terpilih seperti *goal_scored* menampilkan jumlah gol yang diciptakan oleh seorang pemain. fitur *ict_index* yang didapatkan dari kombinasi penilaian sistem FPL dapat melewati proses seleksi fitur. Hasil seleksi fitur pada posisi *forward* berjumlah 10 fitur, jumlah fitur paling sedikit jika dibandingkan dengan posisi lainnya. Terlihat pada Gambar 4.15 merupakan nama-nama fitur yang mempunyai nilai koefisien diatas *threshold*.



Gambar 4.15 hasil seleksi fitur forward

Hasil seleksi fitur memuat jumlah dan fitur yang berbeda untuk setiap posisi. Fitur terpilih ditentukan oleh nilai koefisien yang dihasilkan oleh Lasso. Konsistensi fitur *assists, bonus, ict_index, own_goals, yellow_cards* dan *red_cards* yang selalu hadir di setiap posisi dinilai menjadi fitur penting dalam penelitian ini. fitur lain seperti *name* dan *gw* ditambahkan untuk mengenali pola *time series* dari masing-masing pemain. sedangkan *total_points* juga ditambahkan sebagai target penelitian.

4.2.3 Hasil Normalisasi

Fitur yang telah melewati tahapan seleksi fitur kemudian akan dilakukan normalisasi. Beragam variasi dihasilkan melalui statistik deskriptif pada setiap fitur yang nantinya akan mempengaruhi kinerja model. rentang nilai berbeda di setiap fitur akan diubah agar

mempunyai rentang nilai yang sama. normalisasi *min-max* dapat mengubah seluruh nilai ke dalam rentang nilai 0 hingga 1. Proses pengubahan rentang nilai didapatkan melalui hasil pembagian antara selisih dari nilai pada suatu fitur dengan nilai minimum pada fitur dan selisih dari nilai maksimum fitur dan nilai minimum fitur. hasil normalisasi pada fitur minutes seperti pada Gambar 4.16 memiliki nilai lebih besar dibandingkan fitur lainnya karena pemain diberikan waktu bermain lebih lama. Jika nilai berada pada angka 1 berarti pemain mendapatkan waktu bermain secara penuh dalam suatu pertandingan sekitar 90 menit.

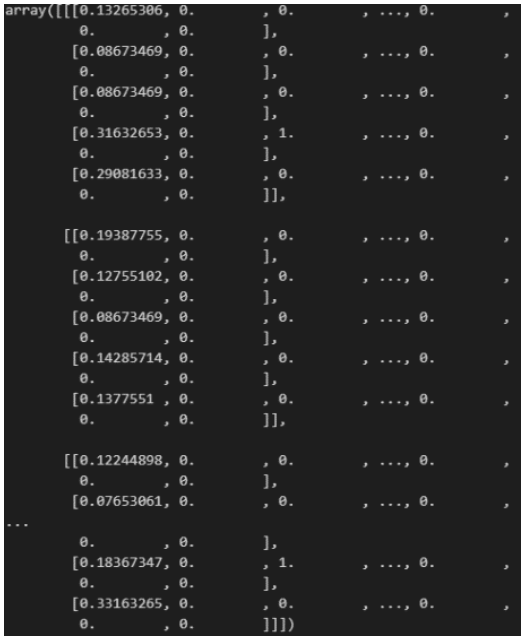
	xP	assists	bonus	bps	goals_scored	ict_index	minutes	penalties_missed	red_cards	yellow_cards	total_points
0	0.132653	0.0	0.0	0.111111	0.00	0.000000	0.000000	0.0	0.0	0.0	0.076923
1	0.086735	0.0	0.0	0.145833	0.00	0.079365	0.322222	0.0	0.0	0.0	0.115385
2	0.086735	0.0	0.0	0.111111	0.00	0.003175	0.644444	0.0	0.0	0.0	0.115385
3	0.316327	0.0	1.0	0.319444	0.25	0.511111	1.000000	0.0	0.0	0.0	0.423077
4	0.290816	0.0	0.0	0.152778	0.00	0.200000	1.000000	0.0	0.0	0.0	0.153846
...	...	...	...	...	...	...	...	...	...	...	...
2807	0.188776	0.0	0.0	0.187500	0.00	0.228571	1.000000	0.0	0.0	0.0	0.153846
2808	0.163265	0.0	0.0	0.159722	0.00	0.177778	1.000000	0.0	0.0	1.0	0.115385
2809	0.122449	0.0	0.0	0.131944	0.00	0.073016	0.222222	0.0	0.0	0.0	0.115385
2810	0.076531	0.0	0.0	0.111111	0.00	0.000000	0.000000	0.0	0.0	0.0	0.076923
2811	0.102041	0.0	0.0	0.083333	0.00	0.000000	0.755556	0.0	0.0	0.0	0.153846

Gambar 4.16 Hasil normalisasi

4.2.4 Hasil Transformasi Data

Transformasi data dilakukan untuk mengubah dimensi data ke dalam bentuk yang dapat dikenali oleh sebuah model. Pada model *time series*, data akan berkaitan dengan waktu. Dalam penelitian ini, data waktu akan di representasikan oleh fitur *gw*. Hal ini dikarenakan total pertandingan yang dilalui setiap klub FPL akan dihitung dari minggu pertama ke minggu berikutnya hingga akhir musim melalui fitur *gw*. Sehingga data historis pertandingan sebelumnya dapat digunakan untuk melakukan prediksi para pertandingan selanjutnya.

Penggunaan kedua algoritma yang telah dibahas sebelumnya, data perlu dilakukan transformasi sesuai dengan data masukan yang dapat diterima oleh model. data setiap pemain akan diubah mengikuti langkah waktu yaitu 5. Langkah waktu *(timesteps)* dalam penelitian ini berarti akan mengambil data setiap pemain sebanyak 5 pertandingan yang dimulai dari pertandigan pertama, kedua, ketiga hingga kelima. Data historis ini akan dibentuk ke dalam sebuah *array* yang setiap baris nya akan merepresentasikan pertandingan ke 1 hingga 5. Sedangkan kolom berisi fitur-fitur yan telah terpilih. Pada Gambar 4.17 merupakan hasil transformasi data pemain yang berposisi sebagai *forward*.



Gambar 4.17 Hasil transformasi data

4.3 Hasil Pembagian Data

Data bersih selanjutnya akan dibagi menjadi dua bagian menjadi data latih dan data uji. pembagian data menggunakan *train_test_split* pada *library sklearn* dengan mengatur *test_size* untuk menentukan jumlah data uji. Dalam penelitian ini pembagian data akan menggunakan dua skenario berbeda untuk mengetahui pengaruh performa model terhadap jumlah data yang diberikan. Skenario pembagian data pertama, nilai *test_size* adalah 0.3 dan *random_state* adalah 42 seperti yang ditunjukkan pada Gambar 4.18.

```
# train test split data

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Gambar 4.18 *Train test split* skenario pertama

Skenario pembagian data kedua, nilai parameter *test_size* dan *random_size* pada *train_test_split* yaitu 0.2 dan 42. Pada Gambar 4.19 dilakuan set parameter untuk mengatur jumlah data uji yang digunakan sebanyak 20% dari total dataset. Sedangkan *random_state* berfungsi untuk mengontrol data latih dan data uji agar selalu sama pada saat pembagian data berlangsung.

```
# train test split data

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Gambar 4.19 *Train test split* skenario kedua

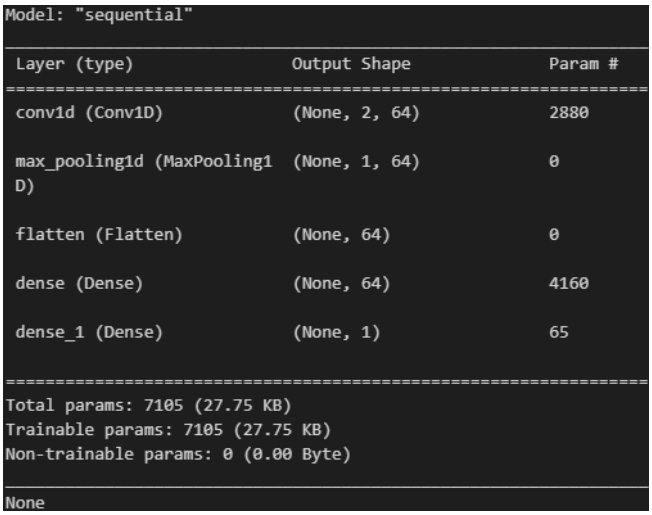
4.4 Hasil Model

Pada bagian ini akan mendeskripsikan hasil dari pemodelan dalam penelitian. implementasi model memanfaatkan algoritma *deep learning*, yang dinilai dapat mengatasi permasalahan *time series* dan memiliki performa model bagus untuk melakukan prediksi. Algoritma *deep learning* yang digunakan ialah *Convolutional Neural Network* dan *Long Short-Term Memory*. Kode program perlu menambahkan *library keras* untuk mengembangkan kedua algoritma *deep learning*. Dalam penelitian ini, model akan dibedakan berdasarkan posisi dengan penggunaan skenario pembagian data. model arsitektur yang dibangun untuk setiap algoritma berbeda, maka model arsitektur adalah sebagai berikut.

4.4.1 Hasil Model Convolutional Neural Network

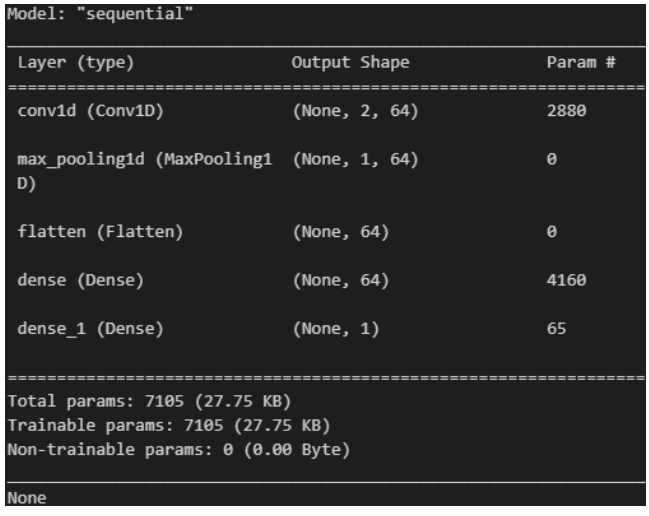
Implementasi model CNN akan memiliki 4 buah model arsitektur yang setiap dataset mempunyai model arsitektur yang sama. Data bersih yang telah siap dijadikan sebagai data masukan pada model, akan diteruskan ke lapisan-lapisan yang dimiliki oleh model CNN.

Pada Gambar 4.20 merupakan model arsitekur pada pemain berposisi sebagai *goalkeeper*. Lapisan pertama yang berfungsi sebagai lapisan masukan menggunakan *conv1D*. *Max pooling 1D* ditambahkan untuk mereduksi dimensi data kemudian dilakukan *flatten* untuk menghubungkan lapisan sebelumnya dengan lapisan *dense*. Sedangkan parameter yang dioptimalkan pada lapisan *dense* sebanyak 4160 yang akan diteruskan pada lapisan keluaran 1 dimensi.



Gambar 4.20 Model arsitektur CNN *goalkeeper*

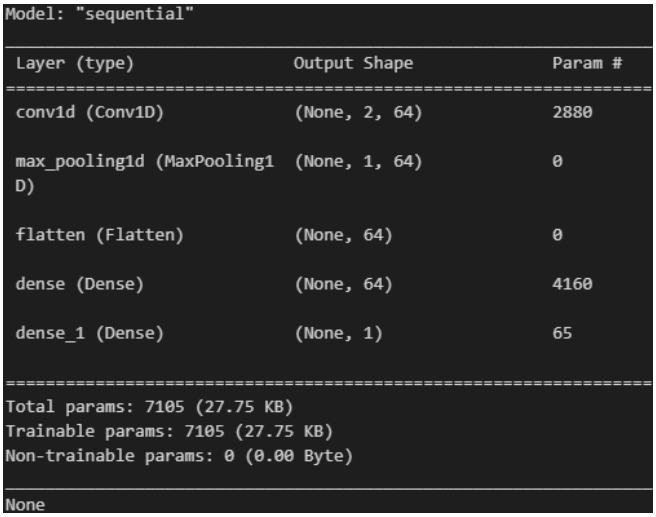
Model arsitektur *defender* terdiri dari sebuah lapisan *conv1D*, *max pooing1D*, *flatten*, dan *dense*. Set parameter pada lapisan *conv1D* menggunakan 64 *unit* dan ukuran kernel sebesar 4 untuk tujuan konvolusi. Selanjutnya, lapisan *max pooling* akan memilih nilai maksimum yang secara *default* pada lapisan ini bernilai 2. Penggunaan lapisan *dense* sebanyak 64 *unit* yang menambahkan fungsi aktivasi *ReLu* akan diteruskan ke lapisan akhir yang berisi 1 *unit* sebagai keluaran model. Pada Gambar 4.21 merupakan model arsitektur *defender*.



Gambar 4.21 Model arsitektur CNN *defender*

Model arsitektur selanjutnya yaitu *midfielder*. Model yang memiliki dataset paling banyak dibandingkan model lain, juga memiliki kemiripan pada model arsitektur CNN.

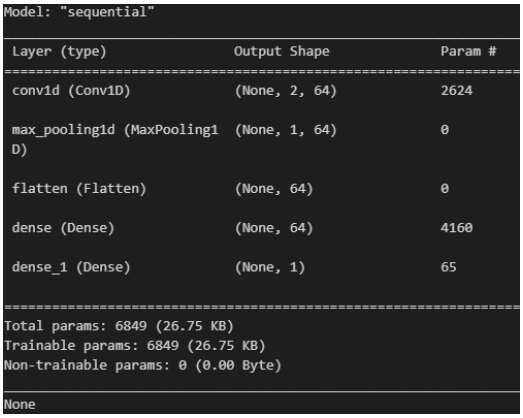
pada Gambar 4.22 menggambarkan model arsitektur *midfielder* yang seluruh komponen atau lapisan memiliki kesamaan dengan model arsitektur sebelumnya.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 2, 64)             2880

 max_pooling1d (MaxPooling1  (None, 1, 64)             0
 D)

 flatten (Flatten)           (None, 64)                0

 dense (Dense)               (None, 64)                4160

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 7105 (27.75 KB)
Trainable params: 7105 (27.75 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

Gambar 4.22 Model arsitektur CNN *midfielder*

Sedangkan model yang memiliki dataset paling sedikit memiliki lapisan-lapisan yang terdiri dari komponen yang sama juga. Namun, jumlah parameter optimal yang dihasilkan pada lapisan *conv1D* berbeda dibandingkan dengan model pada posisi lainnya. parameter optimal yang dihasilkan pada model arsitektur *forward* yaitu 2624. Jumlah yang lebih kecil dibandingkan dengan jumlah parameter optimal sebelumnya yang menghasilkan 2880 dapat dilihat pada Gambar 4.23.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 2, 64)             2624

 max_pooling1d (MaxPooling1  (None, 1, 64)             0
 D)

 flatten (Flatten)           (None, 64)                0

 dense (Dense)               (None, 64)                4160

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 6849 (26.75 KB)
Trainable params: 6849 (26.75 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

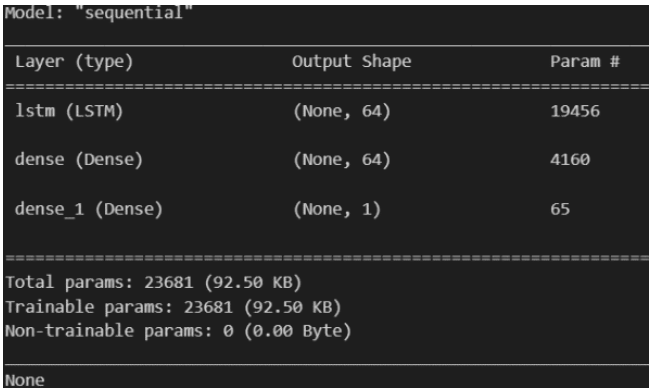
Gambar 4.23 Model arsitektur CNN *forward*

Model arsitektur yang dirancang memiliki lapisan dan parameter yang sejenis. Penggunaan lapisan *conv1D, maxpooling, flatten dan dense* dalam perancangan model,

setiap lapisan akan mempunyai 64 *unit* kecuali pada lapisan keluaran. Fungsi aktivasi *ReLu* juga dikombinasikan pada lapisan *conv1D* dan *dense*. Pelatihan model akan dilakukan menggunakan data latih yang telah dipisahkan variabel dan target. *Epoch* yang digunakan sebesar 50.

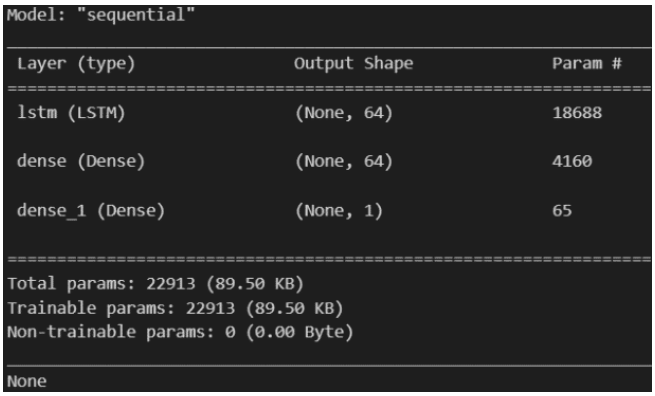
4.4.2 Hasil Model Long Short-Term Memory

Model *Long Short-Term Memory* memiliki model arsitektur sederhana dari model CNN. Kemampuan ketiga pintu dinilai sangat baik dalam menangani kasus *time series*. Model arsitektur LSTM ini terdiri dari sebuah lapisan LSTM dan dua buah lapisan *dense* yang salah satunya merupakan lapisan keluaran. Secara keseluruhan, model arsitektur yang diterapkan ke 4 bagian data memiliki nilai parameter yang sama. Lapisan LSTM dan *dense* pada dua lapisan awal mempunyai 64 *unit*, diikuti fungsi aktivasi *ReLu*. Nilai *epoch* yang digunakan ialah 50. Pada Gambar 4.24 merupakan model arsitektur yang dirancang menggunakan data *goalkeeper*.



Gambar 4.24 Model arsitektur LSTM *goalkeeper*

Untuk data pemain yang berposisi *defender* dan *midfielder* juga memiliki *summary model* yang sama dengan *goalkeeper*. Mulai dari *layer*, *output shape* dan *param* yang dihasilkan tidak ditemukan perbedaan nilai. Perbedaan muncul ketika model arsitektur LSTM menggunakan dataset *forward*. Dimana jumlah parameter optimal yang dihasilkan pada lapisan LSTM lebih rendah dari ketiga model sebelumnya. Pada Gambar 4.25 merupakan model arsitektur LSTM *forward* yang memanfaatkan lapisan LSTM sebagai lapisan masukan.



Gambar 4.25 Model arsitektur LSTM *forward*

4.5 Hasil Evaluasi

Nilai evaluasi akan merepresentasikan performa dari suatu model yang dirancang. Metrik evaluasi pada penelitian ini menggunakan *Mean Square Error* (MSE) yang telah dibahas pada bagian 3.6. Perbandingan model dilakukan untuk mengetahui algoritma terbaik yang dapat menghasilkan model prediksi optimal dalam penelitian ini.

4.5.1 Hasil Perbandingan Model

Perbandingan model dilakukan antara dua algoritma *deep learning* yang telah dirancang pada tahap pemodelan. Tabel 4.1 merupakan hasil perbandingan evaluasi model prediksi dengan menggunakan pembagian dataset skenario pertama. Algoritma LSTM menghasilkan nilai rata-rata lebih kecil sebesar 0.0027 sedangkan algoritma CNN sebesar 0.0052. Setiap model yang dibedakan berdasarkan posisi, algoritma LSTM selalu menggunguli CNN dengan nilai *error* optimal dimulai dari posisi *goalkeeper* hingga *forward* diantarnya ialah 0.0081, 0.0008, 0.0002, dan 0.00020. Jika dibandingkan dengan model CNN maka nilai *error* secara terurut ialah 0.0098, 0.0035, 0.0012, dan 0.0063. LSTM menghasilkan nilai evaluasi lebih rendah dibandingkan CNN dari hasil rata-rata nilai evaluasi pemain berdasarkan posisi. LSTM mampu menghasilkan nilai evaluasi sebesar 0.0027, dimana algoritma terbaik pada skenario pertama. Evaluasi model pada posisi goalkeeper memiliki nilai evaluasi tertinggi diantara posisi lainnya, mengingat jumlah data yang dimiliki tidak mencapai setengah dari jumlah data pada posisi *midfielder*. Sedangkan evaluasi model pada posisi *midfielder* merupakan yang terbaik diantara posisi lainnya, nilai evaluasi model terendah yaitu 0.0002.

Tabel 4.1 Perbandingan evaluasi model skenario pertama

Position	**CNN**	**LSTM**
Goalkeeper	0.0098	0.0081
Defender	0.0035	0.0008
Midfielder	0.0012	0.0002
Forward	0.0063	0.0020
Average	0,0052	0,0027

Pada skenario lainnya dengan rasio pembagian data yang digunakan sebesar 80% data latih dan 20% data uji akan menghasilkan nilai evaluasi model menggunakan metrik MSE seperti ditunjukkan pada Tabel 4.2. Berbeda dengan skenario pertama, seluruh model LSTM sangat mendominasi. Model CNN dapat menggunguli model LSTM pada posisi *goalkeeper* di skenario kedua, dengan perbandingan nilai evaluasi sebesar 0.0040 untuk CNN dan 0.0060 untuk LSTM. Namun keunggulan ini tidak bertahan lama dikarenakan model CNN tidak dapat mempertahankan nilai minimum *error* di ketiga model lainnya yaitu *defender*, *midfielder* dan *forward*. Dimana model LSTM menghasilkan nilai *error* yang lebih kecil sehingga performa model dapat meningkat. Adapun hasil evaluasi model terbaik pada ketiga posisi lainnya secara terurut diantaranya ialah 0.0003 untuk *defender* dan *midfielder*, 0.0022 untuk *forward*. Algoritma LSTM unggul di 3 posisi sedangkan CNN hanya meraih satu hasil terbaik pada posisi *goalkeeper*. Nilai rata-rata evaluasi model yang dihasilkan antara kedua algoritma ini adalah 0.0027 pada model CNN dan 0.0025 untuk LSTM seperti ditunjukkan pada Tabel 4.2. Penggunaan data latih yang lebih banyak dalam proses pelatihan dapat memperkecil hasil evaluasi model pada kedua algoritma. CNN mengalami peningkatan secara signifikan dibandingkan dengan skenario sebelumnya, terlihat dari perbedaan rata-rata nilai evaluasi yang sangat kecil. Meskipun peningkatan performa terjadi pada CNN, namun algoritma LSTM sederhana yang dibangun masih sangat mendominasi dalam kasus *time series* pada permainan FPL.

Tabel 4.2 Perbandingan evaluasi model skenario kedua

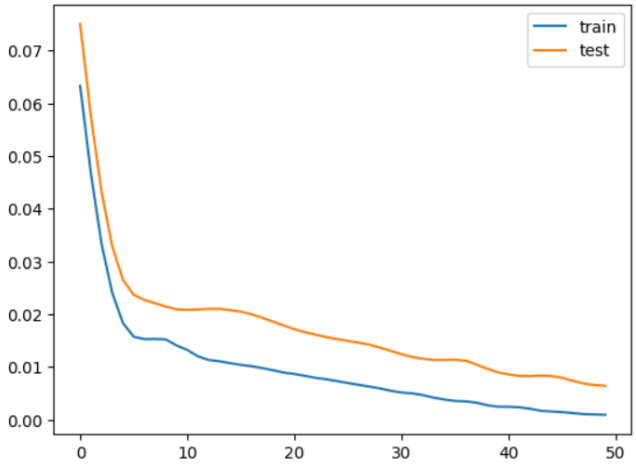
Position	**CNN**	**LSTM**
Goalkeeper	0.0040	0.0064
Defender	0.0020	0.0003
Midfielder	0.0020	0.0003

Forward	0.0030	0.0032
Average	0.0027	0,0025

4.5.2 Hasil Evaluasi Model

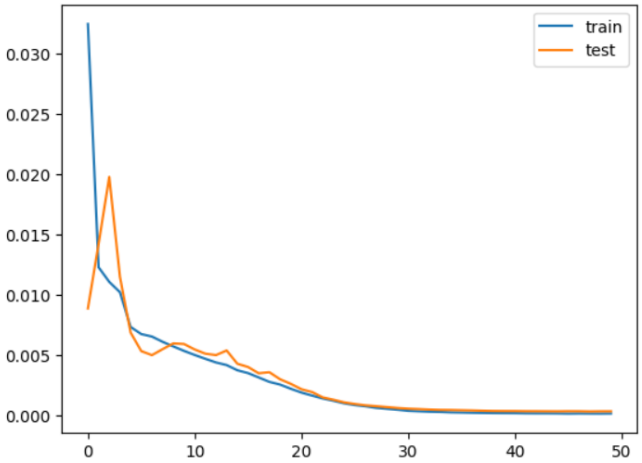
Pada subbab ini akan menampilkan hasil perjalanan evaluasi model yang dihasilkan oleh algoritma terbaik berdasarkan skenario pengujian yang dilakukan. Kemampuan yang dimiliki algoritma LSTM dalam mempelajari pola urutan data dalam jumlah besar memanfaatkan ketiga pintu menghasilkan nilai *error* lebih kecil dibandingkan algoritma CNN sehingga terpilih sebagai algoritma terbaik. Berdasarkan kedua skenario pengujian, nilai evaluasi error paling rendah tercipta pada skenario kedua dengan perbandingan 80% data latih dan 20% data uji pada model yang memiliki jumlah 50 *epoch*. setiap *epoch* akan menghasilkan besaran evaluasi pada masing-masing data yang dimulai dari epochs pertama yang direpresentasikan dalam bentuk grafik garis. Terdapat dua warna garis berbeda untuk membedakan antara *train-loss* dan *test-loss*. Garis berwarna biru menunjukkan nilai kerugian yang terjadi pada data latih selama proses pelatihan model, dikenal sebagai *train-loss*. Warna garis lainnya yaitu orange merepresentasikan *test-loss*, dimana nilai kerugian yang dihitung pada data uji yang bertujuan mengukur kemampuan generalisasi model terhadap data baru yang belum dikenali sebelumnya oleh model.

Dimulai dari pemain berposisi sebagai *goalkeeper* menujukkan nilai *test-loss* pada *epoch* ke 50 sebesar 0.0064. posisi yang memiliki nilai evaluasi paling tinggi dibandingkan dengan posisi lainnya, mengalami penurunan yang sangat signifikan pada *epoch* ke-0 hingga ke-12. Terjadi kenaikan nilai evaluasi model dalam 3 *epoch* selanjutnya mengakibatkan performa model sedikit terganggu. Kemampuan algoritma LSTM dalam menangani kasus *time series* mengembalikan kestabilan nilai evaluasi model yang mengalami penurunan kembali pada grafik *test-loss* hingga *epoch* ke-50 digambarkan pada Gambar 4.26. epochs terakhir menjadi nilai evaluasi model terbaik pada posisi *goalkeeper*.



Gambar 4.26 Test-loss pemain posisi *goalkeeper*

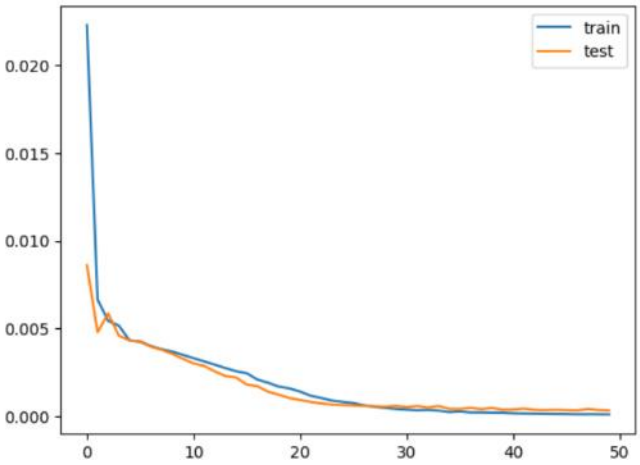
Model arsitektur LSTM yang memiliki 64 unit diimplementasikan pada data *defender* sebanyak 7.812 menghasilkan salah satu nilai evaluasi model terendah pada penelitian ini. besaran *test-loss* berhenti pada nilai 0.0003 pada jumlah epochs yang telah ditentukan dalam penelitian seperti ditunjukkan pada Gambar 4.27. jika dilihat pada garis berwarna biru, *train-loss* mengalami penurunan secara berurutan. Sedangkan garis orange lebih stabil mengalami penurunan ketika melewati epoch ke-20. Pada akhirnya kedua garis saling berdekatan menunjukkan bahwa model mampu mengeneralisasi secara baik, tidak hanya bagus pada data pelatihan saja namun memiliki performa baik pada data yang belum dikenali juga.



Gambar 4.27 Test-loss pemain posisi *defender*

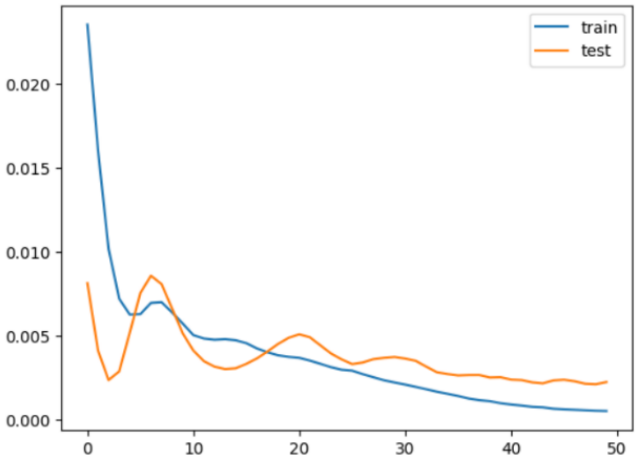
Selanjutnya, evaluasi model dilakukan menggunakan data pemain berposisi sebagai *midfielder*. Model mampu mengenaralisasi data dengan sangat baik, terbukti dengan *test-*

loss lebih rendah dibandingkan *train-loss* dari *epoch* ke-9 hingga *epoch* ke-28 seperti ditunjukkan pada Gambar 4.28. Namun, setelah itu, *test-loss* mulai meningkat hingga akhirnya berada sedikit di atas *train-loss.* Pada akhirnya, 11 fitur yang berhasil melewati proses seleksi fitur menghasilkan model dengan nilai *test-loss* sebesar 0.0003. Hasil yang didapatkan setara dengan pencapaian dalam evaluasi sebelumnya sebagai model terbaik berdasarkan posisi pemain. Penggunaan data lebih besar membantu memperkecil nilai evaluasi model sehingga model mampu melakukan prediksi secara akurat.



Gambar 4.28 Test-loss pemain posisi *midfielder*

Pada evaluasi model terakhir dimana pemain yang menempati posisi paling depan dalam skema permainan dalam suatu tim mengalami fluktuatif pada data uji. Model mulai menunjukkan kondisi stabil ketika mendekati *epoch* ke-40 seperti yang ditunjukkan pada Gambar 4.29. Salah satu faktor yang menyebabkan terjadinya kondisi ini ialah minimnya variasi data pemain karena terbatasnya nama unik yang didaftarkan oleh seluruh klub. Namun, model LSTM mampu mengatasi permasalahan keterbatasan data dan menghasilkan nilai test-loss menggunakan data *forward* yaitu 0.0032. Hasil evaluasi dapat ditingkatkan dengan penambahan data agar model LSTM dapat mempelajari lebih banyak pola dan urutan data sehingga menghasilkan nilai kerugian yang rendah. model *forward* berada di belakang hasil evaluasi terbaik, diikuti oleh model *goalkeeper*.

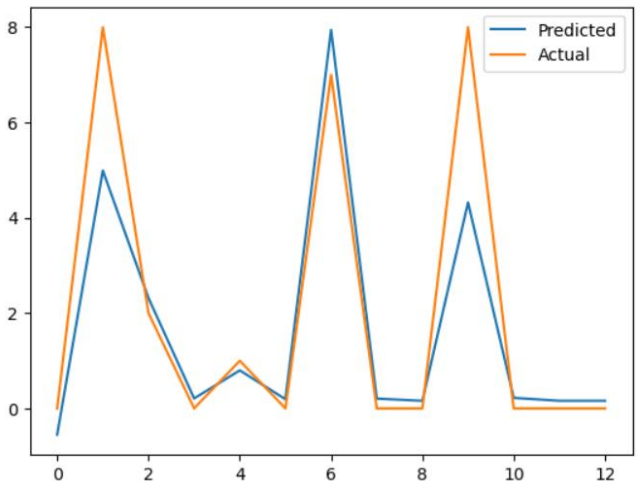


Gambar 4.29 Test-loss pemain posisi forward

4.5.3 Hasil Prediksi Model

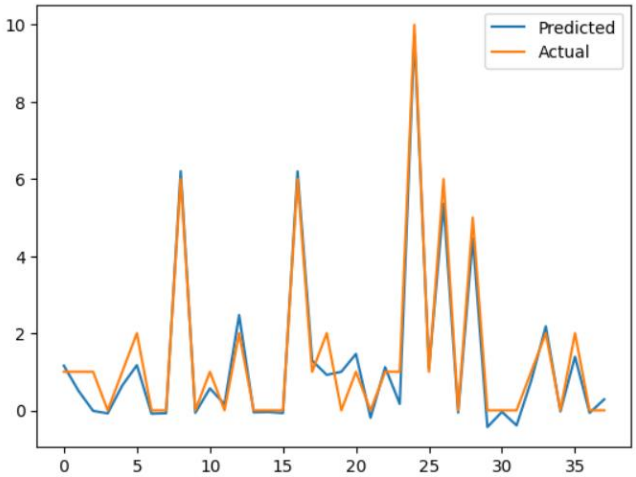
Pada bagian ini akan menampilkan kemampuan model prediksi yang telah dirancang pada data latih. Target prediksi model yang akan digunakan ialah fitur *total_points*. Dimana model yang telah dirancang akan melakukan prediksi *total_points* pemain yang berada pada data uji yang belum dikenali oleh model sebelumnya. Hasil prediksi model akan dilakukan pada model LSTM menggunakan pembagian data skenario kedua. Sumbu x akan merepresentasikan jumlah pemain dalam data uji sedangkan sumbu y menunjukkan *total_point*. Selain itu, terdapat dua warna garis berbeda yang masing-masing menunjukkan nilai prediksi dan nilai aktual. Garis berwarna biru menujukkan hasil prediksi nilai *total_points* yang dihasilkan oleh model. sedangkan, nilai aktual atau nilai sebenarnya yang telah dipisahkan sebelumnya pada data uji digambarkkan oleh garis orange. Performa model dapat dihitung dari selisih antara nilai prediksi model dan nilai aktual yang disediakan oleh data uji.

Untuk mengetahui hasil prediksi model LSTM pada posisi *goalkeeper* dilakukan penerapan pada data uji dengan posisi yang sama. Gambar 4.30 merupakan hasil prediksi model untuk setiap pemain berposisi sebagai *goalkeeper* yang terdaftar pada data uji. Pada grafik prediksi model diperoleh bahwa dua dari tiga nilai aktual yang menghasilkan nilai yang cukup tinggi yaitu 8, model belum mampu memprediksi mendekati angka aktual. Model prediksi hanya berada antara nilai 4 sampai 5 ketika nilai aktual cukup tinggi. Namun, dalam beberapa prediksi lainnya, nilai prediksi berada diatas nilai akutal.



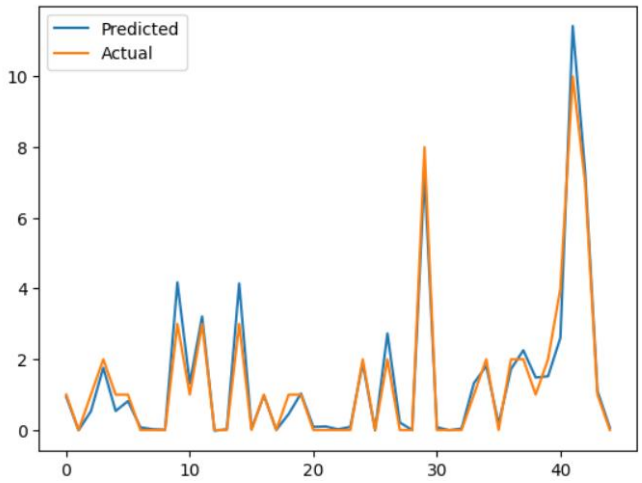
Gambar 4.30 Hasil prediksi pemain posisi goalkeeper

Hasil prediksi model *defender* yang diterapkan pada data uji yang belum dikenali oleh model sebelumnya. Garis biru menunjukkan kedekatan terhadap garis orange yang merepresentasikan nilai aktual, jika dibandingkan dengan model prediksi *goalkeeper*. pelatih FPL berpeluang menghasilkan nilai *total_points* tinggi pada posisi *defender* karena model tidak mengalami kehilangan *total_points* dalam jumlah besar. Ketika nilai aktual mulai mendekati 4, nilai prediksi memiliki jarak yang sangat tipis dengan nilai aktual. sedangkan nilai aktual sama dengan 2 atau dibawahnya, nilai prediksi sedikit tidak konsisten dalam upaya menjaga jarak dengan nilai aktual. Selisih jarak antara hasil prediksi dan aktual pemain defender ditunjukkan oleh Gambar 4.31.



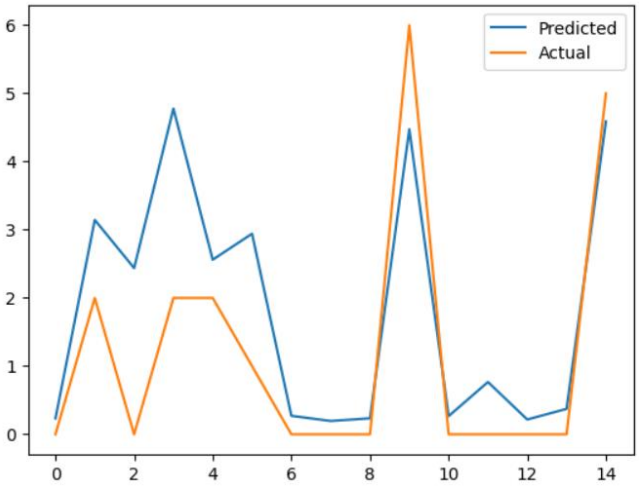
Gambar 4.31 Hasil prediksi pemain posisi defender

Pada Gambar 4.32 menunjukkan hasil prediksi *total_points* yang diterapkan pada pemain yang berposisi sebagai *midfielder* sebanyak 45 pemain. Beberapa nilai prediksi *total_points* berada dalam satu garis yang sama dengan nilai aktual. Hal ini dapat mengurangi selisih antara nilai prediksi dan nilai aktual yang dapat mempengaruhi evaluasi model. semakin banyak data yang diberikan memudahkan model untuk menemukan pola dan variasi data selama proses pelatihan sehingga meminimalisir kesalahan dalam melakukan prediksi.



Gambar 4.32 Hasil prediksi pemain posisi midfielder

Hasil model prediksi terhadap 15 pemain *forward* pada data uji digambarkan oleh Gambar 4.38. Nilai prediksi akan menghasilkan nilai yang lebih besar dibandingkan nilai aktual pada rentang nilai 0 sampai 2. Hal ini mengakibatkan selisih lebar jarak antara nilai prediksi dan nilai aktual yang dapat mempengaruhi hasil evaluasi model. sedangkan nilai aktual pada rentang 5 sampai 6, model prediksi mampu mendekatinya meskipun berada sedikit dibawah nilai aktual. Penambahan data pada posisi forward dapat menjadi solusi untuk memperkecil selisih jarak, sehingga lebih banyak variasi yang dapat dikenali oleh model dalam proses pelatihan.



Gambar 4.33 Hasil prediksi pemain posisi forward

BAB 5
Kesimpulan dan Saran

5.1 Kesimpulan

Dari penelitian yang telah dilakukan, maka diperoleh kesimpulan sebagai berikut:

1. Model *deep learning time series* terbaik untuk memprediksi total poin pemain pada pertandingan gameweek ke-6 menggunakan data historis 5 pertandingan sebelumnya pada game Fantasy Premier League ialah algoritma *Long Short-Term Memory*.
2. Berdasakan dua skenario pembagian data yang dilakukan, diperoleh bahwa skenario pembagian data terbaik adalah rasio pembagian data sebesar 80% data latih dan 20% data uji.
3. Model prediksi yang memiliki jumlah dataset yang lebih besar seperti pada posisi *defender* dan *midfielder* mampu menghasilkan nilai *mean squared error* lebih rendah dibandingkan pada posisi *goalkeeper* dan *forward*.
4. Diperoleh hasil evaluasi model LSTM menggunakan skenario kedua dari posisi *goalkeeper* hingga *forward* ialah 0.0060, 0.0003, 0.0003 dan 0.0022.

5.2 Saran

Adapun saran yang bertujuan untuk pengembangan pada penelitian ini, sebagai berikut :

1. Pengembangan model dapat dilakukan dengan *tuning hyperparameter* untuk meningkatkan performa model.
2. Penentuan jumlah *timesteps* yang digunakan pada model *time series*.
3. Perancangan model menggunakan algoritma *time series* lainnya dalam studi kasus game Fantasy Premier League.
4. Evaluasi model dapat dilakukan dengan menggunakan metode evaluasi lainnya untuk dijadikan perbandingan evaluasi pada metrik yang berbeda.

Daftar Pustaka

Alonso, R. P., & Babac, M. B. (2022). Machine learning approach to predicting a basketball game outcome. *International Journal of Data Science*, *7*(1), 60. https://doi.org/10.1504/ijds.2022.124356

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. In *Journal of Big Data* (Vol. 8, Issue 1). Springer International Publishing. https://doi.org/10.1186/s40537-021-00444-8

Ambarwari, A., Adrian, Q. J., & Herdiyeni, Y. (2020). Analisis Pengaruh Data Scaling Terhadap Performa Algoritme Machine Learning untuk Identifikasi Tanaman. *JURNAL RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *4*(1), 117–122. http://jurnal.iaii.or.id

Ardiansyah, M., Notodiputro, K. A., & Sartono, B. (2020). Peningkatan Presisi Dugaan Berat Gabah Melalui Proses Seleksi Peubah dalam Pembelajaran Mesin Statistika. *Prosiding Seminar Nasional VARIANSI*, 171–183.

Ardiansyah, Risnita, & Jailani, M. S. (2023). Teknik Pengumpulan Data Dan Instrumen Penelitian Ilmiah Pendidikan Pada Pendekatan Kualitatif dan Kuantitatif. *Jurnal IHSAN : Jurnal Pendidikan Islam*, *1*(2), 1–9. https://doi.org/10.61104/ihsan.v1i2.57

Association, T. F. S. & G. (2023). *Industry Demographics*. https://thefsga.org/industry-demographics/

Bangdiwala, M., Choudhari, R., Hegde, A., & Salunke, A. (2022). Using ML Models to Predict Points in Fantasy Premier League. *2022 2nd Asian Conference on Innovation in Technology, ASIANCON 2022*, 1–6. https://doi.org/10.1109/ASIANCON55314.2022.9909447

Beal, R., Norman, T. J., & Ramchurn, S. D. (2020). Optimising Daily Fantasy Sports Teams with Artificial Intelligence. *International Journal of Computer Science in Sport*, *19*(2), 21–35. https://doi.org/10.2478/ijcss-2020-0008

Bonello, N., Beel, J., Lawless, S., & Debattista, J. (2019). Multi-stream data analytics for enhanced performance prediction in fantasy football. *CEUR Workshop Proceedings*, *2563*, 284–292.

Carnegie, M. D. A. (2023). *Perbandingan Long Short Term Memory (LSTM) dan Gated*

Recurrent Unit (GRU) Untuk Memprediksi Curah Hujan. *7*, 1022–1032. https://doi.org/10.30865/mib.v7i3.6213

Dicoding. (2020). *Belajar Machine Learning untuk Pemula*. 29 Desember 2020. https://www.dicoding.com/academies/184

Faiqoh, H. (2023). *Penerapan Ensemble Feature Selection untuk Mengurangi Dimensionalitas dalam Prediksi Data Time Series*. 1–84.

Gde Agung Brahmana Suryanegara, Adiwijaya, & Mahendra Dwifebri Purbolaksono. (2021). Peningkatan Hasil Klasifikasi pada Algoritma Random Forest untuk Deteksi Pasien Penderita Diabetes Menggunakan Metode Normalisasi. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *5*(1), 114–122. https://doi.org/10.29207/resti.v5i1.2880

Granat, M. (2019). *How to Use Convolutional Neural Networks for Time Series Classification*. 5 Oct 2019. https://towardsdatascience.com/how-to-use-convolutional-neural-networks-for-time-series-classification-56b1b0a07a57

Guenther, J., & Sawodny, O. (2019). Feature selection for thermal comfort modeling based on constrained LASSO regression. *IFAC-PapersOnLine*, *52*(15), 400–405. https://doi.org/10.1016/j.ifacol.2019.11.708

Gupta, A. (2017). *Time Series Modeling for Dream Team in Fantasy Premier League*. *October*, 23–25. http://arxiv.org/abs/1909.12938

Handoko, W. T., & Handayani, A. N. (2023). Forecasting Solar Irradiation on Solar Tubes Using the LSTM Method and Exponential Smoothing. *Jurnal Ilmiah Teknik Elektro Komputer Dan Informatika (JITEKI)*, *9*(3), 649–660. https://doi.org/10.26555/jiteki.v9i3.26395

Kartiko, R. (2020). *FPL dalam Kacamata Drone Emprit Academic*. 9 January 2020. https://football-tribe.com/indonesia/2020/01/09/fpl-dalam-kacamata/

Khamsan, M. M., & Maskat, R. (2019). Handling Highly Imbalanced Output Class Label: a Case Study on Fantasy Premier League (Fpl) Virtual Player Price Changes Prediction Using Machine Learning. *Malaysian Journal of Computing*, *4*(2), 304–316. https://www.calculator.net/standard-deviation-calculator.html

Kim, Y., Hao, J., Mallavarapu, T., Park, J., & Kang, M. (2019). Hi-LASSO: High-Dimensional LASSO. *IEEE Access*, *7*, 44562–44573. https://doi.org/10.1109/ACCESS.2019.2909071

Kristiansen, B. K., Gupta, A., & Eilertsen, W. (2018). *Developing a Forecast-Based Optimization Model for Fantasy Premier League William Eilertsen Akash Gupta*

Bjørn Kåre Kristiansen.

Lim, B., & Zohren, S. (2021). Time-series forecasting with deep learning: A survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *379*(2194). https://doi.org/10.1098/rsta.2020.0209

Lindberg, A., & Söderberg, D. (2020). *Comparison of Machine Learning Approaches Applied to Predicting Football Players Performance*. Chalmers University of Technology.

Lorente, Ò., Riera, I., & Rana, A. (2021). *Image Classification with Classic and Deep Learning Techniques*. http://arxiv.org/abs/2105.04895

Marfu'ah, N. J. L., & Kurniawardhani, A. (2020). Comparison of CNN and SVM for Ship Detection in Satellite Imagery. *Automata*, *1*(1). https://journal.uii.ac.id/AUTOMATA/article/view/13973

Masters, R. (2022a). *Annual Report 2021/22*. https://www.premierleague.com/about/publications

Masters, R. (2022b). *Annual Report 2022/2023*. https://www.premierleague.com/about/publications

Murni, D., Studi Matematika, P., & Matematika dan Ilmu Pengetahuan Alam, F. (2023). *Prediksi Hasil Pertandingan Sepak Bola Liga Premier Inggris Dengan Artificial Neural Network Backpropagation*. *4*(3), 1523–1531. http://lebesgue.lppmbinabangsa.id/index.php/home

Nugroho, A. T. (2021). Gamifikasi, Pemasaran di Era Digital: Studi pada Pengguna Game Fantasy Premier League di Indonesia. *Jurnal Riset Komunikasi*, *4*(2), 261–274. https://doi.org/10.38194/jurkom.v4i2.376

Prabowo, D. . (2020). *Prediksi hasil pertandingan sepakbola english premier league dengan menggunakan algoritma k-nearest neighbors dan naive bayes classifer*. Universitas Islam Indonesia.

Prasetya, M. R. A., Priyatno, A. M., & Nurhaeni. (2023). Penanganan Imputasi Missing Values pada Data Time Series dengan Menggunakan Metode Data Mining. *Jurnal Informasi Dan Teknologi*, *5*(2), 52–62. https://doi.org/10.37034/jidt.v5i2.324

Purbolaksono, M. D., Irvan Tantowi, M., Imam Hidayat, A., & Adiwijaya, A. (2021). Perbandingan Support Vector Machine dan Modified Balanced Random Forest dalam Deteksi Pasien Penyakit Diabetes. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *5*(2), 393–399. https://doi.org/10.29207/resti.v5i2.3008

Ramdas, D. (2022). *Using Convolution Neural Networks to Predict the Performance of*

Footballers in the Fantasy Premier League (Issue 60241500). University of South Africa.

Rosihan, R., Fhadli, M., & Usman, A. A. H. (2023). Klasifikasi Kelayakan Pemberian Kredit Menggunakan Metode Decision Tree dengan Seleksi Fitur (Studi Kasus: PT. Adira Finance Cabang Kota Ternate). *Jurnal Pendidikan Tambusai*, *7*, 21517–21524. https://www.jptam.org/index.php/jptam/article/view/9915

Sabar Sautomo, & Hilman Ferdinandus Pardede. (2021). Prediksi Belanja Pemerintah Indonesia Menggunakan Long Short-Term Memory (LSTM). *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *5*(1), 99–106. https://doi.org/10.29207/resti.v5i1.2815

Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*, *Volume 404*(132306). https://doi.org/https://doi.org/10.1016/j.physd.2019.132306

Sudrajat, W., & Cholid, I. (2023). K-Nearest Neighbor (K-Nn) Untuk Penanganan Missing Value Pada Data Umkm. *Jurnal Rekayasa Sistem Informasi Dan Teknologi*, *1*(2), 54–63. https://doi.org/10.59407/jrsit.v1i2.77

Tarigan, I. D. S., Roni Habibi, & Rd. Nuraini Siti Fatonah. (2023). Evaluasi Algoritma Klasifikasi Machine Learning Kategori Nilai Akhir Tunjangan Kinerja Pegawai. *Jurnal Sistem Cerdas*, *6*(3), 251–261. https://doi.org/10.37396/jsc.v6i3.246

Van Eetvelde, H., Mendonça, L. D., Ley, C., Seil, R., & Tischer, T. (2021). Machine learning methods in sport injury prediction and prevention: a systematic review. *Journal of Experimental Orthopaedics*, *8*(1). https://doi.org/10.1186/s40634-021-00346-x

Yang, R. (2019). *Using Supervised Learning to Predict English Premier League Match Results From Starting Line-up Player Data*. 1–76.

Zhang, Y., & Thorburn, P. J. (2022). Handling missing data in near real-time environmental monitoring: A system and a review of selected methods. *Future Generation Computer Systems*, *128*, 63–72. https://doi.org/10.1016/j.future.2021.09.033

Zulaikhah Hariyanti Rukmana, S., Aziz, A., & Harianto, W. (2022). Optimasi Algoritma K-Nearest Neighbor (Knn) Dengan Normalisasi Dan Seleksi Fitur Untuk Klasifikasi Penyakit Liver. *JATI (Jurnal Mahasiswa Teknik Informatika)*, *6*(2), 439–445. https://doi.org/10.36040/jati.v6i2.4722

LAMPIRAN A