BAB 3
METODOLOGI PENELITIAN

3.1 Lingkungan Pengembangan dan Pustaka

Implementasi penelitian dilakukan menggunakan *notebook* pada lingkungan Python. Proses utama meliputi pra-pemrosesan data, rekayasa fitur, pelatihan model, evaluasi, serta interpretabilitas. Pustaka yang digunakan dirangkum sebagai berikut (versi disesuaikan dengan *environment* penelitian):

- Python (`v3.13.5`)
- *Jupyter Notebook* (`v7.3.2`)
- *NumPy* (`v2.1.3`) dan *pandas* (`v2.2.3`) untuk pengolahan data
- *scikit-learn* (`v1.6.1`) untuk pemodelan dan evaluasi
- *imbalanced-learn* (`v0.13.0`) untuk *random undersampling*
- *TensorFlow* (`v2.20.0`) *Keras* (`v3.12.0`) untuk pemodelan BiLSTM
- *KerasTuner* (`v1.4.8`) untuk *hyperparameter tuning*
- *SHAP* (`v0.49.1`) untuk interpretabilitas model

3.2 Pra-pemrosesan Data

Tahap pra-pemrosesan dilakukan untuk memastikan data domain berada pada format yang konsisten sebelum masuk ke tahapan ekstraksi fitur dan pelatihan model. Tahapan utama meliputi unifikasi sumber data (domain DGA dan domain *legit*), ekstraksi *Second Level Domain* (SLD), pembersihan karakter, serta penanganan duplikasi dan konflik label.

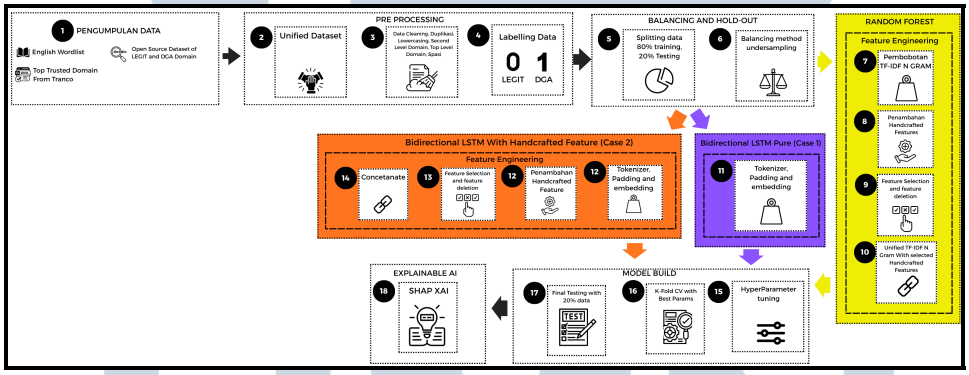
3.2.1 Sumber Dataset dan Unifikasi Data

Dataset penelitian ini dibentuk melalui unifikasi beberapa sumber. Kelas DGA (*malicious*) dihimpun dari UMUDGA [31], Data Driven Security [32], ExtraHop [33], serta Kaggle [34]. Kelas *legit* (*benign*) dihimpun dari daftar domain

populer (misalnya Alexa Top-1M) dan digunakan juga pembanding reputasi yang lebih modern melalui Tranco [35].

Seluruh domain dari berbagai sumber kemudian diseragamkan formatnya, diekstraksi pada level *second-level domain*/SLD, dilakukan deduplikasi, serta diperiksa konflik label sebelum membentuk *unified dataset* yang dipakai pada eksperimen.

Alur penelitian ditunjukkan pada Gambar 3.1. Tahapan dimulai dari pengumpulan dan unifikasi dataset domain DGA dan domain *legit*, dilanjutkan pra-pemrosesan (ekstraksi SLD, pembersihan karakter, serta penanganan duplikasi), ekstraksi fitur (fitur *handcrafted* dan TF–IDF *character n-gram*), penyeimbangan data, pembagian data latih/uji, pelatihan model (Random Forest dan BiLSTM), evaluasi menggunakan metrik klasifikasi, serta analisis interpretabilitas menggunakan SHAP.



Gambar 3.1. Alur penelitian deteksi DGA berbasis nama domain, mulai dari penyusunan dataset (kelas DGA dan *legit*) hingga evaluasi dan interpretabilitas (XAI).

Tahap yang dijelaskan pada bagian ini merujuk pada dua *notebook* berikut:

- `Import-cleaning-andUnifyData.ipynb`: menyusun dataset, membersihkan data, mengekstraksi SLD, menghapus duplikasi, menangani konflik label, dan menghasilkan dataset final terlabel.
- `RF_Gradual_Features_V2.ipynb`: melakukan *hold-out* 80:20 dan penyeimbangan kelas pada data latih menggunakan *random undersampling*.

3.2.2 Konfigurasi Input, Output, dan Seed Eksperimen

Pada *notebook* `Import-cleaning-andUnifyData.ipynb`, data mentah disiapkan pada direktori `Traning Dataset`. Dua berkas utama yang dipakai

sebagai masukan adalah:

- `DGA_Domain.csv` untuk kelas domain DGA.
- `LEGIT_Domain.csv` untuk kelas domain *legit*.

Seluruh hasil pra-pemrosesan disimpan pada folder `Traning Dataset/_Preprocessed`. Untuk menjaga konsistensi proses acak pada tahap tertentu, *seed* ditetapkan dengan nilai `SEED = 42`.

3.2.3 Normalisasi String dan Standarisasi Kolom Domain

Karena sumber dataset dapat memiliki format kolom yang berbeda, langkah awal adalah menentukan kolom domain yang benar menggunakan fungsi `choose_domain_col`. Setelah kolom domain ditentukan, dilakukan normalisasi teks menggunakan fungsi `normalize_text_series` dengan prinsip:

- menghapus spasi di awal dan akhir dengan `strip`,
- mengubah seluruh karakter menjadi huruf kecil dengan `lower`.

Langkah ini memastikan entri domain lebih stabil untuk proses deduplikasi dan ekstraksi SLD.

3.2.4 Ekstraksi Second-Level Domain

Penelitian ini memfokuskan analisis pada *second-level domain* (SLD). Oleh karena itu, setiap entri domain diekstraksi menjadi SLD menggunakan fungsi `sld`. Implementasi ekstraksi dilakukan dengan pendekatan pemisahan komponen domain, serta menyediakan opsi penggunaan *library* `tldextract` untuk kasus *public suffix* bertingkat bila diperlukan.

Hasil ekstraksi disimpan dalam kolom `sld` melalui fungsi `add_sld_column`. Setelah itu, data yang tidak menghasilkan SLD valid (misalnya kosong atau *null*) dibuang agar tidak menambah *noise* pada tahap berikutnya.

3.2.5 Deduplikasi dan Pembersihan Tahap 1

Setelah normalisasi dan ekstraksi SLD, dilakukan deduplikasi pada level SLD sehingga setiap nilai `sld` hanya muncul satu kali dalam masing-masing kelas. Hasil pembersihan tahap 1 disimpan sebagai:

- `DGA_stage1_clean.csv`
- `LEGIT_stage1_clean.csv`

3.2.6 Pemeriksaan Konflik Antarkelas dan Pembuatan Data Tanpa Konflik

Karena dataset berasal dari banyak sumber, terdapat kemungkinan SLD yang sama muncul pada kedua kelas. Untuk menjaga konsistensi label, dilakukan pemeriksaan konflik SLD antarkelas. SLD yang terdeteksi konflik disimpan pada:

- `conflicts_same_sld.csv`

Setelah konflik diidentifikasi, SLD konflik dikeluarkan dari kedua kelas, lalu dibuat versi data tanpa konflik sebagai:

- `DGA_stage1_clean_noConflicts.csv`
- `LEGIT_stage1_clean_noConflicts.csv`

3.2.7 Pembersihan Lanjutan SLD dan Pelabelan Biner

Pada tahap berikutnya dilakukan pembersihan lanjutan untuk memastikan kolom SLD benar-benar berisi string domain tanpa artefak format, misalnya kasus string yang mengandung pola `domain,label`. Hasil pembersihan lanjutan disimpan sebagai:

- `DGA_stage2_pure_sld.csv`
- `LEGIT_stage2_pure_sld.csv`

Selanjutnya, dataset diberi label biner sesuai skema penelitian:

- label `1` untuk kelas DGA,
- label `0` untuk kelas *legit*.

Hasil pelabelan disimpan sebagai:

- `DGA_stage3_labeled.csv`
- `LEGIT_stage3_labeled.csv`

Terakhir, kedua kelas digabung menjadi satu dataset final terunifikasi:

- `All_Domain_DGA_LEGIT_Unified.csv`

3.2.8 Hold-out dan Penyeimbangan Kelas

Tahap ini mengatur pembagian data latih dan data uji, lalu menerapkan penyeimbangan kelas hanya pada data latih. Implementasi dilakukan pada *notebook* `RF_Gradual_Features_V2.ipynb`.

3.2.9 Memuat Dataset Kerja untuk Splitting

Dataset kerja dimuat dari berkas:

- `Dataset_Balanced_V2.csv`

Pada saat pemuatan, dilakukan validasi dasar:

- membuang baris yang memiliki *missing value* pada kolom `domain` dan `label`,
- memastikan `domain` bertipe string dan `label` bertipe integer.

Dataset yang dimuat berjumlah 28.330.712 baris. Distribusi kelas sebelum proses *splitting* adalah:

- label `1`: 26.991.674
- label `0`: 1.339.038

3.2.10 Hold-out Split 80:20

Pembagian data dilakukan dengan skema *hold-out* menggunakan `train_test_split` dengan:

- `test_size = 0.2`
- `random_state = 42`
- `stratify = y` untuk menjaga proporsi kelas tetap serupa dengan distribusi awal.

Hasil splitting kemudian disimpan sebagai:

- Data latih (80%): `Dataset_Balanced_Train80_V2.csv` sebanyak 22.664.569 sampel.

- Data uji (20%): `Dataset_Balanced_Test20_V2.csv` sebanyak 5.666.143 sampel.

Distribusi kelas pada data latih sebelum penyeimbangan adalah:

- label `1`: 21.593.339
- label `0`: 1.071.230

Distribusi kelas pada data uji (tidak disentuh) adalah:

- label `1`: 5.398.335
- label `0`: 267.808

3.2.11 Penyeimbangan Data Latih dengan Random Undersampling

Setelah skema *hold-out* selesai, penyeimbangan kelas diterapkan hanya pada data latih menggunakan *random undersampling*. Implementasi dilakukan dengan *library* `imbalanced-learn` melalui `RandomUnderSampler` dengan parameter:

- `random_state = 42`

Mekanisme ini mereduksi jumlah sampel kelas mayoritas pada data latih hingga setara dengan kelas minoritas. Setelah *undersampling*, distribusi kelas data latih menjadi seimbang:

- label `0`: 1.071.230
- label `1`: 1.071.230

Total sampel baru pada data latih seimbang adalah 2.142.460 dan disimpan sebagai `Undersampled_Train80_V2.csv`. Ringkasan dari jumlah sampel dan distribusi kelas pada setiap tahap pembagian data dapat tersaji pada tabel 3.1 berikut ini

Tabel 3.1. Ringkasan jumlah sampel dan distribusi kelas pada setiap tahap pembagian data

Tahap	**Jumlah sampel**
Dataset setelah pembentukan data seimbang (V2) sebelum pembagian (*train-test split*)	28,330,712
Data latih (80%) sebelum penyeimbangan (`Balanced Train80 V2.csv`)	22,664,569
label 1 (DGA) pada data latih sebelum penyeimbangan	21,593,339
label 0 (*legit*) pada data latih sebelum penyeimbangan	1,071,230
Data uji (20%) tanpa penyeimbangan (`Balanced Test20 V2.csv`)	5,666,143
label 1 (DGA) pada data uji	5,398,335
label 0 (*legit*) pada data uji	267,808
Data latih setelah *random undersampling* (rasio 1:1)	2,142,460
label 1 (DGA) pada data latih setelah undersampling	1,071,230
label 0 (*legit*) pada data latih setelah undersampling	1,071,230

Pada penelitian ini, data uji `Dataset_Balanced_Test20_V2.csv` dibiarkan dalam distribusi aslinya agar evaluasi performa model mencerminkan kondisi masalah yang lebih realistis.

Meskipun berkas dinamai `Balanced Train80 V2.csv`, distribusi kelas pada data latih sebelum undersampling masih tidak seimbang; penyeimbangan dilakukan hanya pada data latih melalui *random undersampling*.

3.2.12 Ringkasan Artefak Data yang Dihasilkan

Berikut ringkasan berkas keluaran utama yang menjadi masukan untuk eksperimen inti pada tahap selanjutnya:

- `All_Domain_DGA_LEGIT_Unified.csv`: dataset final hasil unifikasi dan pelabelan.
- `Dataset_Balanced_Train80_V2.csv`: data latih hasil *hold-out* sebelum penyeimbangan.
- `Dataset_Balanced_Test20_V2.csv`: data uji hasil *hold-out* tanpa penyeimbangan.

- `Undersampled_Train80_V2.csv`: data latih yang sudah seimbang dengan *random undersampling*.

3.3 Metodologi Eksperimen Random Forest

Tahapan eksperimen *Random Forest* pada penelitian ini dijalankan secara operasional melalui rangkaian proses yang menghasilkan berkas antara, model, dan keluaran evaluasi. Seluruh proses disusun agar konsisten dari sisi data masukan, rekayasa fitur, pelatihan, hingga pengujian pada data *hold-out*.

3.3.1 Penyiapan Direktori Kerja dan Data Masukan

Penelitian ini menggunakan struktur folder proyek yang memisahkan *input* dan *output*. Data utama hasil pembersihan dan penyatuan data dibaca dari berkas `Dataset_Unified_Clean_NoConflicts.csv`. Seluruh artefak eksperimen *Random Forest* kemudian disimpan ke folder keluaran yang dibagi per tahapan, misalnya folder `Splitting Data`, `TF-IDF Features`, `Handcrafted Features`, dan `Combined Features`.

3.3.2 Pembagian Data *Hold-out* dan Penyeimbangan Kelas pada Data Latih

Dataset `Dataset_Unified_Clean_NoConflicts.csv` dibagi menggunakan skema *train-test split* terstratifikasi dengan `test_size = 0.2` dan `random_state = 42`. Hasil pembagian disimpan menjadi:

- `Dataset_Balanced_Train80_V2.csv` untuk data latih 80%.
- `Dataset_Balanced_Test20_V2.csv` untuk data uji 20%.

Penyeimbangan kelas kemudian dilakukan hanya pada data latih menggunakan *random undersampling* dengan `RandomUnderSampler(random_state = 42)`. Rincian jumlah sampel dan distribusi kelas dilaporkan pada Bab 4.

3.3.3 Ekstraksi Fitur TF–IDF *n-gram* Karakter

Fitur *TF–IDF* dibentuk dari kolom domain pada data latih yang telah diseimbangkan (hasil *undersampling*, yaitu `Undersampled_Train80_V2.csv`). Konfigurasi yang digunakan adalah:

- `analyzer = 'char'`
- `ngram_range = (2, 3)`
- `max_features = 20000`
- `lowercase = False`

Model *vectorizer* disimpan agar dapat digunakan ulang pada data uji, yaitu `tfidf_vectorizer_optim20k_v2.pkl`. Hasil transformasi data latih disimpan dalam bentuk *joblib* untuk efisiensi:

- `X_train_tfidf_v2_optim20k.joblib`
- `y_train_v2_optim20k.joblib`

Dimensi matriks *TF–IDF* pada data latih adalah (n_train $\times$ 20.000).

3.3.4 Ekstraksi Fitur *Handcrafted* dan Standardisasi

Selain *TF–IDF*, penelitian ini membangun 20 fitur *handcrafted* yang dilabeli `F01` sampai `F20`. Fitur dihitung dari string domain dan sebagian memanfaatkan basis pengetahuan eksternal:

- `EnglishWordlist.txt`
- `TrancoTop-1m_SLD_Only.txt`

Untuk mempercepat pencarian kata kamus, digunakan *Aho-Corasick* pada fitur berbasis kamus. Untuk statistik *n-gram* pada domain populer, digunakan konfigurasi `NGRAM_SIZES = [2, 3]` dan `TOP_N = 100000`. Seluruh fitur *handcrafted* kemudian distandardisasi menggunakan *StandardScaler* agar skala fitur lebih seragam. Objek *scaler* disimpan sebagai `scaler_handcrafted_v2.joblib`. Dimensi fitur *handcrafted* sebelum reduksi adalah (n_train $\times$ 20).

3.3.5 Seleksi Fitur *Handcrafted* berdasarkan *Feature Importance*

Untuk mengurangi fitur yang tidak informatif dan menjaga efisiensi komputasi, dilakukan seleksi fitur *handcrafted* menggunakan *feature importance* dari model pemeriksaan cepat. Proses ini menggunakan:

- `SAMPLE_SIZE = 100000`
- `RANDOM_SEED = 42`
- `RandomForestClassifier(n_estimators = 50, random_state = 42, n_jobs = -1)`

Berdasarkan hasil *feature importance*, fitur yang dihapus adalah: `F02, F03, F19, F11, F12, F16, F17, F08`. Fitur yang dipertahankan adalah: `F01, F04, F05, F06, F07, F09, F10, F13, F14, F15, F18, F20`. Setelah seleksi, dimensi fitur *handcrafted* menjadi (n_train $\times$ 12).

Daftar lengkap fitur leksikal/*handcrafted* yang digunakan pada penelitian ini disajikan pada Tabel 3.2.

Tabel 3.2. Daftar fitur ekstraksi untuk deteksi DGA

Kode	**Deskripsi Fitur**
F01	Panjang *string* domain.
F02	Jumlah subdomain (dihitung berdasarkan jumlah karakter titik).
F03	*Placeholder* TLD (nilai konstan 0).
F04	Rasio karakter digit terhadap total panjang domain.
F05	Rasio huruf vokal terhadap total panjang domain.
F06	Rasio huruf konsonan terhadap total panjang domain.
F07	Panjang maksimum deret konsonan yang muncul berturut-turut.
F08	Panjang maksimum deret digit yang muncul berturut-turut.
F09	Entropi Shannon karakter (mengukur tingkat keacakan distribusi karakter).
F10	Entropi *Bigram* karakter.
F11	Entropi *Trigram* karakter.
F12	Entropi *4-gram* karakter.
F13	Rata-rata log-frekuensi *bigram* berdasarkan korpus Tranco.
F14	Standar deviasi log-frekuensi *bigram*.
F15	Proporsi *bigram* tak dikenal (yang tidak ditemukan di korpus Tranco).
F16	Rasio karakter domain yang tertutup oleh kata-kata kamus Inggris (menggunakan algoritma *Aho-Corasick*).
F17	Panjang *substring* kata kamus Inggris terpanjang yang ditemukan dalam domain.
F18	Rasio transisi pola Vokal-Konsonan (seberapa sering V $\leftrightarrow$ K berpindah).
F19	Jumlah karakter titik (.) dalam domain.
F20	Jumlah karakter tanda hubung (-) dalam domain.

3.3.6 Penggabungan fitur TF–IDF dan *handcrafted*

Hasil TF–IDF dan fitur *handcrafted* kemudian digabungkan melalui konkatenasi untuk membentuk fitur gabungan. Tahap ini menghasilkan matriks fitur gabungan yang menjadi masukan bagi model Random Forest. Artefak keluaran disimpan pada:

- `X_train_combined_v2_optim20k.joblib`
- `y_train_combined_v2_optim20k.joblib`

3.3.7 *Hyperparameter tuning* dengan *RandomizedSearchCV*

Optimasi *hyperparameter* dilakukan menggunakan *RandomizedSearchCV*. Rentang parameter utama yang diuji meliputi *n_estimators*, *max_depth*, *min_samples_split*, dan *min_samples_leaf*. Proses pencarian parameter terbaik dilakukan pada data latih, kemudian parameter terbaik digunakan untuk pelatihan model final.

3.3.8 Pelatihan Model Akhir dan Penyimpanan Model

Setelah parameter terbaik diperoleh, model *Random Forest* final dilatih menggunakan seluruh data latih `X_train_combined_v2_optim20k.joblib`. Model akhir disimpan sebagai `rf_final_optim20k_v2.joblib` agar dapat digunakan ulang pada tahap evaluasi.

3.3.9 Evaluasi Model

Evaluasi model dilakukan pada data uji menggunakan beberapa metrik klasifikasi, yaitu *accuracy*, *precision*, *recall*, *F1-score*, dan ROC-AUC, serta dianalisis menggunakan *confusion matrix*. Seluruh nilai hasil evaluasi untuk tiap skenario eksperimen disajikan dan dibahas pada Bab 4.

Implementasi evaluasi menghasilkan *classification report* dan *confusion matrix* sebagai ringkasan performa pada data uji.

3.4 Eksperimen *BiLSTM* Murni (*Pure*)

Pada skenario ini, model *Bidirectional LSTM* (*BiLSTM*) digunakan secara murni berbasis urutan karakter *second-level domain* (SLD), tanpa penambahan

fitur *handcrafted*. Seluruh proses eksperimen terdokumentasi dalam *pipeline* yang memuat tahap pemuatan data hasil *hold-out* dan *balancing*, tokenisasi karakter, pelatihan dengan validasi silang, pelatihan model final, hingga evaluasi pada data uji.

3.4.1 Input Data untuk *BiLSTM Pure*

Eksperimen *BiLSTM Pure* menggunakan keluaran tahap pemisahan data (*hold-out*) dan penyeimbangan kelas yang telah dilakukan sebelumnya. Berkas yang digunakan adalah:

- `Dataset_Balanced_Train80_V2.csv` sebagai data latih (berada pada folder `Splitting Data`).
- `Dataset_Balanced_Test20_V2.csv` sebagai data uji (berada pada folder `Splitting Data`).

Sesuai catatan pada *pipeline*, data latih sudah dalam kondisi seimbang (50:50) akibat proses *random undersampling* pada subset 80%, sedangkan data uji dibiarkan merepresentasikan distribusi aslinya. Setelah dibaca, jumlah sampel yang terlapor adalah 2,142,460 baris untuk data latih dan 5,666,143 baris untuk data uji. Kolom yang digunakan sebagai masukan adalah `domain`, sedangkan target klasifikasi adalah `label` (0 untuk *legit* dan 1 untuk DGA). Untuk menjaga konsistensi, entri kosong pada `domain`/`label` dibuang, `domain` dipastikan bertipe string, dan `label` dipastikan bertipe integer.

3.4.2 Tokenisasi Karakter dan *Padding*

Karena *BiLSTM* memerlukan masukan berupa sekuens numerik, nama domain diubah menjadi urutan indeks karakter menggunakan *tokenizer* tingkat karakter (*character-level*). Pada tahap ini digunakan *oov token* `<UNK>` untuk menangani karakter yang tidak dikenali. Setelah *tokenizer* dibentuk dan dilatih pada data latih, setiap domain dikonversi menjadi sekuens indeks.

Panjang sekuens kemudian diseragamkan menggunakan *padding* dan *truncating* di sisi belakang (*post*). Nilai panjang maksimum (`MAX_LEN`) ditentukan berdasarkan panjang maksimum aktual pada data latih, kemudian ditambah 5 sebagai *buffer*. Hasil perhitungan yang tercatat adalah:

- Panjang domain terpanjang pada data latih: 63 karakter.

- `MAX_LEN` yang digunakan: 68.

Artefak penting dari tahap ini disimpan agar konsisten digunakan pada eksperimen lanjutan:

- *Tokenizer*: `tokenizer_pure_v2.pickle`
- Panjang maksimum: `max_len_pure.joblib`

3.4.3 Penetapan *Hyperparameter* dan Reproduksibilitas

Untuk menjaga reproduksibilitas, *pipeline* menetapkan `RANDOM_SEED = 42` yang digunakan pada proses pembagian *fold*. Parameter model yang digunakan pada skenario ini dirangkum pada variabel `best_params`, yaitu:

- `embedding_dim` = 32
- `lstm_units` = 128
- `dropout` = 0.2
- `learning_rate` = 0.01
- `n_splits` = 5

3.4.4 Validasi Silang *Stratified K-Fold* pada Data Latih

Sebelum melatih model final, dilakukan validasi silang menggunakan *Stratified K-Fold* dengan 5 *fold* (`n_splits=5`), *shuffle* aktif, dan *random state* mengikuti `RANDOM_SEED`. Pada setiap *fold*, model dilatih menggunakan konfigurasi berikut:

- Arsitektur: *Embedding* $\rightarrow$ *Bidirectional LSTM* $\rightarrow$ *Dropout* $\rightarrow$ *Dense* (sigmoid).
- *Optimizer*: Adam dengan `learning_rate = 0.01`.
- *Loss*: *binary cross-entropy*.
- Pelatihan per *fold*: 10 epoch dengan batch size 2048.
- *Early stopping*: memantau *val_loss*, `patience=3`, dan *restore best weights*.

Pada akhir tiap *fold*, *pipeline* menghitung metrik evaluasi pada subset validasi dan merangkum performa lintas *fold*. Ringkasan nilai metrik dilaporkan pada Bab 4.

3.4.5 Pelatihan Model Final dan Penyimpanan Model

Setelah validasi silang, model final dilatih menggunakan seluruh data latih (hasil *undersampling*) dengan konfigurasi:

- Epoch maksimum: 20
- Batch size: 2048
- Validation split: 0.1 dari data latih
- *Early stopping*: memantau *val_loss*, `patience=5`, *restore best weights*
- *Model checkpoint*: menyimpan bobot terbaik berbasis *val_loss*

Model terbaik disimpan pada berkas `bilstm_pure_final.keras` di folder keluaran skenario, sehingga dapat digunakan kembali tanpa melatih ulang.

3.4.6 Evaluasi pada Data Uji

Evaluasi akhir dilakukan pada `Dataset_Balanced_Test20_V2.csv` yang diproses menggunakan *tokenizer* dan `MAX_LEN` yang sama. *Pipeline* menghasilkan probabilitas prediksi melalui `predict` dan mengonversinya menjadi label biner menggunakan ambang 0,5. Kinerja model dihitung menggunakan metrik klasifikasi biner (akurasi, presisi, *recall*, *F1-score*, *ROC-AUC*), serta disajikan *confusion matrix* dan *classification report*. Hasil numerik disajikan pada Bab 4.

3.5 Eksperimen BiLSTM Hibrida dengan Tambahan Fitur *Handcrafted*

3.5.1 Tujuan

Skenario BiLSTM hibrida dirancang untuk menggabungkan dua sumber informasi, yaitu pola urutan karakter dari SLD yang dipelajari BiLSTM dan sinyal statistik yang dirangkum melalui fitur *handcrafted*. Dengan strategi ini, model tidak hanya mengandalkan pembelajaran representasi sekuens, tetapi juga memanfaatkan fitur-fitur ringkas yang secara eksplisit menggambarkan karakteristik string domain.

3.5.2 Artefak dan Struktur Penyimpanan

Seluruh artefak eksperimen BiLSTM disimpan pada folder `BiLSTM_Data`. Berkas-berkas utama yang digunakan dan dihasilkan pada skenario hibrida adalah:

- Data latih: `Dataset_Balanced_Train80_V2.csv`
- Data uji: `Dataset_Balanced_Test20_V2.csv`
- *Tokenizer* karakter: `tokenizer_char_v2.pickle`
- Panjang sekuens: `max_len_v2.joblib`
- Sekuens latih (numerik): `X_train_seq_v2.npy`
- Label latih: `y_train_seq_v2.npy`
- Fitur *handcrafted* latih: `X_train_hc_v2.npy`
- *Scaler* fitur *handcrafted*: `scaler_hc_v2.joblib`
- *Best hyperparameters*: `best_params_bilstm.joblib`
- Model final: `bilstm_final_v2.keras`

3.5.3 Persiapan Data Latih, Tokenisasi Karakter, dan *Padding*

Tahap persiapan dimulai dengan memuat data latih dari `Dataset_Balanced_Train80_V2.csv`. Kolom `domain` digunakan sebagai masukan model dan kolom `label` sebagai target biner. Selanjutnya dilakukan *tokenisasi* pada level karakter menggunakan konfigurasi berikut:

- `VOCAB_SIZE = 1000`
- *Tokenizer* karakter: `char_level=True`, `oov_token='<UNK>'`, `lower=True`

Panjang sekuens ditetapkan dari panjang maksimum SLD pada data latih dan pada eksperimen ini bernilai `MAX_LEN = 63`. Setiap domain kemudian diubah menjadi sekuens indeks dan dipadatkan menggunakan *padding* di bagian akhir (`padding='post'`) serta pemotongan di bagian akhir (`truncating='post'`). Hasil tahap ini disimpan sebagai `X_train_seq_v2.npy`, label sebagai `y_train_seq_v2.npy`, panjang sekuens sebagai `max_len_v2.joblib`, dan *tokenizer* sebagai `tokenizer_char_v2.pickle`.

3.5.4 Ekstraksi dan Seleksi Fitur *Handcrafted* untuk Cabang Hibrida

Pada skenario hibrida, fitur *handcrafted* awalnya dibentuk sebagai 20 fitur dengan penamaan `F01` sampai `F20`. Setelah fitur terbentuk, dilakukan penghapusan fitur tertentu untuk membentuk subset akhir, yaitu:

- Fitur yang dihapus: `F02`, `F03`, `F08`, `F11`, `F12`, `F16`, `F17`, `F19`
- Fitur yang digunakan (12 fitur): `F01`, `F04`, `F05`, `F06`, `F07`, `F09`, `F10`, `F13`, `F14`, `F15`, `F18`, `F20`

Fitur *handcrafted* kemudian dinormalisasi menggunakan *StandardScaler* yang dipelajari dari data latih. Artefaknya disimpan sebagai `scaler_hc_v2.joblib`, sedangkan matriks fitur latih akhirnya disimpan sebagai `X_train_hc_v2.npy`. Pada tahap pelatihan final, dimensi data latih dilaporkan dalam bentuk (n_train $\times$ 63) untuk sekuens dan (n_train $\times$ 12) untuk fitur *handcrafted*. Rincian jumlah sampel dijelaskan pada Bab 4.

3.5.5 *Hyperparameter tuning* model BiLSTM hibrida

Penalaan parameter dilakukan menggunakan *Keras Tuner* dengan strategi *RandomSearch*. Proses tuning dijalankan sebanyak 10 *trials* dan dievaluasi menggunakan `val_accuracy`. Untuk mempercepat penelusuran, tuning menggunakan `epochs=3` dan `batch_size=2048`, serta hasil tuning disimpan pada folder `bilstm_tuning_v2`. Ruang pencarian parameter yang digunakan meliputi:

- `embedding_dim` dari {32, 64}
- `lstm_units` dari {32, 64, 128}
- `dense_hc_units` dari {32, 64, 128}
- `dropout_rate` dari {0.2, 0.3, 0.4}
- `learning_rate` dari {1e-2, 1e-3}

Parameter terbaik kemudian disimpan pada `best_params_bilstm.joblib` untuk digunakan pada pelatihan akhir. Nilai numerik hasil tuning dan pembahasannya disajikan pada Bab 4.

3.5.6 *Benchmark* menggunakan 5-*Fold Stratified K-Fold*

Sebelum pelatihan final, dilakukan pengujian stabilitas menggunakan 5-*fold Stratified K-Fold* pada data latih. Pada setiap *fold*, metrik evaluasi (misalnya akurasi, *F1-score*, dan *ROC-AUC*) dihitung, lalu dirangkum melalui rata-rata dan simpangan baku untuk menilai konsistensi model. Nilai ringkasan metrik dan waktu eksekusi disajikan pada Bab 4.

3.5.7 Pelatihan Final Model Hibrida dan Penyimpanan Model

Setelah parameter terbaik tersedia, model final dibangun menggunakan arsitektur dua masukan:

- Cabang sekuens (SLD): *Input* panjang 63, *Embedding* dengan `input_dim=1000` dan `output_dim=embedding_dim`, lalu *Bidirectional LSTM* dengan `lstm_units`, diikuti *dropout*, dan *Dense* 64 unit beraktivasi ReLU.
- Cabang fitur *handcrafted*: *Input* berukuran 12, lalu *Dense* `dense_hc_units` beraktivasi ReLU dan *BatchNormalization*.

Kedua cabang digabung melalui *concatenate*, dilanjutkan *Dense* 32 unit ReLU, dan *output layer* sigmoid untuk klasifikasi biner.

Model dikompilasi menggunakan *Adam optimizer* dengan `learning_rate` dari parameter terbaik, fungsi rugi *binary cross-entropy*, dan metrik akurasi. Proses pelatihan menerapkan *EarlyStopping* dengan `monitor='val_accuracy'` dan `patience=3`, serta *ModelCheckpoint* untuk menyimpan model terbaik pada `bilstm_final_v2.keras`. Ringkasan epoch terbaik dan waktu pelatihan disajikan pada Bab 4.

3.5.8 Evaluasi pada Data Uji 20%

Evaluasi akhir dilakukan dengan memuat model `bilstm_final_v2.keras` dan memproses data uji menggunakan *tokenizer*, `MAX_LEN`, serta *scaler* yang sama seperti tahap pelatihan. Prediksi probabilitas dikonversi menjadi kelas biner menggunakan ambang 0,5, lalu dihitung metrik evaluasi (akurasi, *F1-score*, *ROC-AUC*, *confusion matrix*, dan *classification report*). Hasil numerik evaluasi disajikan pada Bab 4.