BAB III
METODE PENELITIAN

3.1 Objek Penelitian

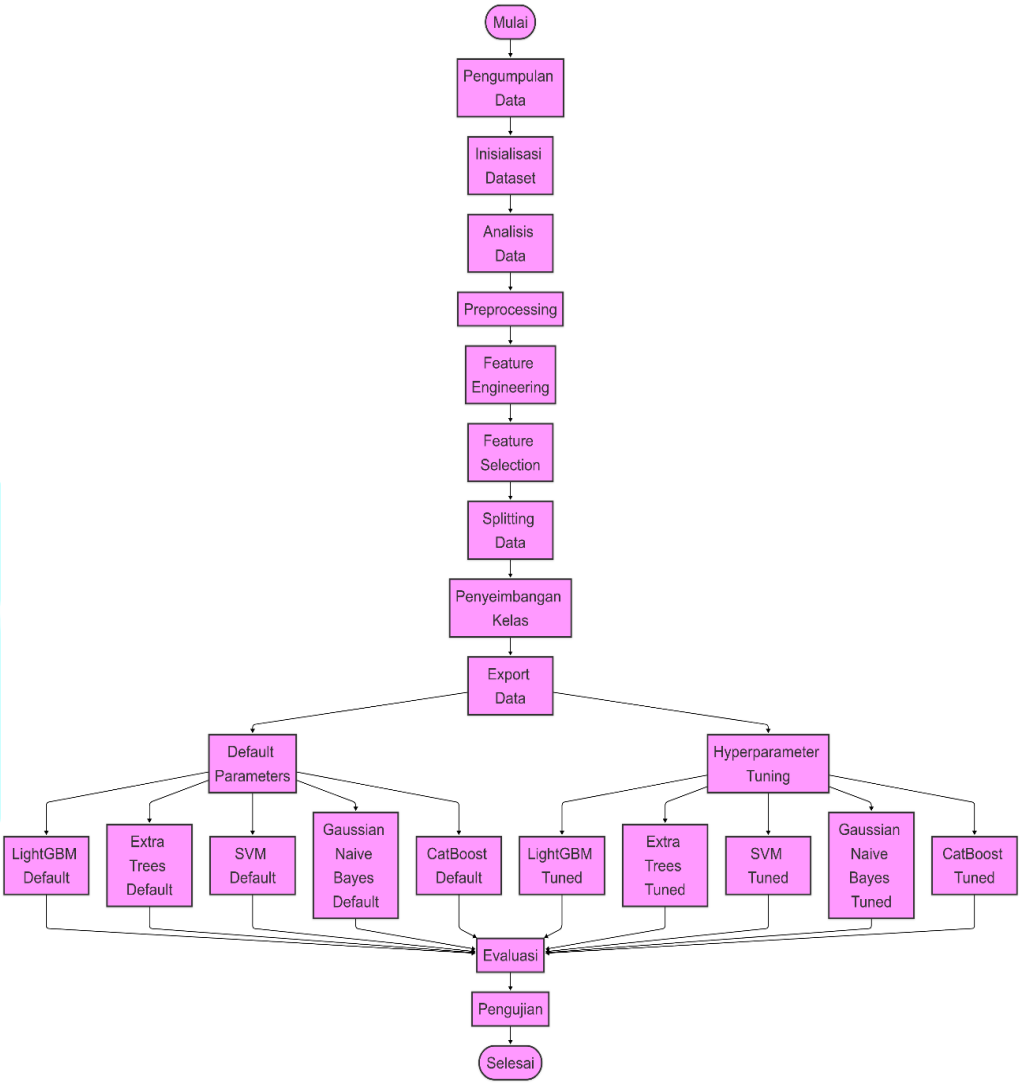
Penelitian ini menggunakan data transaksi *Ethereum* dengan tujuan untuk mendeteksi aktivitas penipuan pada jaringan *blockchain*. *Dataset* yang digunakan telah melalui proses pelabelan, di mana setiap transaksi diklasifikasikan sebagai transaksi normal atau transaksi penipuan. Data ini diambil dari platform *Kaggle* dan terdiri atas 9.841 baris transaksi, yang mencakup periode antara tahun 2016 hingga 2020.

Dalam penelitian ini, dilakukan implementasi dan perbandingan terhadap lima algoritma *machine learning*, yaitu *LightGBM*, *Extra Trees Classifier*, *Support Vector Machine* (SVM), *Gaussian Naïve Bayes*, dan *CatBoost*. Evaluasi performa model dilakukan melalui seleksi fitur menggunakan metode *wrapper Sequential Forward Selection* (SFS) serta penyetelan *hyperparameter* dengan teknik *Grid Search*. Seluruh model kemudian divalidasi menggunakan pendekatan 10-fold *cross-validation*. Rangkaian tahapan ini bertujuan untuk mengevaluasi kinerja masing-masing algoritma dalam mendeteksi aktivitas penipuan pada jaringan *Ethereum* secara lebih optimal dan menyeluruh.

3.2 Prosedur Penelitian

Metode penelitian ini terdiri dari beberapa tahapan utama yang dilakukan secara sistematis untuk menghasilkan model yang akurat dan andal. Proses dimulai dari pengumpulan dan analisis data untuk memahami struktur dan karakteristik *dataset*, dilanjutkan dengan *pre-processing* guna membersihkan data dari nilai yang tidak konsisten. Tahap berikutnya adalah *feature engineering* untuk membentuk fitur baru yang lebih informatif, diikuti oleh *feature selection* untuk memilih fitur yang paling relevan.

Setelah itu, data dibagi menjadi data latih dan data uji, kemudian dilakukan penyeimbangan data agar distribusi kelas seimbang. Data yang telah dipersiapkan kemudian di ekspor dan digunakan dalam proses evaluasi model menggunakan parameter *default*, lalu dilanjutkan dengan *hyperparameter tuning* untuk memperoleh performa model yang optimal. Seluruh tahapan ini tergambar secara lengkap dalam Gambar 3.1 berikut.



Gambar 3.1 Alur Tahapan Proses

3.2.1 Pengumpulan Data

Dataset ini memiliki struktur yang terdiri dari total 51 kolom, yaitu 50 fitur dan 1 kolom target. Fitur-fitur tersebut mencakup berbagai elemen yang berkaitan dengan aktivitas transaksi di jaringan *Ethereum*. Kolom target bernama *FLAG*, yang berisi klasifikasi biner untuk menunjukkan apakah suatu transaksi termasuk penipuan atau bukan, di mana label 0 menunjukkan transaksi *non-fraud* dan label 1 menunjukkan transaksi *fraud*. Secara keseluruhan, *dataset* ini terdiri dari 9.841 baris. Contoh data dapat dilihat pada Gambar 3.2 berikut:

	Index	Address	FLAG	Avg min between sent tnx	Avg min between received tnx	Time Diff between first and last (Mins)	Sent tnx	Received Tnx	Number of Created Contracts	Unique Received From Addresses	...	ERC20 min val sent	ERC20 max val sent	ERC20 avg val sent	ERC20 min val sent contract	ERC20 max val sent contract	ERC avg s contr
3907	975	0x6803d7849a3da62924efe3d88d420a5ea540ff87	0	2.97	0.00	8.92	3	2	0	2	...	0.000000	0.00000	0.000000	0.0	0.0	
7705	44	0x05dd62c007cde143b402fa5da3937c40c70b4b14	1	8921.53	538.88	126876.83	8	103	0	101	...	1871.902601	10389.14506	6130.523832	0.0	0.0	

2 rows × 50 columns

Gambar 3.2 *Sample* Data

3.2.2 Inisialisasi Dataset

Setelah data berhasil dikumpulkan, langkah berikutnya adalah muat *dataset* ke dalam program untuk di analisis menggunakan *Visual Studio Code (VSCode)*. Proses impor dilakukan dengan menggunakan pustaka *pandas*, yang menawarkan struktur data yang efisien untuk mengelola dataset besar dan kompleks. *Dataset* kemudian dimuat dalam format *DataFrame*, sehingga memudahkan manipulasi dan analisis data.

3.2.3 Analisis Data

Setelah *dataset* berhasil di muat, langkah selanjutnya adalah melakukan analisis pada data. Analisis data bertujuan untuk memahami informasi yang ada, sehingga data tersebut dapat diseleksi dan dikelola dengan lebih baik. Tahap analisis melibatkan eksplorasi mendalam terhadap karakteristik *dataset* menggunakan *pandas*. Proses ini mencakup pemeriksaan nilai yang hilang *(missing values)*, identifikasi duplikasi data, analisis tipe data setiap kolom. Proses ini sangat penting untuk memahami karakteristik *dataset* dan menentukan langkah-langkah berikutnya dalam pemrosesan data.

3.2.4 Pre-processing

Setelah melakukan analisis awal, langkah selanjutnya adalah *pre-processing* data. Tujuan dari tahap ini adalah untuk menyiapkan data yang terstruktur dan siap digunakan dalam proses selanjutnya. Salah satu langkah utama yang dilakukan adalah menangani data yang hilang dengan menggunakan *SimpleImputer*. Alat ini untuk menggantikan nilai yang hilang dengan nilai lain yang sesuai. Untuk data numerik, nilai yang hilang akan digantikan dengan rata-rata, sementara untuk data *kategorikal*, nilai yang paling sering muncul dipilih sebagai pengganti.

Selanjutnya, dilakukan pengecekan untuk menghapus baris duplikat. Kemudian kolom-kolom yang tidak relevan, seperti Index dan *Address*, akan dihapus karena tidak memberikan kontribusi signifikan terhadap prediksi. Setelah itu, data dibagi menjadi dua bagian, fitur (X) dan target (y). Fitur berisi informasi yang digunakan untuk melakukan prediksi, sedangkan target adalah nilai yang ingin kita peroleh. Proses ini dilakukan dengan memilih kolom yang relevan untuk fitur dan memisahkan kolom yang menjadi target. Untuk menjaga konsistensi, nama-nama kolom pada fitur juga diubah, mengganti spasi dengan garis bawah.

3.2.5 Feature engineering

Setelah proses *pre-processing* data selesai, langkah selanjutnya adalah *Feature Engineering*, sebuah tahap kreatif yang bertujuan untuk meningkatkan kualitas fitur yang akan digunakan oleh model. Pada tahap ini, dilakukan transformasi fitur menggunakan teknik *labelencoder*, yang berfungsi mengubah fitur kategori menjadi representasi numerik. Metode ini memungkinkan model untuk memproses data *kategorikal* dengan cara yang lebih efisien. Dalam proses ini, setiap nilai pada kolom *kategorikal* diubah menjadi label numerik yang dapat dipahami oleh model.

Selain itu, kami juga menerapkan transformasi logaritmik pada fitur numerik untuk mengatasi *skewness* dan membuat distribusi data menjadi lebih normal. Penerapan transformasi ini dilakukan menggunakan *FunctionTransformer*, yang memungkinkan kami menerapkan fungsi logaritmik pada data numerik yang nilainya lebih besar dari nol. Transformasi ini berperan dalam memperbaiki distribusi data, sehingga memudahkan model dalam mengenali pola-pola yang ada.

3.2.6 Feature Selection

Setelah fitur-fitur disiapkan dan direkayasa, langkah berikutnya adalah melakukan *Feature Selection*, yang bertujuan untuk memilih fitur terbaik yang akan digunakan dalam model *machine learning*. Tahap ini sangat penting karena dapat meningkatkan kinerja model dengan mengurangi kompleksitas sekaligus mencegah *overfitting*.

Salah satu teknik yang digunakan dalam proses ini adalah *Sequential Feature Selection* (SFS). Teknik ini bekerja dengan memilih fitur satu per satu, berdasarkan kinerjanya. Untuk keperluan ini, model yang diterapkan adalah *Gradient Boosting Classifier*, yang memiliki kemampuan untuk menilai pentingnya setiap fitur. Proses pemilihan fitur dilakukan melalui seleksi maju *(forward selection)*, di mana fitur terbaik ditambahkan secara bertahap berdasarkan evaluasi kinerja model menggunakan metode *cross-validation*, dengan *f1-score* yang dijadikan sebagai metrik evaluasi. Setelah fitur-fitur terbaik teridentifikasi, hanya fitur-fitur yang relevan yang akan dipertahankan, sementara fitur-fitur lainnya akan disingkirkan.

3.2.7 Splitting Data

Setelah memilih fitur-fitur yang relevan, langkah selanjutnya adalah membagi dataset menjadi dua bagian, yaitu data latih *(training)* dan data uji *(testing)*. Proses

ini dikenal sebagai *Train Test Split*. Tujuan dari langkah ini adalah untuk melatih model menggunakan data latih, kemudian menguji kinerjanya pada data uji yang tidak digunakan selama pelatihan. Dengan cara ini, kita dapat mengukur seberapa baik model dapat menggeneralisir terhadap data yang belum pernah dilihat sebelumnya.

Pada tahap ini, pembagian data dilakukan dengan proporsi 4:1 atau setara dengan 80% untuk pelatihan dan 20% untuk pengujian. Pembagian ini dilakukan secara acak, dan dengan menggunakan parameter *random_state* untuk memastikan bahwa hasil yang diperoleh dapat direproduksi.

3.2.8 Penyeimbangan Kelas

Setelah membagi data menjadi dua set, yaitu data latih dan data uji, langkah selanjutnya adalah menangani masalah ketidakseimbangan kelas yang muncul dalam data. Ketidakseimbangan kelas ini dapat berdampak signifikan pada performa model, terutama dalam konteks klasifikasi, di mana terdapat kecenderungan bagi model untuk lebih sering memprediksi kelas mayoritas dan mengabaikan kelas minoritas.

Synthetic Minority Oversampling Technique (SMOTE) merupakan algoritma *pre-processing* yang sering digunakan untuk mengatasi masalah ketidakseimbangan data. Untuk mengatasi permasalahan ini, kita dapat menggunakan Teknik SMOTE. Teknik ini berfungsi untuk menghasilkan sampel sintetis bagi kelas minoritas untuk menyeimbangkan, sehingga proporsinya menjadi lebih seimbang dengan kelas mayoritas. Melalui proses ini, model dapat belajar dengan lebih efektif meskipun data yang tersedia tidak sepenuhnya mewakili, yang pada akhirnya meningkatkan kemampuan model dalam memprediksi kedua kelas secara adil.

3.2.9 Export Data

Setelah menyelesaikan proses *pre-processing*, penyeimbangan kelas, dan pemilihan fitur, langkah berikutnya adalah mengekspor data yang telah diproses. Data latih, yang telah mengalami *oversampling* menggunakan *SMOTE*, bersama dengan data uji, disimpan dalam format CSV untuk memudahkan penggunaannya dalam pemodelan. Target data untuk latih dan uji juga disimpan sebagai array satu dimensi dalam format yang sama.

Data yang telah disiapkan ini akan diterapkan dalam dua pendekatan pemodelan. Pertama, dengan menggunakan parameter default dari berbagai algoritma *machine learning*, seperti *LightGBM, Extra Trees, SVM, Gaussian Naive Bayes*, dan *CatBoost*. Kedua, dengan melakukan *tuning hyperparameter* untuk mengoptimalkan kinerja model.

Alur proses pemodelan untuk setiap algoritma ini akan dijelaskan lebih rinci pada sub bab berikutnya, di mana setiap langkah, mulai dari bahan atau alat *(library)* yang digunakan, pemilihan parameter, serta metrik evaluasi yang dijadikan tolak ukur perbandingan antara model-model yang diterapkan akan dibahas secara mendalam.

3.2.10 Implementasi LightGBM Default Parameter

1. *Tools* dan *Library* yang digunakan:
    a. *VSCode* - digunakan untuk menulis dan mengelola kode python
    b. *Python* 3.12.3 – sebagai bahasa pemrograman
    c. *Pandas* - untuk manipulasi dan membaca data
    d. *NumPy* - untuk operasi numerik dan membaca data
    e. *Joblib* - untuk menyimpan model
    f. *os* – untuk cek lokasi penyimpanan model
    g. *LGBMClassifier* – algoritma untuk implementasi model
    h. *Dataset* yang sudah di *pre-processing*
        a) *X_train*: berisi fitur-fitur hasil *pre-processing*
        b) *y_train*: berisi label hasil *pre-processing*
2. Proses Implementasi *LightGBM* untuk pembuatan model klasifikasi penipuan dengan menggunakan *default* parameter

Algoritma 1. Model Algoritma *LightGBM default* parameter

Algoritma: Modeling Klasifikasi Penipuan dengan *LightGBM*

Input: *Dataset* hasil *pre-processing* (*X_train*, *y_train*)
Output: Model Klasifikasi Penipuan *Ethereum*
1. Baca *X_train* dari *file* CSV menggunakan *pandas*
2. Baca *y_train* dari *file* CSV menggunakan *numpy*
3. Menerapkan *default* parameter dengan menetapkan *n_estimators*=100
4. *Dataset* dilatih menggunakan model LGBM, yang membangun pohon keputusan secara bertahap dengan metode *leaf-wise tree*. Proses ini dilakukan seperti yang ditunjukkan pada Gambar 2.2 dengan menggunakan persamaan (1)
5.Waktu Komputasi juga di hitung selama proses pelatihan

6. Hasil Model klasifikasi diperoleh setelah proses *training* selesai
7. Setelah diperoleh, model klasifikasi disimpan ke lokasi yang telah ditentukan

3.2.11 Implementasi LightGBM Hyperparameter

1. *Tools* dan *Library* yang digunakan:
    a. *VSCode* - digunakan untuk menulis dan mengelola kode *python*
    b. *Python* 3.12.3 – sebagai bahasa pemrograman
    c. *Pandas* - untuk manipulasi dan membaca data
    d. *NumPy* - untuk operasi numerik dan membaca data
    e. *Joblib* - untuk menyimpan model
    f. *os* – untuk cek lokasi penyimpanan model
    g. *GridSearchCV* – untuk pencarian *hyperparameter* terbaik
    h. *StratifiedKFold* – untuk melakukan validasi silang
    i. *LGBMClassifier* – algoritma untuk implementasi model
    j. *Dataset* yang sudah di *pre-processing*
        a) *X_train*: berisi fitur-fitur hasil *pre-processing*
        b) *y_train*: berisi label hasil *preprocessing*
2. Proses Implementasi *LightGBM* untuk pembuatan model klasifikasi penipuan dengan menggunakan *hyperparameter*

Algoritma 2. Model Algoritma *LightGBM hyperparameter*

Algoritma: Modeling Klasifikasi Penipuan dengan *LightGBM*

Input: *Dataset* hasil *pre-processing* (*X_train*, *y_train*)
Output: Model Klasifikasi Penipuan *Ethereum*
1. Baca *X_train* dari *file* CSV menggunakan *pandas*
2. Baca *file y_train* menggunakan *numpy*
3. Mendefinisikan objek *StratifiedKFold* dengan *n_splits*=10 untuk membagi data menjadi 10 lipatan
4.Tentukan parameter *grid* untuk pencarian *hyperparameter* yang mencakup
n_estimators dengan nilai 150 dan 300
5. Mendefinisikan objek dari kelas *LGBMClassifier* untuk digunakan sebagai model klasifikasi
6. Tentukan objek *GridSearchCV* dengan parameter-parameter sebelumnya, yaitu model, *grid* parameter, dan *cross-validation*
7. Data *training* dilatih menggunakan model *LightGBM* sesuai dengan persamaan (1), dengan metode *fit* serta penyesuaian parameter yang dioptimalkan melalui *GridSearch*. Selain itu, regulasi dalam model diterapkan menggunakan parameter lambda L2, sebagaimana dijelaskan dalam persamaan (3)
8.Waktu Komputasi juga di hitung selama proses pelatihan

9. Hasil Model klasifikasi diperoleh setelah proses *training* selesai
10. Setelah diperoleh, Model klasifikasi disimpan ke lokasi yang telah ditentukan

3.2.12 Implementasi Extra Trees Default Parameter

1. *Tools* dan *Library* yang digunakan:
    a. *VSCode* - digunakan untuk menulis dan mengelola kode *python*
    b. *Python* 3.12.3 – sebagai bahasa pemrograman
    c. *Pandas* - untuk manipulasi dan membaca data
    d. *NumPy* - untuk operasi numerik dan membaca data
    e. *Joblib* - untuk menyimpan model
    f. *os* – untuk cek lokasi penyimpanan model
    g. *ExtraTreesClassifier* – algoritma untuk implementasi model
    h. *Dataset* yang sudah di *pre-processing*
        a) *X_train*: berisi fitur-fitur hasil *pre-processing*
        b) *y_train*: berisi label hasil *pre-processing*
2. Proses Implementasi *Extra Trees* untuk pembuatan model klasifikasi penipuan dengan menggunakan *default* parameter

Algoritma 3. Model Algoritma *Extra Trees default* parameter

Algoritma: Modeling Klasifikasi Penipuan dengan *Extra Trees*

Input: *Dataset* hasil *preprocessing* (*X_train, y_train*)
Output: Model Klasifikasi Penipuan *Ethereum*
1. Baca *X_train* dari file CSV menggunakan *pandas*
2. Baca *y_train* dari file CSV menggunakan *numpy*
3. Menerapkan parameter *default* pada model *Extra Trees* dengan menetapkan n_*estimators*=500
4. *Dataset* dilatih menggunakan model *Extra Trees* sesuai pada persamaan (4), hasil akhir ditentukan berdasarkan voting terbanyak, seperti pada Gambar 2.3
5.Waktu Komputasi juga di hitung selama proses pelatihan
6. Hasil Model klasifikasi diperoleh setelah proses *training* selesai
7. Setelah diperoleh, model klasifikasi disimpan ke lokasi yang telah ditentukan

3.2.13 Implementasi Extra Trees Hyperparameter

1. *Tools* dan *Library* yang digunakan:
    a. *VSCode* - digunakan untuk menulis dan mengelola kode *python*
    b. *Python* 3.12.3 – sebagai bahasa pemrograman
    c. *Pandas* - untuk manipulasi dan membaca data
    d. *NumPy* - untuk operasi numerik dan membaca data

e. *Joblib* - untuk menyimpan model
f. *os* – untuk cek lokasi penyimpanan model
g. *GridSearchCV* – untuk pencarian *hyperparameter* terbaik
h. *StratifiedKFold* – untuk melakukan validasi silang
i. *ExtraTreesClassifier* – algoritma untuk implementasi model
j. *Dataset* yang sudah di *pre-processing*
    a) *X_train*: berisi fitur-fitur hasil *pre-processing*
    b) *y_train*: berisi label hasil *pre-processing*

2. Proses Implementasi *Extra Trees* untuk pembuatan model klasifikasi penipuan dengan menggunakan *hyperparameter*

Algoritma 4. Model Algoritma *Extra Trees hyperparameter*

Algoritma: Modeling Klasifikasi Penipuan dengan *Extra Trees*
Input: *Dataset* hasil *pre-processing* (*X_train*, *y_train*) *Output*: Model Klasifikasi Penipuan *Ethereum* 1. Baca *X_train* dari file CSV menggunakan *pandas* 2. Baca *file y_train* menggunakan *numpy* 3. Mendefinisikan objek *StratifiedKFold* dengan *n_splits*=10 untuk membagi data menjadi 10 lipatan 4. Tentukan parameter grid untuk pencarian *hyperparameter* yang mencakup *n_estimators* dengan nilai 100, 200 dan 300 5. Mendefinisikan objek dari kelas *ExtraTreesClassifier* untuk digunakan sebagai model klasifikasi 6. Tentukan objek *GridSearchCV* dengan parameter-parameter sebelumnya, yaitu model, grid parameter, dan *cross-validation* 7. *Dataset* dilatih menggunakan model *Extra Trees* sesuai dengan persamaan (4), dengan penyesuaian *hyperparameter* melalui *GridSearch* untuk menemukan kombinasi terbaik. Hasil akhir ditentukan berdasarkan voting terbanyak, seperti yang ditunjukkan pada Gambar 2.3 8. Waktu Komputasi juga di hitung selama proses pelatihan 9. Hasil model klasifikasi diperoleh setelah proses *training* selesai 10. Setelah diperoleh, model klasifikasi disimpan ke lokasi yang telah ditentukan

3.2.14 Implementasi Support Vector Machine Default Parameter

1. *Tools* dan *Library* yang digunakan:
    a. *VSCode* - digunakan untuk menulis dan mengelola kode *python*
    b. *Python* 3.12.3 – sebagai bahasa pemrograman
    c. *Pandas* - untuk manipulasi dan membaca data
    d. *NumPy* - untuk operasi numerik dan membaca data
    e. *Joblib* - untuk menyimpan model

f. *os* – untuk cek lokasi penyimpanan model
g. SVC – algoritma untuk implementasi model
h. *minmax* – untuk normalisasi data
i. *Dataset* yang sudah di *pre-processing*
    a) *X_train*: berisi fitur-fitur hasil *pre-processing*
    b) *y_train*: berisi label hasil *pre-processing*

2. Proses Implementasi *Support Vector Machine* untuk pembuatan model klasifikasi penipuan dengan menggunakan *default* parameter

Algoritma 5. Model Algoritma *Support Vector Machine default* parameter

Algoritma: Modeling Klasifikasi Penipuan dengan *SVM*

Input: *Dataset* hasil *pre-processing* (*X_train*, *y_train*)
Output: Model Klasifikasi Penipuan *Ethereum*
1. Baca *X_train* dari *file* CSV menggunakan *pandas*
2. Baca *y_train* dari *file* CSV menggunakan *numpy*
3. Normalisasi Data menggunakan *minmax*
4. Terapkan parameter *default* pada model SVM dengan menetapkan C = 1
5. *Dataset* dilatih menggunakan model SVM dengan *kernel* RBF sesuai dengan persamaan (6), yang berfungsi untuk membedakan kelas dengan margin optimal
6. Waktu Komputasi juga di hitung selama proses pelatihan
7. Hasil model klasifikasi diperoleh setelah proses *training* selesai
8. Setelah diperoleh, model klasifikasi disimpan ke lokasi yang telah ditentukan

3.2.15 Implementasi *Support Vector Machine Hyperparameter*

1. *Tools* dan *Library* yang digunakan:
    a. *VSCode* - digunakan untuk menulis dan mengelola kode *python*
    b. *Python* 3.12.3 – sebagai bahasa pemrograman
    c. *Pandas* - untuk manipulasi dan membaca data
    d. *NumPy* - untuk operasi numerik dan membaca data
    e. *Joblib* - untuk menyimpan model
    f. *os* – untuk cek lokasi penyimpanan model
    g. *GridSearchCV* – untuk pencarian *hyperparameter* terbaik
    h. *StratifiedKFold* – untuk melakukan validasi silang
    i. SVC – algoritma untuk implementasi model
    j. *minmax* – untuk normalisasi data
    k. *Dataset* yang sudah di *pre-processing*
        a) *X_train*: berisi fitur-fitur hasil *pre-processing*

b) *y_train*: berisi label hasil *pre-processing*

2. Proses Implementasi *Support Vector Machine* untuk pembuatan model klasifikasi penipuan dengan menggunakan *hyperparameter*

Algoritma 6. Model Algoritma *Support Vector Machine hyperparameter*

Algoritma: Modeling Klasifikasi Penipuan dengan *SVM*

Input: *Dataset* hasil *pre-processing* (*X_train*, *y_train*)
Output: Model Klasifikasi Penipuan *Ethereum*
1. Baca *X_train* dari file CSV menggunakan *pandas*
2. Baca *file y_train* menggunakan *numpy*
3. Normalisasi data menggunakan *minmax*
4. Mendefinisikan objek *StratifiedKFold* dengan *n_splits*=10 untuk membagi data menjadi 10 lipatan
5. Tentukan parameter *grid* untuk pencarian *hyperparameter* yang mencakup C dengan nilai 0.01, 0.1, 1, 10, 100, 1000
6. Mendefinisikan objek dari kelas SVC untuk digunakan sebagai model klasifikasi
7. Tentukan objek *GridSearchCV* dengan parameter-parameter sebelumnya, yaitu model, *grid* parameter, dan *cross-validation*
8. *Dataset* dilatih menggunakan model SVM dengan *kernel* RBF sesuai dengan persamaan (6), dengan penyesuaian *hyperparameter* melalui *GridSearch* untuk menemukan kombinasi terbaik dari nilai C
9. Waktu Komputasi juga di hitung selama proses pelatihan
10. Hasil model klasifikasi diperoleh setelah proses *training* selesai
11. Setelah diperoleh, model klasifikasi disimpan ke lokasi yang telah ditentukan

3.2.16 Implementasi Gaussian Naïve Bayes Default Parameter

1. *Tools* dan *Library* yang digunakan:
    a. *VSCode* - digunakan untuk menulis dan mengelola kode *python*
    b. *Python* 3.12.3 – sebagai bahasa pemrograman
    c. *Pandas* - untuk manipulasi dan membaca data
    d. *NumPy* - untuk operasi numerik dan membaca data
    e. *Joblib* - untuk menyimpan model
    f. *os* – untuk cek lokasi penyimpanan model
    g. *GaussianNB* – algoritma untuk implementasi model
    h. *Dataset* yang sudah di *pre-processing*
        a) *X_train*: berisi fitur-fitur hasil *pre-processing*
        b) *y_train*: berisi label hasil *pre-processing*
2. Proses Implementasi *Gaussian Naïve Bayes* untuk pembuatan model klasifikasi penipuan dengan menggunakan *default* parameter

Algoritma 7. Model Algoritma *Gaussian Naïve Bayes default* parameter

Algoritma: Modeling Klasifikasi Penipuan dengan *Gaussian Naive Bayes*
Input: *Dataset* hasil *pre-processing* (*X_train*, *y_train*) *Output*: Model Klasifikasi Penipuan *Ethereum* 1. Baca *X_train* dari *file* CSV menggunakan *pandas* 2. Baca y_*train* dari *file* CSV menggunakan *numpy* 3. Menerapkan parameter *default* pada model *Gaussian Naïve Bayes* dengan menetapkan v*ar_smoothing*=1e-9 4. *Dataset* dilatih menggunakan model *Gaussian Naïve Bayes* sesuai dengan persamaan (8) berdasarkan *Teorema Bayes*. Probabilitas kelas dihitung menggunakan fungsi kepadatan peluang (PDF) sebagaimana dijelaskan dalam persamaan (9), dengan standar deviasi dihitung sesuai persamaan (10) 5. Waktu Komputasi juga di hitung selama proses pelatihan 6. Hasil model klasifikasi diperoleh setelah proses *training* selesai 7. Setelah diperoleh, model klasifikasi disimpan ke lokasi yang telah ditentukan

3.2.17 Implementasi Gaussian Naïve Bayes Hyperparameter

1. *Tools* dan *Library* yang digunakan:
    a. *VSCode* - digunakan untuk menulis dan mengelola kode *python*
    b. *Python* 3.12.3 – sebagai bahasa pemrograman
    c. *Pandas* - untuk manipulasi dan membaca data
    d. *NumPy* - untuk operasi numerik dan membaca data
    e. *Joblib* - untuk menyimpan model
    f. *os* – untuk cek lokasi penyimpanan model
    g. *GridSearchCV* – untuk pencarian *hyperparameter* terbaik
    h. *StratifiedKFold* – untuk melakukan validasi silang
    i. *GaussianNB* – algoritma untuk implementasi model
    j. *Dataset* yang sudah di *pre-processing*
        a) *X_train*: berisi fitur-fitur hasil *pre-processing*
        b) *y_train*: berisi label hasil *pre-processing*
2. Proses Implementasi *Gaussian Naïve Bayes* untuk pembuatan model klasifikasi penipuan dengan menggunakan *hyperparameter*

Algoritma 8. Model Algoritma *Gaussian Naïve Bayes hyperparameter*

Algoritma: Modeling Klasifikasi Penipuan dengan *Gaussian Naive Bayes*
Input: *Dataset* hasil *pre-processing* (*X_train*, *y_train*) *Output*: Model Klasifikasi Penipuan *Ethereum* 1. Baca *X_train* dari file CSV menggunakan *pandas* 2. Baca *file y_train* menggunakan *numpy*

3. Mendefinisikan objek *StratifiedKFold* dengan *n_splits*=10 untuk membagi data menjadi 10 lipatan
4. Tentukan parameter grid untuk pencarian *hyperparameter* yang mencakup *var_smoothing* dengan nilai 1e-15 hingga 1e-3
5. Mendefinisikan objek dari kelas *GaussianNB* untuk digunakan sebagai model klasifikasi
6. Tentukan objek *GridSearchCV* dengan parameter-parameter sebelumnya, yaitu model, *grid* parameter, dan *cross-validation*
7. *Dataset* dilatih menggunakan model *Gaussian Naive Bayes* sesuai dengan persamaan (8) berdasarkan *Teorema Bayes*. Penyesuaian *hyperparameter* dilakukan melalui *GridSearch* dengan variasi nilai *var smoothing* dalam rentang 1e-15 hingga 1e-3. Probabilitas kelas dihitung menggunakan fungsi kepadatan peluang (PDF) sebagaimana dijelaskan pada persamaan (9), dengan standar deviasi dihitung sesuai persamaan (10)
8. Waktu Komputasi juga di hitung selama proses pelatihan
9. Hasil model klasifikasi diperoleh setelah proses *training* selesai
10. Setelah diperoleh, model klasifikasi disimpan ke lokasi yang telah ditentukan

3.2.18 Implementasi CatBoost Default Parameter

1. *Tools* dan *Library* yang digunakan:
    a. *VSCode* - digunakan untuk menulis dan mengelola kode *python*
    b. *Python* 3.12.3 – sebagai bahasa pemrograman
    c. *Pandas* - untuk manipulasi dan membaca data
    d. *NumPy* - untuk operasi numerik dan membaca data
    e. *Joblib* - untuk menyimpan model
    f. *os* – untuk cek lokasi penyimpanan model
    g. *CatBoostClassifier* – algoritma untuk implementasi model
    h. *Dataset* yang sudah di *pre-processing*
        a) *X_train*: berisi fitur-fitur hasil *pre-processing*
        b) *y_train*: berisi label hasil *pre-processing*
2. Proses Implementasi *CatBoost* untuk pembuatan model klasifikasi penipuan dengan menggunakan *default* parameter

Algoritma 9. Model Algoritma *CatBoost default* parameter

Algoritma: Modeling Klasifikasi Penipuan dengan *CatBoost*

Input: *Dataset* hasil *pre-processing* (*X_train*, *y_train*)
Output: Model Klasifikasi Penipuan *Ethereum*
1. Baca *X_train* dari *file* CSV menggunakan *pandas*
2. Baca *y_train* dari *file* CSV menggunakan *numpy*
3. Menerapkan parameter *default* pada model *CatBoost* dengan menetapkan *iterations*=1000

4. Dataset dilatih menggunakan model *CatBoost* sesuai dengan persamaan (12), yang menjelaskan rumus dasar dari cara kerja algoritma ini. Proses pembelajaran dilakukan dengan pendekatan yang ditunjukkan pada Gambar 2.4
8. Waktu Komputasi juga di hitung selama proses pelatihan
9. Hasil model klasifikasi diperoleh setelah proses *training* selesai
10. Setelah diperoleh, model klasifikasi disimpan ke lokasi yang telah ditentukan

3.2.19 Implementasi CatBoost Hyperparameter

1. *Tools* dan *Library* yang digunakan:
    a. *VSCode* - digunakan untuk menulis dan mengelola kode *python*
    b. *Python* 3.12.3 – sebagai bahasa pemrograman
    c. *Pandas* - untuk manipulasi dan membaca data
    d. *NumPy* - untuk operasi numerik dan membaca data
    e. *Joblib* - untuk menyimpan model
    f. *os* – untuk cek lokasi penyimpanan model
    g. *GridSearchCV* – untuk pencarian hyperparameter terbaik
    h. *StratifiedKFold* – untuk melakukan validasi silang
    i. *CatBoostClassifier* – algoritma untuk implementasi model
    j. *Dataset* yang sudah di *pre=processing*
        a) *X_train*: berisi fitur-fitur hasil *pre-processing*
        b) *y_train*: berisi label hasil *pre-processing*
2. Proses Implementasi *CatBoost* untuk pembuatan model klasifikasi penipuan dengan menggunakan *hyperparameter*

Algoritma 10. Model Algoritma *CatBoost hyperparameter*

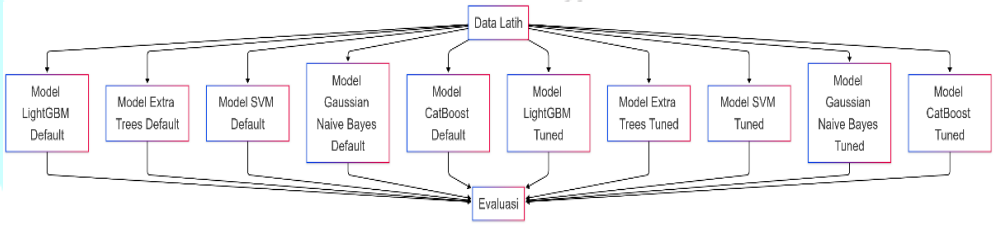
Algoritma: Modeling Klasifikasi Penipuan dengan *CatBoost*

Input: *Dataset* hasil *pre-processing* (*X_train*, *y_train*)
Output: Model Klasifikasi Penipuan *Ethereum*
1. Baca *X_train* dari *file* CSV menggunakan *pandas*
2. Baca *file y_train* menggunakan *numpy*
3. Mendefinisikan objek *StratifiedKFold* dengan *n_splits*=10 untuk membagi data menjadi 10 lipatan
4. Tentukan parameter grid untuk pencarian *hyperparameter* yang mencakup *iterations* dengan nilai 500 dan 1500
5. Mendefinisikan objek dari kelas *CatBoostClassifier* untuk digunakan sebagai model klasifikasi
6. Tentukan objek *GridSearchCV* dengan parameter-parameter sebelumnya, yaitu model, *grid* parameter, dan *cross-validation*
7. *Dataset* dilatih menggunakan model *CatBoost* sesuai dengan persamaan (12), yang menjelaskan rumus dasar dari cara kerja algoritma ini.

Penyesuaian *hyperparameter* dilakukan melalui *GridSearch* untuk mengoptimalkan kinerja model, dengan parameter jumlah iterasi dan lainnya. Proses pembelajaran mengikuti pendekatan yang ditunjukkan pada Gambar 2.4
8. Waktu Komputasi juga di hitung selama proses pelatihan
9. Hasil model klasifikasi diperoleh setelah proses *training* selesai
10. Setelah diperoleh, model klasifikasi disimpan ke lokasi yang telah ditentukan

3.2.20 Evaluasi

Evaluasi dilakukan dengan menggunakan data latih yang telah dihasilkan dari proses pembagian *dataset* dengan proporsi 80% untuk data latih. Dalam tahap ini, berbagai model *machine learning* diterapkan pada data latih, baik dengan parameter *default* maupun setelah dilakukan *tuning*. Skema tahapan evaluasi dapat dilihat pada Gambar 3.3 di bawah ini.

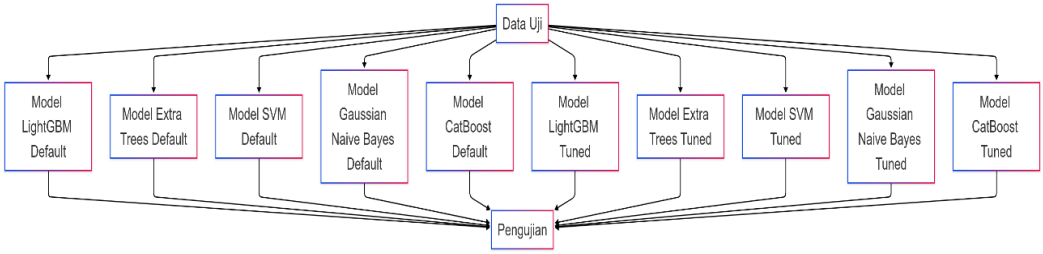


Gambar 3.3 Skema Evaluasi Model Algoritma

Evaluasi bertujuan untuk memastikan bahwa tes yang dilakukan adalah tepat dan untuk mengukur *accuracy* hasil serta menemukan hasil tes yang paling optimal. Pengukuran dilakukan menggunakan metrik evaluasi seperti *accuracy*, *precision*, *recall*, dan *F1-score*, yang masing-masing dihitung menggunakan rumus pada Persamaan (13), (14), (15), dan (16).

3.2.21 Pengujian

Pengujian dilakukan menggunakan data uji yang diperoleh dari proses pembagian *dataset* dengan proporsi 20%. Pada tahap ini, berbagai model *machine learning* diterapkan pada data uji, baik dengan parameter *default* maupun setelah dilakukan *tuning*. Skema tahapan pengujian dapat dilihat pada Gambar 3.4.



Gambar 3.4 Skema Pengujian Model Algoritma

Pengujian ini dilakukan untuk mengevaluasi performa model berdasarkan sejumlah metrik standar, yaitu *accuracy*, *precision*, *recall*, dan *F1-score*. Masing-masing metrik memberikan gambaran mengenai kemampuan model dalam melakukan klasifikasi secara tepat, dan dihitung menggunakan rumus pada Persamaan (13) hingga (16). Fokus utama dari pengujian ini adalah untuk menilai sejauh mana model mampu menghasilkan prediksi yang akurat terhadap data uji yang diberikan