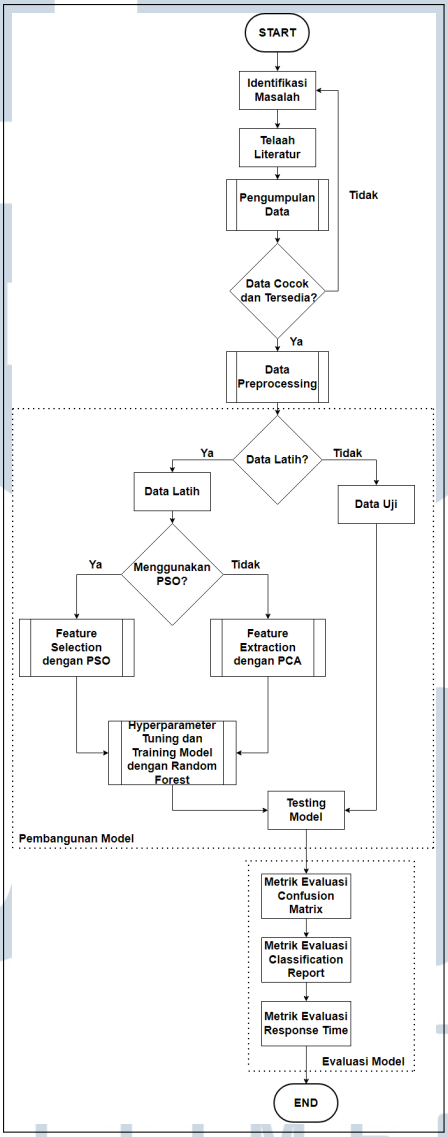
BAB 3
METODOLOGI PENELITIAN

Gambar 3.1 menunjukkan *Flowchart* dari alur penelitian dilakukan mulai dari identifikasi masalah hingga evaluasi model.



Gambar 3.1. *Flowchart* Alur Penelitian

3.1 Identifikasi Masalah

Peneliti melakukan identifikasi masalah dengan mencari dan mengumpulkan informasi untuk memahami permasalahan secara mendalam.

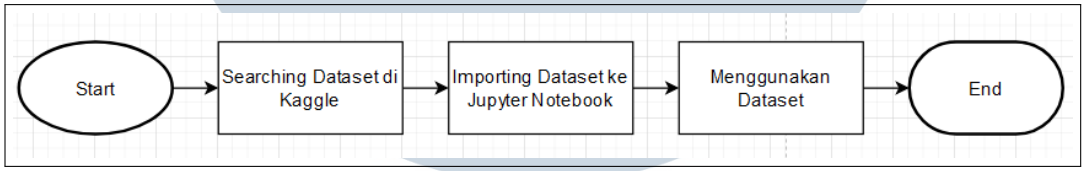
Alat untuk membantu proses ini menggunakan perangkat lunak pencarian seperti *Google* dan *Google Scholar*. *Google Scholar* dapat membantu dalam mengakses berbagai sumber informasi, artikel dan studi kasus mengenai permasalahan yang berkaitan dengan *ransomware* dan kekurangan yang ada pada IDS.

3.2 Telaah Literatur

Peneliti memiliki telaah literatur yang berisi tentang teori menjelaskan *ransomware*, IDS, dan algoritma yang akan digunakan seperti RF dan PSO.

3.3 Pengumpulan Data

Gambar 3.2 adalah alur dalam pengumpulan data yang akan dipakai untuk penelitian.



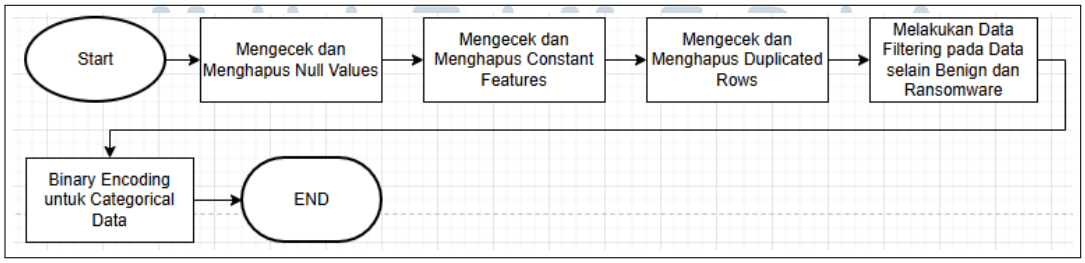
Gambar 3.2. *Flowchart* Alur Pengumpulan Data

Dataset yang digunakan adalah dataset dipilih dari yang tersedia di kaggle seperti *dataset* CIC-MalMem [12] dengan jumlah data sebesar 58596 *entries* dan kolom sebanyak 57.

3.4 Pembuatan Model *Machine Learning*

3.4.1 Data *Pre-processing*

Gambar 3.3 merepresentasikan alur dalam data *pre-processing* yang akan digunakan untuk penelitian.



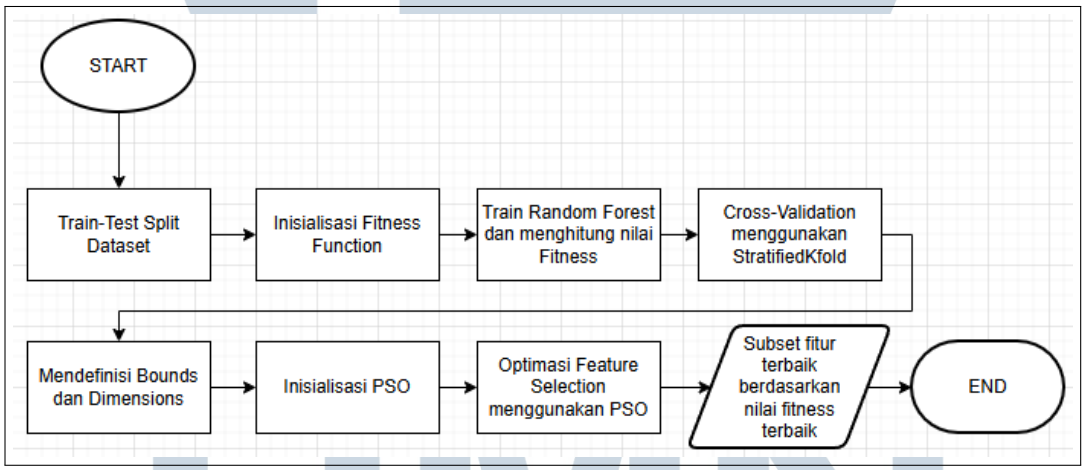
Gambar 3.3. *Flowchart* Alur Data *Pre-processing*

Peneliti akan melakukan data *pre-processing* yaitu mengolah data mentah yang telah dikumpulkan dari *dataset* agar dapat diproses. Proses ini terdiri dari membersihkan data *(data cleaning)* seperti menghapus *null values, constant features* dan *duplicated rows* kemudian menyaring data selain *benign* dan *ransomware*. Tahap *pre-processing* yang terakhir adalah mengubah data *(data transformation) categorical* menjadi *numerical* agar dapat digunakan pada tahap selanjutnya yaitu pembangunan model.

3.4.2 Pembangunan Model

A *Feature Selection* dengan PSO

Gambar 3.4 menunjukkan alur dari *Feature Selection* menggunakan PSO



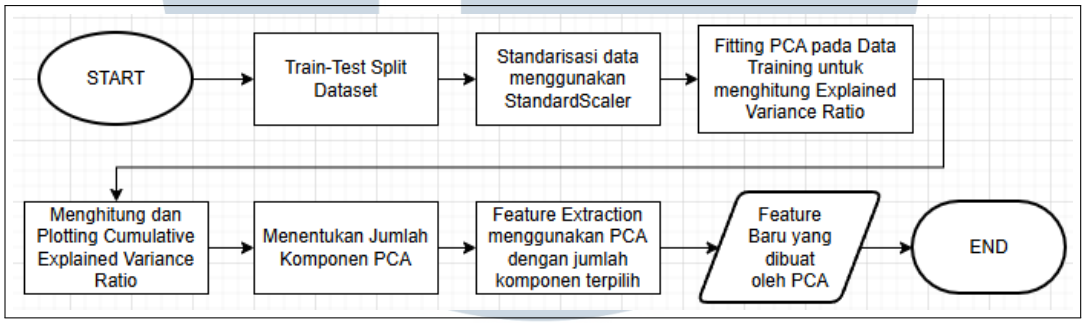
Gambar 3.4. *Flowchart* Alur *Feature Selection* PSO

PSO akan digunakan untuk melakukan proses *feature selection* dengan memilih subset fitur terbaik agar dapat meningkatkan kinerja algoritma RF. *Feature Selection* diawali dengan pembagian data *train-test split* dimana *dataset* terbagi menjadi 70% *training set* dan 30% *testing set*. Pemilihan rasio pembagian data ini dipilih dikarenakan terdapat penelitian terdahulu yang mendapatkan performa tertinggi ketika menggunakan rasio 70% *training set* / 30% *testing set* pada model RF [24]. Kemudian melakukan Inisialisasi *Fitness Function* yang akan digunakan sebagai evaluasi kualitas partikel berdasarkan performa model. Pada *Fitness Function*, dilakukan konversi nilai partikel menjadi *binary mask* dengan nilai dibawah median untuk fitur yang tidak terpilih dan nilai diatas median untuk fitur yang terpilih. Setelah itu, melakukan *training* model awal RF untuk menghitung nilai *fitness* serta melakukan *cross validation* menggunakan *StratifiedKFold*. Lalu,

mendefinisikan *bounds* serta *dimensions* dan melakukan inisialisasi PSO. setelah inisialisasi PSO, melakukan optimasi PSO sebanyak 5 iterasi. Setiap partikel akan mengevaluasi posisi berdasarkan *fitness function* serta memperbarui posisi berdasarkan pBest dan gBest. Akan dilakukan penyimpanan posisi terbaik gBest dan *fitness* terbaik. Setelah optimasi selesai dilakukan, akan mengambil parameter terbaik dari PSO dan dikonversi menjadi *binary mask* dan akan digunakan *binary mask* tersebut untuk memilih *subset* kombinasi fitur yang sudah dioptimasi.

B *Feature Extraction* dengan PCA

Gambar 3.5 menunjukkan alur dari *Feature Extraction* menggunakan PCA

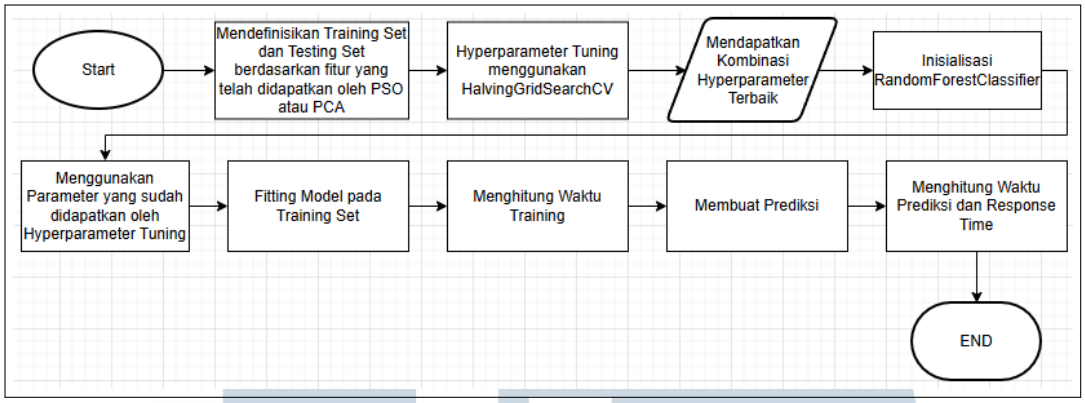


Gambar 3.5. *Flowchart* Alur *Feature Extraction* PCA

PCA akan digunakan untuk melakukan proses *feature extraction* yaitu dengan mengurangi dimensi dataset dan membuat fitur baru berupa*principal component*. Proses *feature extraction* diawali dengan *train-test split* yang memiliki pembagian *training set* dan *testing set* yang sama dengan PSO kemudian dilakukan standarisasi data agar semua fitur memiliki skala yang sama. Setelah itu, melakukan *fitting* pada data *training* menggunakan PCA untuk mereduksi dimensi data dan menghitung serta melakukan plotting *cumulative explained variance ratio* untuk menunjukkan seberapa banyak informasi yang dipertahankan seiring bertambahnya *principal component*. Setelah itu, menentukan jumlah komponen PCA berdasarkan *plotting* tersebut dan melakukan *feature extraction* dengan jumlah komponen PCA yang terpilih sehingga mendapatkan hasil berupa fitur baru berupa *principal component*.

C *Hyperparameter Tuning* dan *Training Model* dengan *Random Forest*

Gambar 3.6 menunjukkan alur dari *Hyperparameter Tuning* dan *Training Model* dengan *Random Forest*



Gambar 3.6. *Flowchart Hyperparameter Tuning* dan *Training Model* dengan *Random Forest*

Proses *Hyperparameter Tuning* diawali oleh mendefinisikan *training set* dan *testing set* yang sudah didapatkan oleh *feature selection* PSO atau *feature extraction* PCA lalu melakukan *tuning* menggunakan *HalvingGridSearchCV*.

Tabel 3.1 menunjukkan *Tuning Values* yang akan digunakan oleh *HalvingGridSearchCV* :

Tabel 3.1. *Tuning Values* yang akan digunakan oleh *HalvingGridSearchCV*

Hyperparameter	**Value / Range**
n_estimators	100, 150, 200, 250, 300
min_samples_split	40, 50, 60, 70, 80
min_samples_leaf	8, 10 , 12 , 14 , 16
criterion	'gini', 'entropy'

Setelah *tuning*, akan mendapatkan kombinasi hyperparameter terbaik yang akan digunakan oleh model *random forest*. Kemudian melakukan inisialisasi *Random Forest Classifier* menggunakan parameter yang sudah didapatkan dan melakukan *fitting* model pada *training set* serta melakukan perhitungan waktu *training*. Proses terakhir adalah membuat prediksi dan menghitung waktu prediksi dan *response time*.

3.5 Evaluasi

Teknik Evaluasi yang digunakan adalah *Classification Report* dan *Confusion Matrix*. *Classification Report* menunjukkan *precision, recall, accuracy* dan *f1 score* pada model yang sudah dilatih. *Precision* digunakan untuk mengukur ketepatan dari prediksi positif model, *Recall* digunakan untuk mengukur kemampuan model dalam menemukan seluruh sampel positif, *Accuracy* menghitung jumlah prediksi

yang benar dari total prediksi yang dilakukan dan yang terakhir *f1 score* adalah *harmonic mean* dari precision dan recall agar memberikan keseimbangan antara keduanya. Selanjutnya adalah *confusion matix*, *confusion matrix* digunakan untuk memberi gambaran mengenai model melakukan prediksi dibandingkan dengan label sebenarnya.

Berikut adalah contoh representasi dari *confusion matrix* :

$$\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix} \tag{3.1}$$

Dimana:

- TP (*True Positive*): total prediksi yang benar untuk *class positive*.
- FP (*False Positive*): total prediksi yang salah untuk *class positive*.
- FN (*False Negative*): total prediksi yang salah untuk *class negative*.
- TN (*True Negative*): total prediksi yang benar untuk *class negative*.

Kemudian jumlah prediksi tersebut dapat dihitung sesuai dengan metrik-metrik evaluasi pada *classification report* yaitu :

$$Precision = \frac{TP}{TP+FP} \tag{3.2}$$

$$Recall = \frac{TP}{TP+FN} \tag{3.3}$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.4}$$

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.5}$$