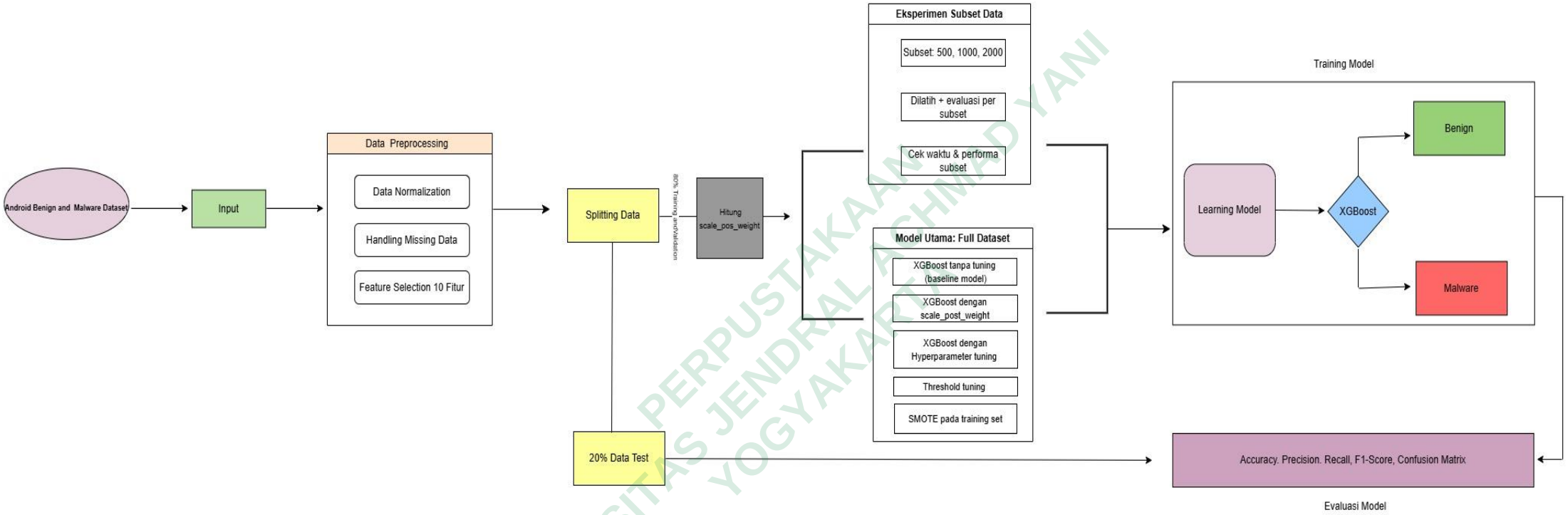
BAB 3
METODE PENELITIAN

Penelitian ini merupakan penelitian eksperimental yang bertujuan untuk menganalisis performa algoritma Extreme Gradient Boosting (XGBoost) dalam mendeteksi malware Android berbasis fitur *permissions*, dengan fokus utama pada penanganan ketidakseimbangan kelas (*imbalanced class*) menggunakan parameter *scale_pos_weight*. Model ini dipilih karena kemampuannya dalam mengolah dataset dengan jumlah fitur besar, menangani data yang tidak seimbang, serta mengoptimalkan hasil klasifikasi melalui mekanisme *boosting*.

Penelitian ini dilakukan dengan serangkaian tahap utama, mulai dari pengumpulan dataset, *preprocessing* data, pembagian dataset menjadi *training* dan *testing*, *tuning Hyperparameter* menggunakan *GridSearchCV* untuk menentukan bobot nilai *scale_post_weight*, pelatihan model XGBoost dengan parameter optimal, serta evaluasi performa model menggunakan metrik seperti *Accuracy, Precision, Recall, F1-Score*, dan *Confusion matrix.*

3.1 Tahapan Penelitian

Penelitian ini dilakukan melalui beberapa tahapan utama, mulai dari pemrosesan dataset hingga analisis hasil model machine learning. Setiap tahapan memiliki peran penting dalam memastikan bahwa model yang dibangun dapat mendeteksi malware dengan akurasi yang optimal. Adapun alur/tahapan penelitian ini dapat dilihat pada Gambar 3.1 Alur Metode Penelitian berikut.



Gambar 3.1 Alur Metode Penelitian

Berikut adalah penjelasan dari setiap alur/tahapan penelitian:

3.1.1 Dataset Penelitian

Dalam penelitian ini, dataset yang digunakan merupakan data sekunder yang diperoleh dari Mahindru, Arvind (2024), yang tersedia di Mendeley Data dengan DOI: 10.17632/rvjptkrc34.1. Dataset ini berjudul "Android Non-malware and Malware Dataset", dipublikasikan pada 6 Maret 2024 dan dapat diakses melalui situs Mendeley Data. Dataset ini berisi data *permissions* yang diekstrak dari berbagai file .apk yang diunduh dari tiga puluh *repository* berbeda. Dataset ini mencakup lebih dari 1.500 *permissions*, dengan aplikasi yang berasal dari tiga puluh kategori berbeda.

Dataset yang digunakan dalam penelitian ini memiliki 121.190 baris data dan 1.430 kolom fitur, yang merepresentasikan berbagai *permissions* yang diminta oleh aplikasi Android. Label dalam dataset terdapat pada kolom "OUTPUT ", yang menunjukkan apakah suatu aplikasi dikategorikan sebagai malware (1) atau non-malware (0) berdasarkan pola *permissions* yang diminta.

Dalam penelitian ini, istilah non-malware diganti dengan non-malware untuk menghindari ambiguitas dalam interpretasi data. Non-malware dalam konteks dataset ini mengacu pada aplikasi yang tidak memiliki indikasi aktivitas berbahaya, bukan malware yang telah mengalami mutasi atau menjadi jinak. Non-malware tidak memiliki fitur berbahaya yang diwarisi dari malware, sehingga berbeda dari aplikasi yang telah terinfeksi atau dikategorikan sebagai ancaman keamanan. Penelitian ini menggunakan istilah malware untuk aplikasi yang memiliki perilaku berbahaya, dan non-malware untuk aplikasi yang aman tanpa indikasi aktivitas mencurigakan.

Dataset ini hanya mencatat *permissions* yang diminta oleh aplikasi, bukan *permissions* yang disetujui atau ditolak oleh pengguna. Artinya, data ini tidak menunjukkan apakah pengguna memberikan izin atau tidak. Dalam beberapa kasus, meskipun pengguna tidak mengizinkan *permissions* tertentu, malware masih dapat mengeksploitasi sistem menggunakan teknik *privilege escalation* atau *social engineering* untuk mendapatkan akses yang tidak sah.

3.1.2 Preprocessing Data

Tahap pemrosesan data dilakukan untuk memastikan bahwa dataset siap digunakan dalam model machine learning. Tahap *preprocessing* data dilakukan untuk memastikan bahwa dataset dalam kondisi bersih dan dapat digunakan dalam pelatihan model machine learning. Langkah-langkah *preprocessing* yang dilakukan dirangkum dalam Tabel 3.1.

Tabel 3.1 Tahapan Preprocessing Data

No	Tahapan	Deskripsi
1.	Penghapusan Kolom Non-Relevan	Menghapus kolom 'Package' dan 'Category' karena keduanya merupakan data teks yang tidak relevan terhadap klasifikasi malware.
2.	Penanganan Missing Values dan Normalisasi	Konversi kolom non-numerik ke numerik (pd.to_numeric) dan hapus baris yang mengandung nilai NaN (dropna).
3.	Memastikan Semua Fitur Berupa Angka	Memastikan seluruh kolom berupa angka (0 atau 1). Kolom yang gagal dikonversi dihapus.
4.	Memisahkan Fitur dan Label	Data dipisah menjadi fitur (X) dan label (y) untuk proses pelatihan model.
5.	Pemilihan Fitur *Permissions* Terbaik	Memilih 10 fitur *permission* paling relevan berdasarkan frekuensi dan keterkaitan terhadap label malware.

3.1.3 Data Splitting

Tahap ini bertujuan untuk membagi dataset menjadi dua bagian utama agar model dapat dilatih dan diuji dengan data yang berbeda. Pembagian dilakukan dengan teknik *splitting* untuk memastikan distribusi kelas malware dan non-malware tetap seimbang di kedua subset data. Pembagian 80:20 dipilih karena merupakan standar umum dalam machine learning yang memberikan keseimbangan antara jumlah data pelatihan yang cukup untuk membangun model, serta data uji yang cukup untuk mengevaluasi performanya.

1) *Training Data* (80%) → *Training set* merupakan bagian terbesar dari dataset, yang digunakan untuk melatih model machine learning. Dalam tahap ini, model akan mempelajari pola dan karakteristik yang membedakan

aplikasi malware dan non-malware berdasarkan fitur *permissions* yang dimiliki. Dengan data pelatihan yang cukup, model diharapkan mampu memahami perbedaan pola akses *permissions* yang digunakan oleh aplikasi berbahaya dan aman.

2) *Testing Data* (20%) → *Testing set* digunakan untuk menguji seberapa baik model dapat mengklasifikasikan aplikasi sebagai malware atau non-malware pada data yang belum pernah dilihat sebelumnya. Data ini tidak digunakan dalam proses pelatihan, sehingga hasil pengujian mencerminkan kemampuan model dalam menggeneralisasi pola yang telah dipelajari. Dengan menggunakan *testing set*, performa model dapat dievaluasi berdasarkan metrik seperti *accuracy, Precision, Recall, specificity,* dan *F1-Score.*

3.1.4 Hyperparameter Tuning

Setelah pembagian dataset, tahap berikutnya adalah *tuning Hyperparameter* untuk meningkatkan performa model dalam menangani ketidakseimbangan kelas. Dalam penelitian ini, *tuning* dilakukan hanya pada *Scale_pos_weight* menggunakan *GridSearchCV*.

GridSearchCV digunakan untuk mencari nilai optimal dari *Scale_pos_weight* agar model dapat lebih baik dalam mengenali malware tanpa mengorbankan akurasi kelas non-malware. *Scale_pos_weight* bekerja dengan memberikan bobot lebih besar pada kelas malware, sehingga model lebih sensitif terhadap deteksi aplikasi berbahaya tanpa mengabaikan kelas lainnya.

3.1.5 Pelatihan Model XGBoost

Pada tahap ini, model Extreme Gradient Boosting (XGBoost) dilatih untuk mendeteksi malware Android berdasarkan fitur *permissions* yang diminta oleh aplikasi. XGBoost dipilih karena memiliki kemampuan yang tinggi dalam menangani dataset yang besar, serta mampu mengatasi ketidakseimbangan kelas (*imbalanced class*) yang sering terjadi dalam data deteksi malware.

Sebelum pelatihan model, dilakukan *tuning Hyperparameter* menggunakan *GridSearchCV*. Proses *tuning* ini bertujuan untuk menentukan nilai terbaik dari *scale_pos_weight*, yang akan membantu model dalam mengenali malware dengan lebih baik tanpa mengabaikan kelas non-malware. Dengan adanya *scale_pos_weight*, model dapat memberikan perhatian lebih pada data malware yang jumlahnya lebih sedikit dibandingkan dengan data non-malware.

Setelah mendapatkan nilai *Scale_pos_weight* yang optimal, model XGBoost dilatih menggunakan 80% data *training*. Pada tahap ini, model akan belajar mengenali pola *permissions* yang sering muncul dalam aplikasi malware dan membedakannya dengan pola *permissions* pada aplikasi non-malware. XGBoost bekerja dengan membangun sekumpulan pohon keputusan yang digunakan untuk menentukan apakah suatu aplikasi tergolong malware atau non-malware berdasarkan *permissions* yang dimilikinya.

Setelah proses pelatihan selesai, model diuji menggunakan 20% *data testing*, yaitu data yang tidak digunakan dalam pelatihan. Pengujian ini bertujuan untuk melihat sejauh mana model dapat mengklasifikasikan aplikasi dengan benar, sebelum akhirnya dilakukan evaluasi menggunakan berbagai metrik untuk menilai performanya.

3.1.6 Evaluasi Model

Evaluasi model dilakukan untuk mengukur seberapa baik XGBoost dalam mengklasifikasikan aplikasi Android sebagai malware atau non-malware berdasarkan fitur *permissions*. Evaluasi ini bertujuan untuk memastikan bahwa model mampu mendeteksi ancaman dengan tingkat akurasi yang tinggi, serta meminimalkan kesalahan dalam identifikasi malware.

Beberapa metrik evaluasi yang digunakan dalam penelitian ini adalah:

a) Akurasi (*Accuracy*)

Akurasi mengukur persentase prediksi yang benar dari keseluruhan data uji. Meskipun akurasi dapat memberikan gambaran umum mengenai kinerja model, namun akurasi saja tidak cukup dalam kasus data yang tidak seimbang. Model bisa saja mendapatkan akurasi tinggi, tetapi

tetap gagal dalam mendeteksi malware jika lebih banyak mengklasifikasikan data sebagai non-malware. Oleh karena itu, akurasi tetap diperhitungkan tetapi tidak menjadi satu-satunya indikator utama keberhasilan model.

b) Presisi (*Precision*)

Precision digunakan untuk melihat seberapa banyak aplikasi yang diklasifikasikan sebagai malware benar-benar merupakan malware. *Precision* tinggi menunjukkan bahwa model tidak terlalu banyak salah dalam menganggap aplikasi aman sebagai malware (*False Positive* rendah). Ini penting karena jika *Precision* terlalu rendah, pengguna dapat mengalami banyak aplikasi aman yang dianggap berbahaya secara keliru.

c) *Recall* (*Sensitivity*)

Recall menjadi metrik utama dalam penelitian ini. *Recall* mengukur kemampuan model dalam menemukan semua aplikasi malware yang ada di dalam dataset, karena fokus utama adalah memastikan bahwa sebanyak mungkin aplikasi malware dapat terdeteksi tanpa ada yang terlewat. Jika *Recall* tinggi, berarti model mampu mengenali malware dengan baik, sehingga mengurangi risiko aplikasi berbahaya lolos sebagai non malware.

Recall berperan dalam menyeimbangkan model (*balancing*), terutama dataset ini memiliki jumlah non-malware yang jauh lebih banyak dibandingkan malware. Dengan meningkatkan *Recall*, model menjadi lebih responsif terhadap kelas malware, sehingga distribusi klasifikasi antara kedua kelas lebih seimbang.

d) *F1-Score*

F1-Score adalah gabungan antara *Precision* dan *Recall,* yang memberikan gambaran keseimbangan model dalam mendeteksi malware dengan benar, tanpa terlalu banyak salah mengklasifikasikan aplikasi aman sebagai malware dan menjadi *trade-off* diantara kedua metrik tersebut.

e) *Confusion matrix*

Confusion matrix digunakan untuk melihat jumlah prediksi yang benar dan salah dalam masing-masing kelas (malware dan non-malware). Dari *Confusion matrix*, dapat dilihat apakah model lebih cenderung melakukan kesalahan dalam mendeteksi malware sebagai non-malware (*False Negative*) atau sebaliknya mendeteksi aplikasi aman sebagai malware (*False Positive*).

3.1.7 Analisis Hasil

Setelah model XGBoost diuji menggunakan *data testing*, dilakukan analisis terhadap performa hasil prediksi untuk mengamati bagaimana model merespons ketidakseimbangan kelas dalam deteksi malware Android berbasis fitur *permissions*. Penilaian dilakukan dengan menggunakan metrik evaluasi klasifikasi, yaitu akurasi, *precision*, *recall*, *F1-Score*, dan *confusion matrix*.

Akurasi digunakan untuk memperoleh gambaran umum seberapa besar proporsi prediksi yang benar dari keseluruhan data uji. Dalam konteks data yang tidak seimbang seperti pada penelitian ini, akurasi bukanlah indikator utama, karena model dapat saja memperoleh nilai akurasi tinggi meskipun gagal mendeteksi sebagian besar malware. Oleh sebab itu, fokus utama analisis diarahkan pada metrik *recall*, *precision*, dan *F1-Score*, yang lebih representatif dalam mengukur keberhasilan model terhadap kelas minoritas (malware).

Metrik *recall* menunjukkan kemampuan model dalam menemukan sebanyak mungkin aplikasi yang benar-benar tergolong malware. *Recall* tinggi menunjukkan bahwa model berhasil mengenali lebih banyak kasus malware, dan hal ini menjadi indikator penting dalam menilai keberhasilan penyesuaian parameter seperti *scale_pos_weight* maupun *threshold tuning*. *Precision*, di sisi lain, menggambarkan proporsi prediksi malware yang benar-benar malware. Jika *precision* terlalu rendah, akan terjadi banyak

False Positive, yaitu aplikasi non-malware yang salah diklasifikasikan sebagai ancaman.

Metrik *F1-Score* digunakan untuk menangkap keseimbangan antara *precision* dan *recall*, terutama ketika salah satu metrik mengalami *trade-off* terhadap lainnya. Nilai *F1-Score* yang tinggi menandakan bahwa model tidak hanya sensitif terhadap malware, tetapi juga tidak terlalu banyak melakukan kesalahan dalam klasifikasi.

Untuk mendalami sebaran klasifikasi, digunakan *confusion matrix*, yang menampilkan jumlah *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Nilai TP yang tinggi menunjukkan bahwa malware berhasil dikenali dengan baik, sedangkan nilai FN yang rendah menandakan bahwa hanya sedikit malware yang tidak terdeteksi. Sebaliknya, FP menunjukkan banyaknya aplikasi aman yang salah dikategorikan sebagai malware, dan TN merepresentasikan aplikasi non-malware yang diklasifikasikan dengan benar.

Secara keseluruhan, peningkatan nilai *recall* dan penurunan *False Negative* merupakan indikator bahwa model semakin mampu mengenali pola *permissions* yang mengindikasikan malware, terutama setelah dilakukan penyesuaian bobot kelas melalui *scale_pos_weight*. *Precision* tetap diperhatikan untuk memastikan bahwa kenaikan *recall* tidak menyebabkan model menjadi terlalu agresif dalam mendeteksi malware. *F1-Score* yang meningkat menunjukkan bahwa keseimbangan antara *recall* dan *precision* tetap terjaga, dan tidak hanya fokus pada salah satu metrik saja.

Sebagai visualisasi pendukung, *confusion matrix* divisualisasikan untuk membandingkan hasil klasifikasi sebelum dan sesudah penyesuaian bobot kelas. *Precision-recall curve* digunakan untuk memperlihatkan *trade-off* antara dua metrik tersebut dalam bentuk grafik, yang memberikan gambaran lebih menyeluruh mengenai sensitivitas model terhadap kelas malware.

3.1.8 Eksperimen Subset Data

Tahap ini dilakukan untuk menganalisis pengaruh jumlah Penggunaan subset, sebagaimana dirangkum dalam Tabel 3.2, memungkinkan pengamatan terhadap kestabilan dan efisiensi model dalam kondisi data yang terbatas, serta memberikan gambaran awal tentang kapasitas minimum yang dibutuhkan untuk menghasilkan klasifikasi yang representatif. data terhadap performa model dan kompleksitas waktu pelatihan.

Tabel 3.2 Rangkaian Eksperimen pada Subset Data

No	Tahapan	Deskripsi
1.	Pengambilan Subset	Diambil tiga subset dari dataset utama dengan ukuran 500, 1000, dan 2000 sampel. Pengambilan dilakukan secara stratified sampling untuk menjaga proporsi kelas malware dan non-malware tetap seimbang di masing-masing subset.
2.	Pelatihan dan Evaluasi	Masing-masing subset dibagi menjadi data latih dan uji (80:20), lalu dihitung ulang nilai *scale_pos_weight* berdasarkan distribusi kelas pada data latih. Model XGBoost dilatih ulang menggunakan parameter tuning sebelumnya, lalu dievaluasi menggunakan akurasi, *precision*, *recall*, dan *F1-Score*.
3.	Analisis Waktu Komputasi	Dicatat waktu pelatihan model pada tiap subset untuk menilai efisiensi proses komputasi. Analisis dilakukan untuk mengevaluasi dampak pertambahan ukuran data terhadap performa dan waktu yang dibutuhkan dalam pelatihan dan prediksi.

3.1.9 Threshold Tuning

Threshold tuning dilakukan untuk menguji pengaruh perubahan ambang klasifikasi terhadap keseimbangan performa model, terutama dalam meningkatkan sensitivitas terhadap kelas minoritas. Pendekatan ini

relevan dalam konteks klasifikasi biner yang *imbalanced*, di mana ambang *default* (0.5) sering kali menyebabkan bias terhadap kelas mayoritas. Tahapan yang dilakukan dalam proses *threshold tuning* dirangkum dalam Tabel 3.3 untuk memberikan gambaran sistematis mengenai pendekatan evaluasi terhadap perubahan ambang klasifikasi.

Tabel 3.3 Tahapan Evaluasi *Threshold* Tuning pada Model XGBoost

No	Tahapan	Deskripsi
1.	Pengujian Ambang Batas	Model terbaik hasil tuning diuji dengan nilai *threshold* : 0.3, 0.4, 0.5, 0.6, dan 0.7. Probabilitas keluaran (predict_proba) digunakan sebagai dasar klasifikasi, dengan ambang tertentu untuk menentukan kelas malware.
2.	Evaluasi Setiap *Threshold*	Setiap nilai *threshold* dievaluasi dengan metrik akurasi, *precision*, *recall*, dan *F1-Score*. Evaluasi ini dilakukan untuk memahami *trade-off* antara *False Positive* dan *False Negative*, serta mencari keseimbangan performa model.

3.1.10 Penerapan SMOTE

SMOTE (*Synthetic Minority Over-sampling Technique*) diterapkan sebagai salah satu strategi penyeimbangan data untuk mengatasi dominasi kelas mayoritas. Teknik ini menciptakan sampel sintetis untuk kelas minoritas berdasarkan kemiripan fitur, sehingga memungkinkan model belajar dari distribusi yang lebih seimbang. Rangkaian proses yang dilakukan dalam penerapan SMOTE ditampilkan secara ringkas dalam Tabel 3.4 untuk menggambarkan alur eksperimen penyeimbangan kelas melalui teknik *oversampling*.

Tabel 3.4 Tahapan Penerapan SMOTE dalam Pelatihan Model

No	Tahapan	Deskripsi
1.	*Oversampling* pada Data Training	SMOTE diterapkan hanya pada data latih untuk menjaga keaslian distribusi pada data uji. Teknik ini menghasilkan sampel sintetis untuk kelas minoritas (malware) berdasarkan kemiripan fitur guna menyeimbangkan distribusi kelas.
2.	Pelatihan Ulang Model	Model XGBoost dilatih ulang menggunakan data latih hasil SMOTE dengan konfigurasi parameter yang sama seperti sebelumnya, agar proses evaluasi tetap konsisten dan adil.
3.	Evaluasi Performansi Pasca-SMOTE	Evaluasi dilakukan menggunakan metrik akurasi, *precision*, *recall*, dan *F1-Score*. Hasil dibandingkan dengan model sebelum SMOTE untuk menilai peningkatan performa, khususnya pada *recall* dan F1 tanpa penurunan *precision* yang besar.

3.2 Bahan Penelitian

Bahan penelitian merupakan komponen utama yang digunakan dalam penelitian ini, termasuk dataset dan perangkat lunak yang mendukung pengolahan data. Dalam penelitian ini, bahan yang digunakan meliputi dataset yang berisi informasi mengenai *permissions* aplikasi Android serta perangkat lunak yang digunakan untuk mengolah dan menganalisis data tersebut.

1) Dataset Penelitian

Dataset yang digunakan dalam penelitian ini adalah "*Android Benign and Malware Dataset*", yang diperoleh dari Mahindru, Arvind (2024) dan tersedia di Mendeley Data dengan DOI: 10.17632/rvjptkrc34.1. Dataset ini berisi data *permissions* yang diekstrak dari berbagai file .apk yang diunduh dari tiga puluh repository dan mencakup lebih dari 1.500 *permissions*. Dataset ini terdiri dari 121.190 baris data dan 1.430 kolom fitur, yang

digunakan untuk mengidentifikasi aplikasi sebagai non-malware (0) atau malware (1) berdasarkan *permissions* yang digunakan.

2) Perangkat Lunak dan Perangkat Keras

Alat penelitian terdiri dari perangkat keras dan perangkat lunak yang digunakan untuk pemrosesan data, pelatihan model, serta evaluasi performa algoritma XGBoost dalam mendeteksi malware pada sistem Android.

3.3 Alat Penelitian

Alat yang digunakan dalam penelitian ini mencakup perangkat keras dan perangkat lunak yang mendukung proses eksperimen serta pemrosesan data. Spesifikasi perangkat keras merujuk pada komputer yang digunakan selama pelaksanaan pelatihan dan pengujian model, sedangkan perangkat lunak terdiri atas sistem operasi, platform analisis data, serta pustaka (*library*) pemrograman yang digunakan untuk membangun dan mengevaluasi model. Rincian komponen perangkat keras dan perangkat lunak yang digunakan dalam penelitian ini disajikan pada Tabel 3.5.

Tabel 3.5 Alat Penelitian

Alat Penelitian		
No	**Perangkat Keras**	**Perangkat Lunak**
1.	Laptop	Sistem Operasi: Windows 10 Pro
2.	Processor Intel® Core™ i5-4300U @ 1.90GHz (2.49 GHz)	Google Colab
3.	RAM 8 GB	Bahasa Pemrograman: Python
4.	Arsitektur Sistem 64-bit OS, x64-based processor	Library: Scikit-learn, XGBoost, Pandas, NumPy, Matplotlib, Seaborn
5.	Koneksi Internet	

3.4 Jalan Penelitian

Penelitian ini dilakukan secara eksperimental dan sistematis melalui serangkaian tahapan yang dirancang untuk mengevaluasi performa algoritma XGBoost dalam mendeteksi malware Android berbasis fitur *permissions*, dengan fokus utama pada penanganan ketidakseimbangan kelas. Alur pelaksanaan

penelitian ini divisualisasikan dalam Tabel 3.6 untuk mempermudah pemahaman terhadap tahapan eksperimen yang dilakukan.

Tabel 3.6 Urutan Jalan Penelitian

No	Tahapan	Uraian Singkat
1.	Persiapan Penelitian	Studi literatur, pengumpulan referensi, dan pemilihan dataset penelitian.
2.	Pengumpulan Data	Dataset "Android Benign and Malware Dataset" diunduh dan dianalisis strukturnya.
3.	Pengolahan Data	Preprocessing: penghapusan kolom non-relevan, konversi data ke numerik, penanganan NaN. Fitur dan label dipisahkan.
4.	Pemilihan Fitur	Seleksi 10 fitur *permissions* terbaik dari dataset untuk efisiensi pemodelan.
5.	Implementasi Model	Pelatihan awal dilakukan menggunakan XGBoost dengan parameter default sebagai baseline model. Selanjutnya, dilakukan pelatihan ulang dengan penyesuaian parameter scale_pos_weight untuk mengatasi ketidakseimbangan kelas. Setelah itu, model dioptimalkan melalui hyperparameter tuning menggunakan GridSearchCV untuk memperoleh kombinasi parameter terbaik.
6.	Evaluasi Model	Evaluasi menggunakan Accuracy, *Precision*, *Recall*, *F1-Score*, dan Confusion Matrix. Fokus pada *recall* & *F1-Score*.
7.	Eksperimen Subset Data	Dilakukan pada subset data berukuran 500, 1000, dan 2000 untuk menganalisis pengaruh ukuran data.
8.	*Threshold* Tuning	Nilai ambang prediksi diuji (0.3 – 0.7) untuk melihat pengaruh terhadap metrik performa.
9.	Penerapan SMOTE	SMOTE digunakan untuk menyeimbangkan data latih. Model dilatih ulang dan dibandingkan.
10.	Penyusunan Laporan	Dokumentasi seluruh proses ke dalam laporan tugas akhir secara sistematis.