Penerapan Model Ensemble Stacking untuk Deteksi Malware dengan Optimasi Hyperparameter pada Support Vector Machine, Random Forest, dan XGBoost

Jonathan Fanosara Telaumbanua
Fakultas Informatika
Telkom University
Bandung, Indonesia
jonathantelaumbanua@student.telkomuniversity.ac.id

Aji Gautama Putrada
Fakultas Informatika
Telkom University
Bandung, Indonesia
ajigps@telkomuniversity.ac.id

Ryan Lingga Wicaksono
Fakultas Informatika
Telkom University
Bandung, Indonesia
wicaksonoryan@telkomuniversity.ac.id

Peningkatan jumlah pengguna Android menjadikannya sasaran utama dalam penyebaran malware Android, sehingga dibutuhkan model deteksi malware yang akurat dan efisien. Penelitian ini bertujuan untuk membangun dan menganalisis model *ensemble stacking* yang menggunakan tiga algoritma pembelajaran mesin, yaitu *Support Vector Machine* (SVM), *Random Forest*, dan *XGBoost*, yang dioptimasi menggunakan *hyperparameter tuning GridSearchCV*. *Ensemble stacking* digunakan untuk meningkatkan performa deteksi dengan menggabungkan keunggulan masing-masing *base learner* agar menghasilkan model yang lebih kuat. Dataset yang digunakan adalah *NaticusDataset*, yaitu dataset malware Android berbasis fitur *permissions* yang diminta oleh aplikasi. Hasil pengujian menunjukkan bahwa model SVM menghasilkan *accuracy* sebesar 0.9651, *precision* 0.9585, *recall* 0.9714, dan *F1-score* 0.9649. Model Random Forest memperoleh *accuracy* 0.9700, *precision* 0.9756, *recall* 0.9643, dan *F1-score* 0.9699. Model XGBoost mencapai *accuracy* 0.9695, *precision* 0.9765, *recall* 0.9622, dan *F1-score* 0.9693. Model *ensemble stacking* memberikan peningkatan performa dengan *accuracy* 0.9702, *precision* 0.9769, *recall* 0.9633, dan *F1-score* 0.9700. Analisis fitur menggunakan *feature importance* dan SHAP menunjukkan kontribusi signifikan dari fitur *permissions* terhadap hasil deteksi malware.

Kata kunci— malware android, ensemble stacking, support vector machine, random forest, xgboost.

I. PENDAHULUAN

Di era digital saat ini, penggunaan perangkat Android semakin meningkat seiring dengan pesatnya perkembangan teknologi digital saat ini. Banyaknya aplikasi yang mampu membantu kegiatan manusia saat ini membuat perangkat seluler Android menjadi sangat populer. Namun, popularitas yang tinggi ini juga menjadikan Android sebagai sasaran utama dalam penyebaran malware. Malware merupakan *software* berbahaya yang dibuat dengan tujuan merusak sistem perangkat atau mengambil alih perangkat tanpa diketahui oleh pengguna. Ancaman ini tidak hanya membahayakan data pribadi pengguna, tetapi juga dapat menyebabkan kerugian finansial dan terganggunya operasional yang dilakukan pengguna maupun organisasi [1].

Dengan meningkatnya penyebaran malware pada Android ini, diperlukannya sistem deteksi malware yang lebih canggih dan efisien untuk perangkat Android. Sistem yang dikembangkan ini harus mampu mengidentifikasi malware dengan tingkat efisiensi serta akurasi yang tinggi. Pada penelitian ini, pendekatan sistem berbasis pembelajaran mesin menjadi sangat penting karena metode ini mampu mempelajari data dan mengenali pola-pola yang tidak mudah terdeteksi secara langsung [1].

Beberapa penelitian sebelumnya telah menggunakan teknik pembelajaran mesin secara terpisah untuk mendeteksi malware Android. Seperti penelitian oleh Akbar et al. [2] menunjukkan bahwa Random Forest, Support Vector Machine, Naive Bayes, serta Rotation Forest memiliki kinerja yang tinggi yaitu di angka 89%. Penelitian menurut J. M. Arif et al [3] menunjukkan kinerja model Random Forest, Multilayer Percepton, KNN, Decision Tree, dan Adaboost memberikan hasil nilai True Positive Rate (TPR) tertinggi pada Random Forest yaitu 91,6% dalam mendeteksi malware Android.

Meskipun penelitian sebelumnya menunjukkan hasil akurasi deteksi malware yang tinggi, hal ini masih bisa ditingkatkan dengan menggunakan *ensemble learning*. Dalam *ensemble learning* beberapa model pembelajaran mesin berbeda untuk menghasilkan prediksi yang lebih akurat. Teknik juga berguna untuk menutupi kekurangan masing-masing model, mengatasi ketidakseimbangan data, dan meningkatkan kinerja sistem dalam mendeteksi malware [4].

Dalam penelitian ini, pendekatan *ensemble stacking classifier* digunakan untuk menggabungkan model Support Vector Machine (SVM), Random Forest, dan XGBoost dengan tujuan meningkatkan tingkat akurasi model. Selain itu, penelitian ini juga akan mengoptimalkan parameter model dengan *hyperparameter tuning* untuk masing-masing model untuk mencapai performa model dengan tingkat akurasi tertinggi. Evaluasi kinerja model dalam penelitian ini akan menggunakan metrik evaluasi yang umum digunakan, yaitu metrik *accuracy*, *precision*, *recall*, dan *F1-score* untuk

menghitung efektivitas model dalam mendeteksi malware pada perangkat Android.

Dengan menggunakan *ensemble stacking*, penelitian ini diharapkan mampu menghasilkan sistem deteksi malware pada Android dengan akurasi serta efisiensi yang lebih tinggi dibandingkan dengan pendekatan yang hanya mengandalkan satu model pembelajaran mesin secara individual.

II. KAJIAN TEORI

A. Studi Terkait

Pada penelitian Zhang J. et al [5], melakukan riset deteksi malware dengan *feature fussions* dan *stacking ensemble*. Zhang menggunakan K-Nearest Neighbour, Logistic Regression, Support Vector Machine, dan Random Forest sebagai *base classifier.* Setelah model *base classifier* dibangun, keempat model digabungkan dengan metode *stacking*. Setelah model *stacking* dibentuk, model ini di *improve* yang mana model *improve stacking* ini menghasilkan nilai metrik *accuracy* 98.16%, metrik *precision* 98,69%, metrik *recall* 97,73%, dan metrik *F1-score* 98,21% diatas model *base* dan *stacking* yang tidak di *improve*.

Penelitian menurut Li T. et al [6], melakukan penelitian dengan menggabungkan KNN, Random Forest, dan XGBoost untuk membentuk model *Attention Mechanism* (AM) *Stacking*. Model ini dibentuk untuk mengatasi keterbatasan dari kurang stabilnya performa dan buruknya generalisasi dari model individual, mengurangi redundansi fitur, dan meningkatkan akurasi performa model. Hasil eksperimennya menunjukkan peningkatan sebesar 1,38% dibandingan dengan metode deteksi malware lainnya. Selain itu, peningkatan performa dalam *precision* dan *recall* sebesar 1,6% dan 1,67%.

Penelitian Joshi A. dan Kumar S. [7], melakukan penelitian menggunakan 2 dataset yaitu MalDroid2020 dan MalMem2022. Joshi A. dan Kumar S. membentuk model *ensemble learning* menggunakan *Voting Classifier* dan *Stacking*. Model *Stacking* dibangun dengan 4 *base learner* yaitu HGBT, SVM, CatBoost, Random Forest dan *Voting Classifier* sebagai *final estimator*-nya. Penelitian ini menghasilkan akurasi yang sangat tinggi yaitu 98% pada dataset MalDroid2020 dan 99% pada dataset MalMem2022.

Pada penelitian Sumalatha P dan Mahalakshmi G [8], melakukan percobaan dengan model *stacking ensemble* pada tiga dataset yang berbeda. Percobaan pertama menunjukkan model *stacking ensemble* memberikan akurasi *AUC* 86,89%, percobaan kedua 96,72%, dan percobaan ketiga 89,86%. Hasil tertinggi 96,72% menunjukkan hasil yang performa model yang lebih tinggi dibandingkan dengan model base pembelajaran mesin.

Penelitan menurut Wang et al [9], membangun model MFDroid yang merupakan model *ensemble learning* yang terdiri dari 5 model *base learner*, yaitu SVM, GDBT, XGBoost, LightGBM, dan CatBoost. Model MFDroid ini diuji dengan dataset MFDroid dan OmniDroid, model MFDroid menunjukkan *F1-score* sebesar 96% pada dataset MFDroid dan model MFDroid menggungguli model lain dalam fitur statis pada dataset OmniDroid.

B. Android dan Ancaman Malware

Android merupakan sistem operasi berbasis Linux yang dikembangkan oleh Google pada tahun 2005 dan meluncurkan *smartphone* pertamanya pada tahun 2008 [10]. Sistem operasi ini menawarkan berbagai aplikasi yang dapat memudahkan pengguna dalam beraktivitas sehari-hari, sehingga membuat Android menjadi sangat populer hingga saat ini. Dengan popularitas ini juga menjadikan Android menjadi sasaran utama dalam penyebaran malware.

Malware Android merupakan perangkat lunak berbahaya yang dirancang untuk merusak atau mengambil alih perangkat tanpa sepengetahuan pengguna. Malware Android telah menjadi ancaman bagi bidang keamanan siber sejak 2008. Peretas terus menerus mengembangkan berbagai metode untuk menghindari deteksi dan menimbulkan kerugian [10]. Hingga saat ini malware masih terus berkembang dan dibutuhkannya pendekatan baru yang lebih efektif dalam mengidentifikasi malware pada Android.

C. Support Vector Machine

Support Vector Machine adalah algoritma pembelajaran mesin yang digunakan untuk klasifikasi dan regresi. SVM berkerja dengan membangun *hyperplane* linier yang berfungsi sebagai batas keputusan antara dua kelas, yaitu malware dan benign. *Hyperplane* ini ditentukan berdasarkan proporsi sampel yang diberi label malware dan dibandingkan dengan sampel aplikasi yang diberi label benign. Jika persentase suatu aplikasi melebihi batas tertentu, maka SVM akan mengklasifikasikan aplikasi itu apakah malware atau benign [11]. Kekurangan dari model SVM ini adalah waktu komputasi yang lama karena model komputasi yang kompleks serta sensitif terhadap pemilihan kernel karena dapat mempengaruhi kinerja model.

D. Random Forest

Random Forest adalah metode pembelajaran mesin berbasis algortima *ensemble* yang menggabungkan sejumlah *decision tree* untuk membuat suatu prediksi. Setiap pohon keputusan dilatih dengan subset data latih acak serta fitur acak. Hal ini berguna untuk meminimalkan terjadinya *overfitting* yang umumnya terjadi pada penggunaan satu *decision tree* saja. Random Forest kemudian mengumpulkan hasil dari seluruh *decision tree* dan menentukan hasil berdasarkan suara terbanyak dari *decision tree*. Model ini memiliki keunggulan dalam kecepatan, kesederhanaan dan unggul dalam melakukan klasifikasi [12]. Kekurangan model Random Forest terdapat pada model ini bermasalah dengan fitur yang banyak (High Dimensionality) dan mudah bias dengan fitur-fitur dengan bobot rendah/kurang relevan.

E. Extreme Gradient Boosting (XGBoost)

Extreme Gradient Boosting (XGBoost) adalah algoritma *boosting* yang berguna untuk meningkatkan kinerja model klasifikasi dengan cara menggabungkan beberapa model. XGBoost bertujuan untuk menemukan pemisahan menggunakan algoritma yang efisien, dengan regulasi L1 dan L2 untuk mengatasi *overfitting* [13]. Keunggulan lainnya model ini yaitu memperkuat model, meningkatkan generalisasi model, dan model ini fleksibel dalam melakukan *cross validation* pada setiap iterasi *boosting* [12]. Model XGBoost ini memiliki kekurangan rentan terhadap *overfitting* serta keterbatasan dalam menangani data yang tidak distandarisasi.

F. Hyperparameter Tuning

Hyperparameter tuning adalah proses melatih model dengan parameter yang sudah disesuaikan untuk mencapai performa model yang lebih tinggi. Ada beberapa algoritma yang bisa digunakan untuk tuning hyperparameter suatu model, seperti Grid Search, Random Search, dan Bayesian Optimization. Pada penelitian ini, *hyperparameter tuning* berfokus pada menggunakan algoritma Grid Search. Algoritma Grid Search bekerja dengan menentukan rentang nilai setiap *hyperparameter* yang berguna untuk mengoptimalkan kinerja model. Algoritma ini akan mengeksplorasi dan mencari nilai dengan pengaruh peningkatan tertinggi dari seluruh ruang parameter model [14].

G. Ensemble Stacking

Ensemble Stacking adalah salah satu metode dalam *ensemble learning* yang mengombinasikan beberapa model pembelajaran mesin berbeda untuk menghasilkan prediksi yang lebih akurat. Pada *stacking*, model pembelajaran mesin *base learner* dilatih dengan data yang sama, lalu hasil prediksi dari masing-masing model dasar ini digunakan sebagai input untuk melatih model baru yang disebut *meta-learner*. *Meta-learner* ini kemudian menggabungkan prediksi dari *base learner* dan membuat keputusan akhir. Pada penelitian ini, model *stacking* digunakan untuk meningkatkan akurasi dan kemampuan generalisasi model untuk mendapatkan peningkatan performa model [6].

H. Feature Importance

Feature importance adalah metode yang digunakan untuk menganalisis seberapa besar pengaruh masing-masing fitur dalam menentukan keputusan model. Dalam penelitian ini, *feature importance* akan mendeteksi fitur *permissions* mana yang paling berpengaruh dalam mendeteksi malware Android. Model Random Forest dan XGBoost yang dibangun akan menghitung bagaimana fitur-fitur mempengaruhi kinerja model dalam membuat prediksi. Jika fitur sering dan efektif dalam membagi data, maka semakin tinggi nilai *feature importance* [12].

I. SHapley Additive ExPlanations (SHAP)

SHapley Additive ExPlanations (SHAP) adalah metode yang digunakan untuk mengukur kontribusi setiap fitur dan menjelaskan hasil prediksi yang dihasilkan oleh model. SHAP akan memberikan nilai kontribusi untuk setiap fitur yang dihitung berdasarkan nilai rata-rata kontribusi fitur pada prediksi di seluruh kemungkinan fitur. Ini memungkinkan SHAP untuk memberikan penjelasan hasil prediksi yang lebih transparan dan dampak tiap fitur untuk hasil prediksi model [15].

III. METODE

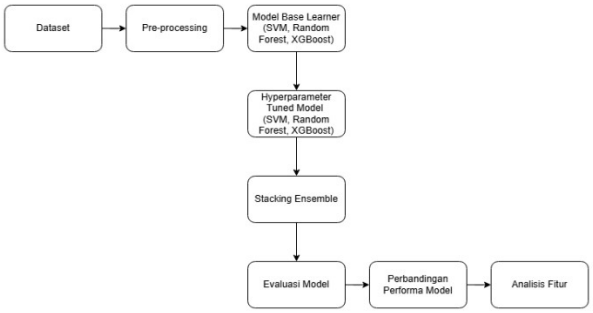
A. Desain Perancangan Sistem

Perancangan sistem pada penelitian ini bertujuan untuk menganalisis kinerja model *ensemble stacking* dalam mendeteksi malware berbahaya pada Android. Model *ensemble stacking* ini dibangun dengan 3 model *base learner*, yaitu: model Support Vector Machine (SVM), Random Forest, dan XGBoost. Masing-masing model dilatih untuk mengenali pola malware berdasarkan fitur-fitur yang diekstraksi dari aplikasi Android.

Sebelum ketiga model *base learner* dilatih, dilakukan *preprocessing* data untuk memastikan data tepat dan siap digunakan untuk melatih model. Setelah *preprocessing* dilakukan, ketiga model *base learner* dilatih dengan parameter yang dioptimalkan menggunakan teknik *hyperparameter tuning GridSearchCV*. Model ini akan dibandingkan dengan model *base learner* yang tidak dioptimasi dan diharapkan mendapatkan peningkatan performa yang lebih tinggi.

Setelah model *base learner* yang sudah dioptimasi dilatih, hasil prediksi dari tiap model digabungkan menggunakan *stacking ensemble* dan membentuk model *meta-learner* menggunakan Logistic Regression untuk menghasilkan prediksi akhir. *Ensemble stacking* akan memanfaatkan kekuatan masing-masing model *base learner* untuk meningkatkan akurasi dan kekuatan dalam mendeteksi malware Android.

Hasil prediksi akhir ini akan dievaluasi menggunakan metrik evaluasi *accuracy, precision, recall,* dan *F1-score* untuk mengukur kinerja model. Hasil dari kinerja model *stacking* dan model *base learner* yang sudah *tuning* akan dibandingkan, akan dianalisis model mana yang menghasilkan performa dengan akurasi yang lebih tinggi serta efektif dalam mendeteksi malware pada Android. Kemudian *feature importance* dan SHAP akan digunakan untuk menganalisis fitur mana yang paling berpengaruh dalam keputusan model dalam mendeteksi malware Android.



GAMBAR 1
Diagram Alir Penelitian

B. Preprocessing Data

Dataset yang digunakan adalah dataset Android Malware yang diperoleh dari platform *Kaggle*. Dataset ini berisi data fitur statis berupa izin akses yang diminta oleh aplikasi Android. Dataset ini terdiri dari 29.332 sampel aplikasi dengan 86 fitur permissions. Data untuk setiap aplikasi direpresentasikan dalam bentuk biner, yang dimana nilai 1 merepresentasikan fitur *permissions* diberikan dan nilai 0 merepresentasikan fitur *permissions* tidak diberikan.

Sebelum dataset ini digunakan untuk melatih model, dataset akan di *preprocessing* untuk memastikan kualitas dan konsistensi dataset. Ini bertujuan agar saat pelatihan model dapat berjalan model dan mendapatkan akurasi yang tinggi. Tahapan *preprocessing* meliputi:

1. Pengecekan nilai yang hilang dilakukan untuk memastikan bahwa tidak terdapat data yang berisi nilai null atau tidak valid.

2. Pengecekan dataset memiliki format yang sama, pada dataset yang digunakan dalam penelitian ini berformat biner

(0/1). Maka proses encoding akan dilakukan untuk melakukan pengecekkan.

3. Pembagian dataset, pada penelitian ini dataset dibagi menjadi 80% data latih dan 20% data uji. Pembagian dilakukan menggunakan parameter *stratify* untuk membuat proporsi antara data uji dan data latih tetap seimbang.

4. Proses penskalaan fitur untuk memastikan setiap fitur berada pada skala yang seragam dan menghindari dominasi fitur dengan nilai besar dilakukan *scalling* pada dataset. Pada penelitian ini *standardscaler* digunakan untuk menormalisasikan dataset, sehingga setiap fitur memiliki nilai *mean* = 0 dan *standard deviation* = 1.

C. Support Vector Machine

Support Vector Machine (SVM) adalah model yang digunakan untuk mengklasifikasikan data dengan membangun *hyperplane* yang memisahkan dua kelas, dalam konteks penelitian ini yaitu kelas malware dan benign. Dalam penelitian ini, SVM dioptimasi menggunakan kernel Linear dan kernel *Radial Basis Function* (RBF). Penggunaan kedua kernel ini bertujuan untuk menangani data-data yang tidak dapat dipisahkan secara linear. *Hyperparameter tuning* juga digunakan dalam penelitian ini dengan *GridSearchCV* untuk mencari kinerja yang optimal dari model SVM.

Parameter yang digunakan dalam *hyperparameter tuning* ini adalah parameter C, kernel, dan Gamma. *GridSearchCV* akan mencoba menggunakan beberapa nilai C (0.1, 1, 10) untuk mengontrol kesalahan dalam mengklasifikasi. Sedangkan nilai Gamma diatur *scale* dan *auto* untuk mengatur kerapatan dari fungsi kernel. Proses ini bertujuan untuk menghindari *overfitting* dan menemukan parameter dengan peningkatan tertinggi untuk menghasilkan akurasi tertinggi pada data latih. Hasil dengan peningkatan tertinggi dari *GridSearchCV* ini nantinya akan digunakan untuk melatih model SVM.

D. Random Forest

Model Random Forest adalah model *ensemble* yang menggunakan banyak *decision tree* untuk membuat prediksi akhir berdasarkan suara terbanyak dari 15 seluruh *decision tree*. Model ini mampu memberikan hasil prediksi yang akurat dalam menangani dataset yang besar dan kompleks. *Hyperparameter tuning* pada Random Forest dilakukan dengan *GridSearchCV* untuk mencari nilai yang optimal untuk parameter model.

Parameter yang digunakan dalam model Random Forest ini adalah *n_estimators* {100, 200} untuk mengatur jumlah pohon keputusan, *max_depth* {10, 20, None} untuk mengatur kedalaman pohon keputusan, *min_samples_split* {2, 5} untuk mengatur jumlah minimum sampel dalam membagi simpul pohon keputusan, dan *min_samples_leaf* {1, 2} untuk mengatur jumlah minum daun disetiap pohon keputusan.

GridSearchCV akan menemukan kombinasi parameter dengan peningkatan tertinggi untuk meningkatkan akurasi model dengan menguji nilai masing-masing parameter. Proses tuning ini berguna untuk mengurangi kemungkinan *overfitting*, mengoptimalkan model, dan meningkatkan performa model Random Forest dalam mendeteksi malware Android.

E. EXtreme Gradient Boosting (XGBoost)

XGBoost adalah algoritma yang membangun pohon keputusan secara sekuensial dan setiap pohon keputusan baru akan memperbaiki kesalahan dari pohon keputusan sebelumnya. XGBoost mampu menangani dataset yang besar dengan kecepatan dan performa yang tinggi. *GridSearchCV* digunakan juga pada model XGBoost untuk menemukan performa tertinggi dan menghindari *overfitting* pada model.

Hyperparameter tuning XGBoost dengan *GridSeachCV* akan mengatur parameter *n_estimators* {100, 200} beguna untuk mengatur jumlah pohon keputusan yang dibangun oleh model XGBoost, *learning_rate* {0.01, 0.1} untuk mengontrol kecepatan dan kestabilan model dalam mempelajari pola, *max_depth* {3, 6, 10} untuk mengatur kedalaman maksimal tiap pohon keputusan yang dibuat oleh model, dan *subsample* {0.8, 1.0} untuk mengontrol proporsi sampel yang digunakan dalam melatih setiap pohon keputusan.

F. Stacking Ensemble

Ensemble learning adalah pendekatan pembelajaran mesin yang mengombinasikan model-model pembelajaran mesin untuk mendapatkan hasil prediksi yang lebih efektif dan akurat daripada model pembelajaran mesin tunggal. Model ini dapat memanfaatkan kekuatan masing-masing model *base leaner* dan menghasilkan performa yang optimal dibandingkan dengan model individual. Dalam penelitian ini, teknik *Stacking Ensemble* digunakan untuk menggabungkan hasil dari 3 model *base learner*, yaitu Support Vector Machine (SVM), Random Forest, dan XGBoost.

Pada pembuatan model *Stacking Ensemble*, diawali dengan membuat dan melatih model *base learner* dengan dataset yang sama secara terpisah. Kemudian hasil prediksi tiap model *base learner* akan digunakan sebagai input pada algoritma Logistic Regression untuk model *meta-learner*. Dengan menggunakan model *stacking ensemble*, model akan mengurangi bias dan variansi prediksi sehingga akan meningkatkan akurasi dan kekuatan model.

G. Evaluasi Model

Setelah model dilatih dan diuji, evaluasi performa dilakukan untuk mengukur seberapa efektif model dalam mengklasifikasikan aplikasi Android sebagai benign atau malware berdasarkan fitur permissions. Evaluasi ini menggunakan beberapa metrik, yaitu metrik *accuracy*, *precision*, *recall* dan *F1-score*.

a. *Accuracy*: digunakan untuk mengukur proporsi prediksi yang benar dari seluruh sampel positif. Rumusnya sebagai berikut:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

b. *Precision*: untuk mengukur ketepatan prediksi kelas malicious. Rumusnya sebagai berikut:

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

c. *Recall*: metrik yang digunakan untuk mengukur kemampuan model dalam menemukan seluruh data malicious secara benar. Rumusnya sebagai berikut:

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

d. *F1-Score*: rata-rata dari precision dan recall. Rumusnya sebagai berikut:

$$F1 - score = 2\,x\,\frac{Presisi\ x\ Recall}{Presisi\ +\ Recall} \quad (4)$$

H. Feature Importance

Feature importance adalah metode yang digunakan untuk mengukur tingkat kontribusi masing-masing fitur terhadap keputusan yang dihasilkan model. Pada penelitian ini model Random Forest dan XGBoost digunakan untuk menghitung nilai kontribusi setiap fitur terhadap model.

Pada Random Forest, *feature importance* dihitung berdasarkan pengurangan *impurity* yang dihasilkan oleh pohon keputusan. Semakin besar kontribusi fitur dalam mengurangi nilai *impurity*, maka semakin tinggi nilai *feature importance* yang diberikan pada fitur tersebut. Sedangkan pada model XGBoost, *feature importance* dihitung berdasarkan kontribusi setiap fitur dalam mengurangi *loss function* pada tiap iterasi *boosting*. Fitur yang paling sering digunakan untuk mengurangi *loss function* dan membagi data akan diberikan nilai *importance* yang lebih tinggi.

I. SHapley Additive ExPlanations (SHAP)

SHapley Additive Explanation (SHAP) adalah metode yang digunakan untuk mengukur kontribusi setiap fitur terhadap hasil prediksi model. SHAP diterapkan pada model XGBoost untuk menganalisis kontribusi setiap fitur. Fitur dengan nilai SHAP yang tinggi akan memberikan gambaran bagaimana pola setiap fitur aplikasi malware dan membantu model dalam membuat prediksi.

IV. HASIL DAN PEMBAHASAN

A. Hasil Model Base Learner

Model base learner yang belum dioptimalkan dengan hyperparameter tuning akan diuji dan hasil dari ketiga model base learner akan dilihat performanya dalam mendeteksi malware pada Android.

TABEL 1
Hasil Model Base Learner

Metode	Metrik			
	Accuracy	**Precision**	**Recall**	**F1-score**
SVM	0.9640	0.9681	0.9599	0.9640
Random Forest	0.9695	0.9736	0.9653	0.9694
XGBoost	0.9673	0.9731	0.9612	0.9671

Pada model *base learner* yang belum dioptimalkan dengan *hyperparameter tuning* ini menunjukkan performa model yang tinggi. Random Forest memberikan performa tertinggi dengan nilai metrik *accuracy* 0.9695, metrik *precision* 0.9736, metrik *recall* 0.9652, dan metrik *F1-score* 0.9694 lebih tinggi dibandingkan dengan 2 model *base learner* lainnya. Model *base learner* sudah menunjukkan performa yang tinggi dalam mendeteksi malware pada Android.

B. Hasil Model Base Learner Hyperparameter Tuning

Selanjutnya model base learner yang dioptimalkan dengan hyperparameter tuning dan diuji untuk melihat performanya dalam meningkatkan performa model setelah dioptimasi.

TABEL 2
Hasil Model Base Learner Hyperparameter Tuning

Metode	Metrik			
	Accuracy	**Precision**	**Recall**	**F1-score**
SVM	0.9651	0.9714	0.9585	0.9649
Random Forest	0.9700	0.9756	0.9643	0.9699
XGBoost	0.9695	0.9765	0.9622	0.9693

Terlihat pada Tabel 2, model yang sudah dioptimalkan dengan *hyperparameter tuning* memberikan hasil evaluasi yang lebih tinggi dibandingkan dengan model yang belum dioptimalkan dengan *hyperparameter tuning*. Terlihat model Random Forest memberikan hasil metrik *accuracy* 0.9700, metrik *precision* 0.9756, metrik *recall* 0.9643, metrik dan F1-score 0.9699. Untuk metrik *recall* terbesar diberikan oleh model XGBoost yang sudah dioptimasi dengan nilai sebesar 0.9765. Namun perbedaan hasil performa model yang sudah dioptimalkan dan yang belum dioptimalkan tidak signifikan.

TABEL 3
Hasil Model Base Learner Hyperparameter Tuning

Metode	Metrik			
	Accuracy	**Precision**	**Recall**	**F1-score**
SVM	0.11%	0.34%	-0.14%	0.10%
Random Forest	0.05%	0.20%	-0.11%	0.05%
XGBoost	0.23%	0.35%	0.11%	0.23%

Pada Tabel 3 menunjukkan perbandingan performa antara model yang belum dioptimalkan dan model yang sudah dioptimalkan dengan *GridSearchCV hyperparameter tuning*. Terlihat peningkatan performa tertinggi terdapat pada peningkatan *precision* XGBoost sebesar 0.35%. Namun pada nilai *recall* model SVM dan Random Forest terjadi penurunan sebesar 0.14% pada SVM dan 0.11% pada model Random Forest.

C. Hasil Model Stacking Ensemble

Metode *ensemble stacking* digunakan untuk menggabungkan beberapa model pembelajaran mesin yang telah dioptimasi dengan *hyperparameter tuning*, yaitu Support Vector Machine (SVM), Random Forest, dan XGBoost. Tujuan penerapan model *stacking* ini untuk memanfaatkan keunggulan masing-masing model *base learner* untuk menghasilkan model *meta-learner* yang mampu memberikan performa yang lebih akurat dalam mendeteksi malware pada Android.

TABEL 4
Hasil Model Stacking Ensemble

Metode	Metrik			
	Accuracy	Precision	Recall	F1-score
Stacking	0.9702	0.9769	0.9633	0.9700

Berdasarkan evaluasi hasil model *stacking*, model menunjukkan kinerja yang sangat baik dengan nilai metrik *accuracy* 0.9702, *precision* 0.9769, *recall* 0.9633, dan *F1-score* 0.9700. Hasil prediksi model menunjukkan bahwa model *stacking* mampu mencapai tingkat akurasi yang lebih tinggi dibandingkan dengan model individual.

D. Analisis Hasil

Setelah semua model diuji dan evaluasi, model *stacking* akan dilihat seberapa jauh peningkatan performanya dibandingkan dengan performa model *base learner* yang sudah dioptimasi.

TABEL 5
Perbandingan Performa Model Stacking dan Model Individual Hyperparameter Tuning

Metode	Metrik			
	Accuracy	Precision	Recall	F1-score
SVM	0.53%	0.57%	0.50%	0.53%
Random Forest	0.02%	0.14%	-0.11%	0.01%
XGBoost	0.07%	0.04%	0.11%	0.07%

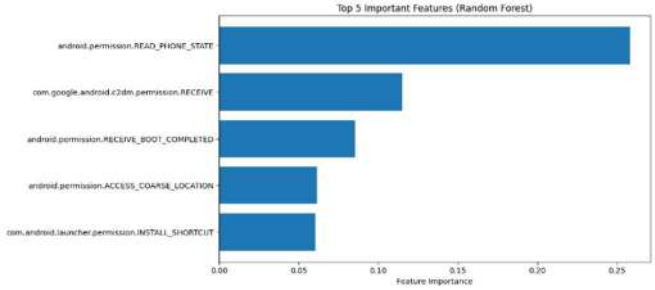
Pada perbandingan model *base learner* pertama SVM, model *stacking* memberikan peningkatan yang konsisten pada tiap metrik evaluasi. Nilai peningkatan *accuracy* sebesar 0.53%, *precision* sebesar 0.57%, *recall* sebesar 0.50%, dan *F1-score* sebesar 0.53%. Hal ini menunjukkan bahwa walau model SVM sudah memiliki performa yang baik setelah dioptimasi, model *stacking* masih mampu menghasilkan model yang lebih akurat.

Pada model Random Forest peningkatan relatif kecil dengan hasil metrik evaluasi *accuracy* sebesar 0.02%, metrik *precision* 0.14%, dan metrik *F1-score* 0.01% sedangkan nilai metrik *recall* mengenai sedikit penurunan sebesar -0.11%. Hasil menunjukkan bahwa model Random Forest sudah bekerja dengan tingkat akurasi yang tinggi sehingga ruang peningkatan yang diberikan oleh model *stacking* tidak terlalu signifikan. Meskipun demikian, model *stacking* tetap menunjukkan performa yang lebih unggul dibandingkan dengan model Random Forest individual yang telah dioptimasi.

Pada model *base learner* terakhir XGBoost menunjukkan peningkatan yang konsisten dengan nilai *accuracy* sebesar 0.07%, *precision* 0.04%, *recall* 0.11%, dan *F1-score* sebesar 0.07%. Walau peningkatan yang diberikan model *stacking* tidak terlalu besar, model *stacking* masih mampu memberikan performa yang lebih akurat serta efektif dibandingkan dengan model individual dalam mendeteksi malware pada Android.

E. Feature Importance

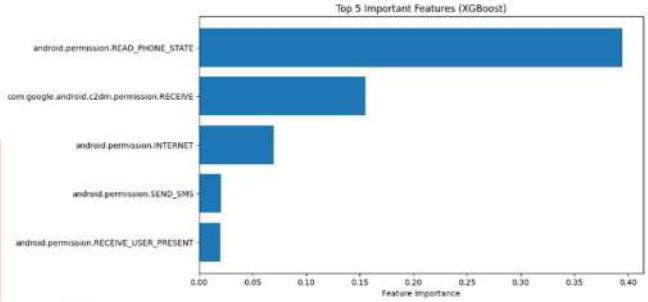
Nilai kontribusi tiap fitur akan dilihat dalam membantu model membedakan antara aplikasi malware dan aplikasi benign. Dalam penelitian ini, diambil lima fitur dengan nilai *importance* tertinggi seperti yang ditampilkan pada **Gambar 2.**



GAMBAR 2
Feature Importance Random Forest

Terlihat pada **Gambar 2**, fitur dengan kontribusi terbesar adalah fitur android.permissions.READ_PHONE_ STATE dengan nilai *importance* 0.25. *Permissions* ini memungkinkan aplikasi untuk membaca status, IMEI, dan informasi perangkat pengguna Android. Kemudian ada fitur com.google.c2dm.permissions.RECIEVE yang memungkin kan aplikasi untuk menerima perintah secara *remote* sehingga fitur ini sangat berbahaya bagi perangkat pengguna.

Fitur android.permission.RECIEVE_BOOT_ COMPLETED, android.Permissions.ACCESS_COARSE_ LOCATION, dan com.android.launcher.permissions. INSTAL_SHORTCUT juga memiliki nilai *importance* yang tinggi. Kelima fitur ini menjadi indikator utama model Random Forest dalam membuat keputusan klasifikasi.



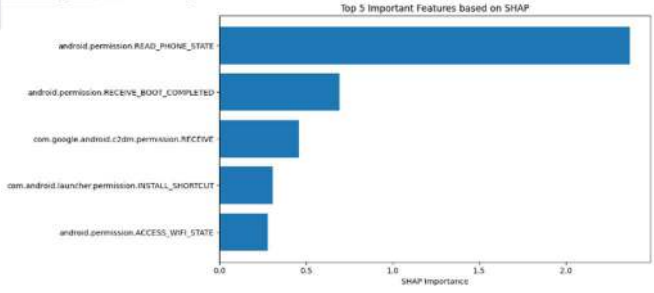
GAMBAR 3
Feature Importance XGBoost

Pada **Gambar 3**, fitur android.permissions.READ_ PHONE_STATE masih memiliki kontribusi terbesar dengan nilai mendekati 0.40. Hal ini menandakan fitur teresbut memiliki peran yang sangat penting dalam membantu model XGBoost dalam membedakan aplikasi benign dan aplikasi malware. Fitur com.google.c2dm.permissions.RECIEVE, android.permissions.INTERNET, android.permissions. SEND_SMS, dan android.permission.RECIEVE_USER_ PRESENT juga sangat berpengaruh dalam model XGBoost mengklasifikasikan aplikasi malware.

Berdasarkan hasil *feature importance* kedua model Random Forest dan XGBoost terlihat pola yang sama dalam penentuan fitur yang paling berpengaruh dalam mendeteksi malware Android. Fitur android.permissions.READ_ PHONE_STATE memiliki nilai kontribusi terbesar di kedua model yang menandakan fitur tersebut memiliki peran signifikan dalam membedakan aplikasi malware dan benign.

F. SHapley Additive ExPlanations (SHAP)

Setiap fitur juga akan dianalisis dengan *SHapley Additive exPlanations* (SHAP) untuk menjelaskan kontribusi tiap fitur dalam membantu model dalam menentukan aplikasi malware dan aplikasi benign.



GAMBAR 4
Hasil Analisis SHAP

Terlihat pada **Gambar 4**, fitur android.permissions. READ_PHONE_STATE masih memberikan kontribusi tertinggi dengan nilai sebesar 2.3716. Fitur lainnya android.

permission.RECEIVE_BOOT_COMPLETED sebesar 0.6939, com.google.android.c2dm.permission.RECEIVE sebesar 0.4592, com.android.launcher.permission. INSTALL_SHORTCUT sebesar 0.3068, dan android. permission.ACCESS_WIFI_STATE sebesar 0.2801 juga memberikan kontribusi yang cukup besar dalam membantu model mendeteksi aplikasi malware dan aplikasi benign.

Secara kesuluruhan fitur dengan izin sensitif dan terkait dengan aktivitas yang mencurigakan memberikan gambaran utama dari aplikasi yang dianggap berbahaya.

V. KESIMPULAN

Berdasarkan hasil analisis evaluasi terhadap tiga algoritma pembelajaran mesin yang telah dioptimasi dengan hyperparameter tuning model Support Vector Machine (SVM) menghasilkan nilai metrik *accuracy* 0.9651, metrik *precision* sebesar 0.9585, metrik *recall* sebesar 0.9714, dan nilai metrik *F1-score* 0.9649. Model Random Forest menghasilkan nilai *accuracy* 0.9700, *precision* 0.9756, *recall* sebesar 0.9643, dan nilai F1-score 0.9699. Model XGBoost memberikan nilai metrik *accuracy* 0.9695, metrik *precision* 0.9765, metrik *recall* 0.9622, dan metrik F1-score 0.9693.

Model stacking ensemble yang dibangun dengan Logistic Regression sebagai meta-learner mampu memberikan peningkatan performa dengan nilai metrik *accuracy* 0.9702, metrik *precision* 0.9769, metrik *recall* 0.9633, dan metrik *F1-score* 0.9700 lebih baik dibandingkan dengan performa model individual. Berdasarkan analisis fitur, fitur dengan nilai tertinggi memberikan nilai importance 0.2583 pada model Random Forest dan nilai importance 0.3946 pada model XGBoost. Analisis menggunakan Shapley Additive ExPlanations (SHAP) untuk fitur dengan nilai kontribusi tertinggi didapatkan nilai 2.3716 dalam mendeteksi malware pada Android.

REFERENSI

[1] N. Polatidis, S. Kapetanakis, M. Trovati, I. Korkontzelos, and Y. Manolopoulos, “FSSDroid: Feature subset selection for Android malware detection,” *World Wide Web*, vol. 27, no. 5, Sep. 2024, doi: 10.1007/s11280-024-01287-y.

[2] F. Akbar, M. Hussain, R. Mumtaz, Q. Riaz, A. W. A. Wahab, and K. H. Jung, “Permissions-Based Detection of Android Malware Using Machine Learning,” *Symmetry (Basel)*, vol. 14, no. 4, Apr. 2022, doi: 10.3390/sym14040718.

[3] J. M. Arif, M. F. A. Razak, S. Awang, S. R. T. Mat, N. S. N. Ismail, and A. Firdaus, “A static analysis approach for Android permission-based malware detection systems,” *PLoS One*, vol. 16, no. 9 September, Sep. 2021, doi: 10.1371/journal.pone.0257968.

[4] H. juan Zhu, Y. Li, L. min Wang, and V. S. Sheng, “A multi-model ensemble learning framework for imbalanced android malware detection,” *Expert Syst Appl*, vol. 234, Dec. 2023, doi: 10.1016/j.eswa.2023.120952.

[5] J. Zhang, Z. Xu, Z. Xiong, and L. Cai, “Android malware detection based on feature fusion and the improved stacking ensemble model,” *Journal of Computer Virology and Hacking Techniques*, vol. 21, no. 1, Dec. 2025, doi: 10.1007/s11416-025-00546-4.

[6] T. Li *et al.*, “Malware detection model based on stacking ensemble technique,” *Appl Soft Comput*, vol. 180, Aug. 2025, doi: 10.1016/j.asoc.2025.113338.

[7] A. Joshi and S. Kumar, “Stacking-based ensemble model for malware detection in android devices,” *International Journal of Information Technology (Singapore)*, vol. 15, no. 6, pp. 2907–2915, Aug. 2023, doi: 10.1007/s41870-023-01392-7.

[8] P. Sumalatha and G. S. Mahalakshmi, “MACHINE LEARNING BASED ENSEMBLE CLASSIFIER FOR ANDROID MALWARE DETECTION,” *International Journal of Computer Networks and Communications*, vol. 15, no. 4, pp. 111–122, Jul. 2023, doi: 10.5121/ijcnc.2023.15407.

[9] X. Wang, L. Zhang, K. Zhao, X. Ding, and M. Yu, “MFDroid: A Stacking Ensemble Learning Framework for Android Malware Detection,” *Sensors*, vol. 22, no. 7, Apr. 2022, doi: 10.3390/s22072597.

[10] A. M. R. AlSobeh, K. Gaber, M. M. Hammad, M. Nuser, and A. Shatnawi, “Android malware detection using time-aware machine learning approach,” *Cluster Comput*, vol. 27, no. 9, pp. 12627–12648, Dec. 2024, doi: 10.1007/s10586-024-04484-6.

[11] M. Gaber, M. Ahmed, and H. Janicke, “Zero day malware detection with Alpha: Fast DBI with Transformer models for real world application,” *Computers and Electrical* Engineering, vol. 128, p. 110751, Dec. 2025, doi: 10.1016/j.compeleceng.2025.110751.

[12] R. Y. Mawoh, J. B. A. Wacka, F. Tchakounte, C. Fachkha, and Kolyang, “An accurate approach to discriminate android colluded malware from single app malware using permissions intelligence,” *Sci Rep*, vol. 15, no. 1, Dec. 2025, doi: 10.1038/s41598-025-86568-w.

[13] H. Fathi, O. Malkawi, A. Al Tawil, A. Shaban, D. Ibrahim, and M. A. Aladaileh, “Reinforcement learning-driven feature selection for enhanced classification in cybersecurity: Applications in IoT security and malware detection,” *International Journal of Data and Network Science*, vol. 9, no. 4, pp. 813–822, Sep. 2025, doi: 10.5267/j.ijdns.2025.8.003.

[14] H. Bakır, “A new method for tuning the CNN pre-trained models as a feature extractor for malware detection,” *Pattern* Analysis *and Applications*, vol. 28, no. 1, Mar. 2025, doi: 10.1007/s10044-024-01381-x.

[15] E. Baghirov, “A comprehensive investigation into robust malware detection with explainable AI,” *Cyber Security and Applications*, vol. 3, Dec. 2025, doi: 10.1016/j.csa.2024.100072.