BAB I
PENDAHULUAN

1.1 Latar Belakang

Kualitas perangkat lunak merupakan faktor penting dalam keberhasilan sistem digital modern. Setiap perubahan kode *(commit)* yang dilakukan pengembang berpotensi memperkenalkan cacat baru yang dapat menurunkan reliabilitas sistem serta meningkatkan biaya pemeliharaan. Pada proyek berskala besar dengan frekuensi perubahan kode yang tinggi, inspeksi manual terhadap *commit* menjadi tidak efisien. Kondisi tersebut mendorong pengembangan metode otomatis berbasis data untuk membantu mengidentifikasi perubahan kode yang berisiko menimbulkan cacat (*bug*) [1].

Pendekatan *Software Defect Prediction* (SDP) tradisional bertujuan memperkirakan kemungkinan cacat pada tingkat modul atau file. Namun, SDP konvensional baru dapat memberikan informasi setelah perubahan digabungkan ke repositori utama sehingga prediksi cenderung terlambat dan kurang sesuai untuk pengembangan yang bersifat cepat dan iteratif. Untuk mengatasi keterlambatan tersebut, *Just-in-Time Defect Prediction* (JIT-DP) diperkenalkan sebagai pendekatan prediksi cacat pada tingkat *commit*. Pendekatan ini memanfaatkan metrik proses seperti *lines added*, *lines deleted*, jumlah file yang diubah, serta pengalaman pengembang untuk mengidentifikasi *commit* yang berpotensi menimbulkan cacat segera setelah perubahan dilakukan [1], [2].

Sejumlah penelitian menunjukkan bahwa model *machine learning* yang kompleks tidak selalu memberikan hasil terbaik dalam konteks *commit-level*. Nam *et al.* menemukan bahwa *Pre-trained Language Model* (PLM) sering mengandalkan pola permukaan *(surface cues)* sehingga tidak selalu unggul dalam memahami semantik perubahan kode [3]. Temuan serupa diperkuat oleh van Dinter *et al.* yang menunjukkan bahwa model *shallow learning* seperti *Random Forest* dan *XGBoost* dapat memberikan performa yang lebih stabil dan efisien dibandingkan *deep learning* untuk tugas JIT-DP [4]. Selain itu, faktor manusia juga terbukti

berpengaruh signifikan. Penelitian Cho *et al.* menunjukkan bahwa metrik pengalaman pengembang, seperti riwayat kontribusi dan kedekatan pengembang dengan subsistem tertentu, dapat meningkatkan kemampuan model dalam mengidentifikasi *commit* yang berisiko menimbulkan cacat. Studi tersebut menekankan bahwa JIT-DP tidak hanya bergantung pada metrik struktural kode, tetapi juga pada konteks perilaku pengembang. Meskipun demikian, banyak penelitian sebelumnya belum memanfaatkan metrik pengalaman secara optimal sehingga kontribusinya terhadap peningkatan performa masih terbatas [5].

Selain tantangan terkait model, kualitas data *commit* juga menjadi faktor kritis. Duan *et al.* menunjukkan bahwa keberadaan *duplicate changes* dalam riwayat proyek dapat menyebabkan bias yang mengakibatkan inflasi akurasi. Di sisi lain, faktor temporal juga memengaruhi performa model [6]. Song dan Minku menekankan bahwa perubahan pola data seiring waktu (*concept drift*) berdampak pada kestabilan prediksi, sementara Cabral dan Minku menunjukkan bahwa keterlambatan pelabelan (*verification latency*) dapat mengurangi reliabilitas model pada konteks pengembangan nyata. Oleh karena itu, pendekatan validasi berbasis waktu diperlukan untuk menghindari kebocoran informasi temporal yang umum terjadi pada pembagian data secara acak [2], [7].

Walaupun berbagai metode berbasis *deep learning* dan *reinforcement learning* telah diusulkan, pendekatan tersebut memiliki keterbatasan dalam interpretabilitas dan kebutuhan komputasi. Studi Nam *et al.* menegaskan bahwa model kompleks tidak selalu memahami konteks perubahan kode dengan baik [3]. Dalam praktiknya, algoritma *Random Forest* tetap menjadi pilihan yang stabil dan efisien, dengan kemampuan menangani ketidakseimbangan kelas serta menyediakan interpretabilitas melalui analisis *Feature Importance*. Temuan ini sejalan dengan penelitian lain pada ranah rekayasa perangkat lunak, seperti prediksi kegagalan aplikasi seluler, yang menunjukkan bahwa *Random Forest* mampu mencapai performa konsisten [8].

Dataset *commit-level* berskala besar seperti ApacheJIT, yang dirilis oleh Keshavarz dan Nagappan melalui Zenodo, menyediakan metrik proses, informasi

pengalaman pengembang, dan label *commit* cacat. Dataset ini sangat relevan untuk penelitian JIT-DP karena memuat konteks temporal dan distribusi kelas yang tidak seimbang sebagaimana terjadi pada proyek perangkat lunak nyata [9].

Pemilihan metode dalam penelitian ini didasarkan pada kebutuhan akan metode dasar yang stabil untuk menilai pengaruh skema validasi yang berfokus pada waktu. Penelitian ini tidak bertujuan untuk menyarankan atau menguji algoritma prediksi cacat yang paling terbaru, tetapi lebih kepada mengevaluasi bagaimana cara pembagian data yang berbeda, terutama antara *Chronological Split* dan *Random Split*, mempengaruhi hasil penilaian model. Dalam hal ini, penggunaan algoritma yang sudah dikenal seperti *Random Forest* menjadi penting karena algoritma ini telah banyak dipakai dalam penelitian *Just-in-Time Defect Prediction* sebelumnya, sehingga hasil yang didapat bisa dibandingkan langsung dengan studi sebelumnya. Dengan memilih algoritma yang cukup sederhana dan sudah banyak dipahami, penelitian ini bisa menghindari kebingungan yang muncul akibat kerumitan model, serta memastikan bahwa perbedaan hasil yang terlihat benar-benar berasal dari skema validasi yang berfokus pada waktu, bukan disebabkan oleh karakteristik algoritma yang dipakai.

Kontribusi penelitian ini adalah penyajian evaluasi prediksi cacat berdasarkan tingkat *commit* yang lebih realistis dengan menggunakan metode *Chronological Split* sebagai pendekatan utama, serta dilakukan analisis *Feature Importance* untuk meningkatkan kemudahan dalam memahami hasil prediksi. Pendekatan ini diharapkan dapat membantu memenuhi kebutuhan akan ketepatan prediksi, evaluasi yang realistis, dan kemudahan dalam memahami dalam konteks *Just-in-Time Defect Prediction*.

1.2 Rumusan Masalah

Dengan mempertimbangkan latar belakang yang telah dijelaskan, fokus dari rumusan masalah dalam studi ini adalah pada kemampuan model, metode penilaian, dan elemen yang memengaruhi estimasi cacat pada tahap *commit*. Pernyataan masalah yang diusulkan adalah sebagai berikut:

1. Bagaimana membangun model prediksi cacat perangkat lunak *(bug) Just-in-time* menggunakan algoritma *Random Forest* pada dataset ApacheJIT?
2. Bagaimana performa model yang telah dibangun dievaluasi secara akurat menggunakan metode validasi kronologis dengan metrik F1-Score, MCC, dan ROC-AUC?
3. Metrik *commit* apa yang paling berpengaruh terhadap prediksi cacat berdasarkan analisis *Feature Importance*?

1.3 Batasan Masalah

Agar penelitian ini tetap fokus dan mendalam, ruang lingkup penelitian dibatasi sebagai berikut:

1. Dataset

   Penelitian ini hanya menggunakan dataset *apachejit_total.csv* dari koleksi dataset ApacheJIT yang dipublikasikan oleh Keshavarz & Nagappan.
2. Algoritma

   Algoritma *machine learning* yang digunakan secara eksklusif adalah *Random Forest*. Penelitian ini tidak melakukan perbandingan performa dengan algoritma lain seperti SVM, *Deep Learning*, atau lainnya.
3. Validasi Model

   Metode validasi yang digunakan untuk evaluasi final adalah *Chronological Split (Time-aware split)*. *Random Split* hanya digunakan sebagai pembanding untuk menunjukkan bias.
4. Metrik Evaluasi

   Performa model diukur secara kuantitatif menggunakan metrik *Precision, Recall, F1-Score, Matthews Correlation Coefficient* (MCC), dan *Area Under the Receiver Operating Characteristic Curve* (ROC-AUC).
5. Lingkup Prediksi

Prediksi dilakukan pada level *commit*, bukan pada level file, modul, atau kelas.

1.4 Tujuan Penelitian

Penelitian ini dilakukan untuk menilai dan menganalisis efektivitas algoritma *Random Forest* dalam memprediksi cacat *Just-in-Time* yang didasarkan pada tingkat *commit*. Tujuan dari penelitian ini dirancang untuk menjawab pertanyaan-pertanyaan yang telah ditentukan, yaitu sebagai berikut:

1. Membangun dan mengoptimalkan model prediksi cacat perangkat lunak *(bug)* menggunakan algoritma *Random Forest* dengan data dari dataset ApacheJIT.
2. Mengevaluasi performa model yang telah dibangun menggunakan metrik *F1-Score*, MCC, dan ROC-AUC dengan menerapkan skema validasi kronologis *(Chronological Split)*.
3. Mengidentifikasi metrik-metrik level *commit* yang paling berpengaruh dalam memprediksi cacat perangkat lunak (bug) melalui analisis *Feature Importance*.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini diharapkan bisa memberikan manfaat baik secara akademik maupun dalam penerapan di dunia nyata di bidang rekayasa perangkat lunak, terutama dalam hal prediksi cacat menggunakan pendekatan *Just-in-Time*.

Secara akademik, penelitian ini diharapkan bisa menambah informasi berbasis pengalaman mengenai penerapan algoritma *Random Forest* dalam *Just-in-Time Defect Prediction* dengan cara evaluasi yang memperhatikan waktu. Hasil penelitian ini bisa digunakan sebagai dasar bagi penelitian lain yang membahas prediksi cacat pada level *commit*, terutama yang mempertimbangkan aspek waktu dan kemudahan dalam memahami model.

Secara praktis, penelitian ini diharapkan bisa memberikan gambaran tentang kemungkinan menggunakan model prediksi cacat sebagai sistem peringatan dini dalam proses pengembangan perangkat lunak. Model yang dibuat bisa

membantu para pengembang untuk fokus pada review kode di *commit* yang memiliki risiko cacat lebih besar, sehingga kemungkinan adanya bug yang masuk ke tahap produksi bisa ditekan, serta biaya pemeliharaan perangkat lunak bisa dikurangi.

1.6 Sistematika Penulisan

1. BAB I: Pendahuluan – berisi latar belakang, rumusan masalah, tujuan, batasan, manfaat, dan sistematika penulisan.
2. BAB II: Tinjauan Pustaka – membahas penelitian terdahulu, konsep dasar *defect prediction*, JIT, dan *Random Forest*.
3. BAB III: Metodologi Penelitian – menjelaskan alur penelitian mulai dari preprocessing data, pembagian dataset, pemodelan, *Tuning*, hingga evaluasi.
4. BAB IV: Hasil dan Pembahasan – menyajikan hasil eksperimen, evaluasi performa, serta analisis fitur penting.
5. BAB V: Kesimpulan dan Saran – memuat ringkasan temuan utama serta rekomendasi untuk penelitian selanjutnya.