JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 2, Juni 2024, Pp. 782-791

NAIVE BAYES CLASSIFICATION UNTUK PREDIKSI CACAT PERANGKAT LUNAK

Edwin Hari Agus Prastyo[1)], Suhartono[2)], M. Faisal[3)], Muhammad Ainul Yaqin[4)], Reza Augusta Jannatul Firdaus[5)]

1. Magister Informatika, Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang, Indonesia
2. Magister Informatika, Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang, Indonesia
3. Magister Informatika, Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang, Indonesia
4. Magister Informatika, Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang, Indonesia
5. Informatika, Teknologi Informasi, Universitas Hasyim Asy'ari, Indonesia

Article Info
Kata Kunci: analisis data; cacat perangkat lunak; *naive bayes*; prediksi; *python.*

Keywords: *data analysis*; *naive bayes; software bug prediction; pyhton.*

Article history:
Received 2 March 2024
Revised 16 March 2024
Accepted 30 March 2024
Available online 1 June 2024

DOI :
https://doi.org/10.29100/jipi.v9i2.5508

* Corresponding author.
Edwin Hari Agus Prastyo
E-mail address:
220605220005@student.uin-malang.ac.id

ABSTRAK
Cacat perangkat lunak atau bug merupakan bagian yang tak terhindarkan dari pengembangan perangkat lunak dan dapat memiliki dampak signifikan terhadap keandalan dan kinerja aplikasi perangkat lunak. Penelitian ini bertujuan untuk meningkatkan prediksi dan pemantauan cacat perangkat lunak dalam konteks pengembangan perangkat lunak. Kestabilan sistem dan pencegahan cacat perangkat lunak menjadi hal yang sangat krusial dalam industri perangkat lunak. Untuk mencapai tujuan tersebut, berbagai tindakan proaktif seperti pengujian perangkat lunak yang ketat, pemeliharaan rutin, dan pemantauan sistem secara berkelanjutan perlu diterapkan. Masalah utama yang menjadi fokus dalam penelitian ini adalah kurangnya prediksi yang efisien terkait dengan cacat perangkat lunak selama proses pengembangan. Dalam penelitian ini, metode Naive Bayes Classification dimanfaatkan untuk mengatasi tantangan tersebut. Hasil uji coba yang dilakukan terhadap seluruh dataset menunjukkan bahwa metode Naive Bayes mampu menghasilkan klasifikasi dengan tingkat akurasi yang sangat tinggi, mencapai nilai 0,98%. Hasil ini mengindikasikan bahwa metode ini berpotensi menjadi alat yang sangat efektif dalam memprediksi cacat perangkat lunak dan mencegahnya selama proses pengembangan perangkat lunak. Selain itu, melalui analisis regresi linear, model ini memiliki nilai "Intercept" (intersepsi) sebesar -0.09359968647139849 dan koefisien "Coef" sebesar 0.00761893. "Intercept" mencerminkan titik awal atau baseline dalam prediksi model ini, sementara koefisien "Coef" mengindikasikan sejauh mana perubahan dalam variabel independen berpengaruh terhadap prediksi variabel dependen. Hasil penelitian ini memiliki dampak yang signifikan dalam konteks implementasi metode Naive Bayes untuk analisis prediksi cacat perangkat lunak, terutama dalam penggunaan bahasa pemrograman Python dengan bantuan Google Colab. Implementasi metode ini dapat membantu dalam memitigasi risiko dan meningkatkan kualitas perangkat lunak selama proses pengembangan

ABSTRACT
Software bugs are an inevitable part of software development and can have a significant impact on the reliability and performance of software applications. This research aims to improve the prediction and monitoring of software defects in the context of software development. System stability and software malfunction prevention have become crucial in the software industry. To achieve that goal, proactive measures such as rigorous software testing, routine maintenance, and continuous system monitoring need to be implemented. The main problem that has been the focus of this research is the lack of efficient prediction associated with software defects during the development process. In this study, the Naive Bayes classification method was used to tackle the challenge. Test results carried out on the entire data set showed that the Native Bayes method was able to produce classifications with a very high degree of accuracy, reaching a value of 0.98%. These results indicate that the method could potentially be a very effective tool for predicting software defects and

preventing them during the software development process. Furthermore, through linear regression analysis, this model has an "intercept" value of -0.09359968647139849 and a coefficient of "coef" of 0.00761893. The results of this research have had a significant impact in the context of the implementation of the Naive Bayes method for software bug prediction analysis, especially in the use of the Python programming language with the help of Google Colab. Implementation of this method can help in mitigating risk and improving the quality of software during the development process.

I. PENDAHULUAN

Di era teknologi yang terus berkembang pesat, memprediksi cacat perangkat lunak telah menjadi langkah krusial dalam mengidentifikasi dan mengurangi potensi masalah dalam pengembangan perangkat lunak. Salah satu pendekatan yang telah dianalisis secara mendalam adalah penerapan klasifikasi *Naive Bayes* untuk meningkatkan prediksi cacat perangkat lunak [1]. Naive Bayes adalah teknik klasifikasi yang didasarkan pada Teorema Bayes dan asumsi independensi bersyarat antara fitur-fitur yang diberikan variabel kelas. Metode ini banyak digunakan dalam memprediksi cacat perangkat lunak karena keefektifannya dalam mengklasifikasikan kode statis [2]. Metode ini bekerja dengan menghitung probabilitas sebuah titik data termasuk dalam kelas tertentu berdasarkan probabilitas fitur-fiturnya. Dalam prediksi cacat perangkat lunak, Naive Bayes dipilih karena kesederhanaan, kecepatan, dan kemampuannya untuk menangani sejumlah besar fitur secara efisien. Selain itu, Naive Bayes dipilih karena kemampuannya untuk menangani data kategorikal dan numerik, sehingga cocok untuk dataset prediksi cacat perangkat lunak yang beragam. Dengan mengasumsikan independensi antar fitur, Naive Bayes menyederhanakan proses pemodelan dan sangat berguna ketika berhadapan dengan data pelatihan yang terbatas. Selain itu, kinerjanya dapat ditingkatkan dengan teknik seperti Information Gain untuk pemilihan atribut, sehingga meningkatkan akurasi prediksi [3].

Masalah utama yang ingin diselesaikan dalam penelitian ini adalah kurangnya prediksi yang efisien terkait dengan cacat perangkat lunak dalam pengembangan perangkat lunak. Pengawasan berkelanjutan dan manajemen risiko adalah aspek penting dari pengembangan dan pemeliharaan perangkat lunak. Manajemen risiko yang efektif melibatkan identifikasi awal dan analisis risiko, implementasi tindakan korektif, pemantauan terus-menerus, dan penilaian ulang [4]. Proses ini penting untuk mengurangi cacat yang tidak terduga dan kecelakaan sistem, memastikan integritas data, dan meningkatkan pengalaman pengguna dan keandalan aplikasi [5]. Pengawasan berkelanjutan sangat penting untuk mengidentifikasi cacat kendaraan utama dan mengurangi kecelakaan akibat gangguan yang terkait dengan kendaraan [6]. Ini juga memainkan peran penting dalam meningkatkan keselamatan kendaraan berat dengan mengurangi kecelakaan terkait cacat melalui inspeksi dan program pemeliharaan [7].

Dalam bidang manajemen risiko, penggunaan Bayesian Networks telah diusulkan untuk mendukung pengambilan keputusan dalam berbagai kegiatan desain perangkat lunak, berkontribusi pada pengelolaan risiko teknologi yang efektif dalam proyek software. [2]. Ini menunjukkan pentingnya memanfaatkan teknik dan metodologi canggih untuk mengatasi risiko dalam pengembangan dan pemeliharaan. Metode klasifikasi Naive Bayes telah banyak digunakan di berbagai bidang, termasuk prediksi kesalahan perangkat lunak, analisis sentiment, dan prediksi penyakit. Dalam bidang prediksi kesalahan perangkat lunak, para peneliti telah menerapkan klasifikator Naive Bayes untuk memprediksi kesalahan software menggunakan teknik seperti sampel dan pemilihan fitur [1], integrasi keseimbangan berbasis distribusi dan bagging ensemble [2], dan perbandingan dengan algoritma klasifikasi lainnya.[3]. Studi ini telah menunjukkan efektivitas klasifikator Naive Bayes dalam memprediksi cacat perangkat lunak. Di bidang analisis perasaan klasifikator Naive Bayes telah digunakan untuk menganalisis perasaan masyarakat di platform media sosial seperti Twitter [4].

Metode ini telah diterapkan untuk menganalisis sentiment selama pemilihan 2020 dalam konteks pandemi COVID-19. Hasil penelitian ini menunjukkan penerapan klasifikator Naive Bayes dalam tugas analisis sentimen. Selain itu, klasifikator Naive Bayes juga telah digunakan dalam prediksi penyakit. Misalnya, di bidang perawatan kesehatan, klasifikator Naive Bayes telah digunakan untuk mengidentifikasi penyakit seperti tuberkulosis [5] dan gingivitis [8]. Studi ini menyoroti akurasi dan efektivitas klasifikator Naive Bayes dalam prediksi penyakit.

Selain itu, klasifikator Naive Bayes telah diterapkan di bidang lain, seperti memprediksi kelulusan siswa [9] dan mendeteksi serangan jaringan.[10]. Studi ini menunjukkan versatilitas klasifikator Naive Bayes dalam berbagai tugas prediksi. Sebagai kesimpulan, klasifikator Naive Bayes adalah metode yang banyak digunakan untuk tugas prediksi di berbagai domain. Efektifitasnya telah ditunjukkan dalam prediksi kesalahan perangkat lunak, analisis sentiment, prediksi penyakit, dan bidang lainnya.

Metode Naive Bayes adalah pilihan yang menarik karena kesederhanaannya. Meskipun asumsi bahwa atribut-atribut adalah independen (asumsi "naif") mungkin tidak selalu berlaku dalam konteks nyata, metode ini sering memberikan hasil yang baik dalam masalah klasifikasi, termasuk prediksi cacat perangkat lunak. Dalam penelitian [11] bahwa Naive Bayes (NB) adalah salah satu metode yang baik dalam klasifikasi. NB menggunakan teori probabilitas sebagai dasar teorinya dan memiliki tingkat kecepatan dan akurasi yang tinggi ketika diterapkan pada basis data yang besar. NB dapat menentukan kelas dari suatu data pada saat klasifikasi dengan menguji semua label pada data menggunakan teorema Bayes. Kelas yang memiliki nilai probabilitas tertinggi menjadi prediksi dari metode tersebut.

Bahasa pemrograman Python dipilih sebagai platform implementasi metode Naive Bayes dalam penelitian ini karena Python menyediakan beragam library yang kuat untuk analisis data dan pembelajaran mesin. Referensi paper yang relevan dalam konteks penggunaan Python untuk analisis data dan pembelajaran mesin adalah "Python for Data Analysis" oleh Wes McKinney [12] Python juga memiliki alat visualisasi yang efektif seperti Matplotlib dan Seaborn untuk memvisualisasikan hasil analisis data. Implementasi Python melibatkan langkah-langkah seperti pembacaan dataset, pemrosesan data, pelatihan model, dan evaluasi kinerja model. Analisis data adalah tahap penting dalam penelitian ini, dan metode Python memudahkan pembacaan data, pembersihan, penghapusan data yang tidak relevan, serta eksplorasi data untuk mengidentifikasi pola atau hubungan antara atribut perangkat lunak dan keberadaan cacat [10]. Evaluasi performa model prediksi cacat perangkat lunak adalah tahap akhir dalam penelitian ini. Metrik performa seperti akurasi, presisi, dan recall akan digunakan untuk mengevaluasi model.

Hasil evaluasi akan membantu dalam memahami sejauh mana model Naive Bayes efektif dalam memprediksi cacat perangkat lunak, pembelajaran mesin berbasis pengawasan digunakan untuk mengevaluasi kemampuan pembelajaran mesin dalam Prediksi Perilaku Perangkat (SBP). Studi ini membahas pengklasifikasi Naïve Bayes (NB), Pengklasifikasi pembelajaran mesin yang dibahas diterapkan pada tiga dataset berbeda yang diperoleh dari buku [13], Para peneliti sebelumnya telah mengembangkan dan menerapkan berbagai pendekatan prediksi bug yang berbeda dalam hal akurasi, kompleksitas, dan data masukan yang dibutuhkan namun belum mencapai hasil yang di inginkan [14].

Prediksi cacat perangkat lunak penting karena memungkinkan pengembang perangkat lunak untuk mengalokasikan sumber daya yang tersedia untuk menciptakan produk perangkat lunak berkualitas tinggi yang dapat membantu dalam setiap proses bisnis perusahaan [15]. Dengan memprediksi cacat modul awal, pengembang dapat mengidentifikasi masalah potensial sejak dini dan mengalokasikan sumber daya secara tepat. Penggunaan klasifikasi Naive Bayes dalam prediksi cacat perangkat lunak telah terbukti efektif.

Dalam sebuah penelitian oleh [16] integrasi SMOTE (Synthetic Minority Oversampling Technique) dengan Naive Bayes dan Regresi Logistik, berdasarkan optimisasi gerombolan partikel, meningkatkan prediksi cacat perangkat lunak. Studi ini menunjukkan bahwa pendekatan ini berhasil melampaui metode sebelumnya dalam hal kinerja. Penelitian ini mencakup lima penelitian yang berfokus pada prediksi cacat perangkat lunak, metode input yang digunakan, hasil, representasi perantara, dan keterbatasan masing-masing penelitian.

Studi pertama oleh [17] menyelidiki prediksi bug perangkat lunak yang terkait dengan penuaan (ARB) menggunakan data dari repositori bug. Mereka mengembangkan algoritme SEARCH_KEYWORD untuk prediksi ARB dengan representasi perantara berupa prediksi ARB. Namun, penelitian ini memiliki keterbatasan seperti proporsi yang tidak seimbang antara file yang rentan ARB dan file yang bebas ARB, ketersediaan data pelatihan yang terbatas, dan analisis komparatif yang terbatas pada sekumpulan pengklasifikasi dan kumpulan data tertentu.

Studi kedua oleh [18] berkaitan dengan prediksi cacat perangkat lunak menggunakan pembelajaran mesin berdasarkan data historis cacat perangkat lunak. Mereka menggunakan algoritma pembelajaran mesin yang diawasi (Naïve Bayes, SVM, ANN) untuk memprediksi kesalahan perangkat lunak di masa depan. Meskipun menyebutkan tingkat akurasi yang tinggi, penelitian ini tidak memiliki metrik khusus, tidak membandingkan dengan algoritma pembelajaran mesin lainnya, dan tidak membahas kesalahan tersembunyi atau memberikan informasi rinci tentang adaptasi perangkat lunak atau peningkatan pemanfaatan sumber daya.Studi ketiga oleh [19] [19] berfokus pada pengembangan model prediksi tingkat keparahan bug menggunakan word2vec pada deskripsi teks laporan bug. Mereka menggunakan word2vec untuk menanamkan dan memprediksi tingkat keparahan bug dengan vektor bernilai real sebagai representasi perantara. Namun, penelitian ini memiliki keterbatasan, seperti penyetelan hyperparameter yang memakan waktu, ketergantungan kinerja pada konfigurasi hyperparameter, fokus pada laporan bug berbasis teks, dan kinerja pengklasifikasi yang bervariasi berdasarkan data dan kemunculan kata.

Studi keempat oleh [20] membahas pembelajaran fitur semantik mendalam untuk prediksi cacat perangkat lunak menggunakan Deep Belief Network. Mereka mengekstrak fitur semantik dari vektor token yang berasal dari Abstract Syntax Trees (AST) dan perubahan kode sumber. Sayangnya, abstrak tersebut tidak memiliki detail spesifik tentang model dan data, yang kemungkinan besar disediakan dalam makalah lengkapnya.

Studi kelima oleh [21]menganalisis asumsi Naive Bayes pada data kerusakan perangkat lunak, menggunakan data kerusakan perangkat lunak yang tersedia untuk umum dari NASA. Mereka melakukan praproses data

menggunakan Principal Component Analysis (PCA) dan menerapkan berbagai metode untuk menganalisis asumsi Naive Bayes. Namun, penelitian ini terbatas pada analisis asumsi Naive Bayes pada data cacat perangkat lunak dari NASA, dan hasilnya mungkin tidak dapat digeneralisasi ke dataset atau domain lain.

Pada penelitian ini, implementasi metode Naive Bayes pada Python di lingkungan Google Colab akan digunakan untuk menganalisis prediksi bug pada dataset JM1. Menurut Zaidi [22], pembobotan atribut pada Naive Bayes dapat membantu mengurangi dampak kegagalan prediksi. Disarankan bahwa algoritma Naive Bayes efektif untuk prediksi cacat perangkat lunak. Dalam sebuah penelitian oleh [23], algoritma Naive Bayes mencapai akurasi 69,18% dan nilai AUC 0,771, yang menunjukkan kinerja klasifikasinya. Implementasi ini bertujuan untuk memanfaatkan kekuatan Naive Bayes dalam prediksi cacat perangkat lunak, terutama dalam konteks dataset JM1, untuk meningkatkan akurasi dan kemampuan prediksi.

Selain itu, dijelaskan di penelitian Khodijah [24] membandingkan metode yang berbeda untuk menangani data yang tidak seimbang dalam prediksi kesalahan perangkat lunak dan menemukan bahwa klasifikator Naive Bayes mengasumsikan independensi kondisional atribut, yang cocok untuk prediksi cacat software. Ini menyoroti relevansi klasifikasi Naive Bayes dalam konteks ini. Diharapkan bahwa hasil penelitian ini dapat menjadi landasan bagi pengembang perangkat lunak untuk meningkatkan kualitas perangkat lunak dan mengidentifikasi potensi masalah sejak dini. Dengan adanya prediksi cacat perangkat lunak yang lebih efisien, dapat diharapkan bahwa risiko dan dampak dari cacat perangkat lunak dapat diminimalkan.

II. METODE PENELITIAN

Dalam penelitian ini, metodologi yang dijalankan terbagi menjadi beberapa tahapan kunci:

A. Pengumpulan Data

Data yang digunakan berasal dari dataset JM1-Dataset-Attributes-Prediction, yang melibatkan atribut-atribut perangkat lunak dan label cacat. Dataset ini menyediakan informasi terkait karakteristik perangkat lunak, dan label cacat memberikan informasi tentang apakah suatu entitas perangkat lunak mengalami cacat atau tidak. Dengan menggunakan dataset ini, dapat dilakukan analisis terhadap atribut-atribut perangkat lunak untuk memahami pola dan hubungan yang mungkin mempengaruhi kemungkinan terjadinya cacat pada perangkat lunak tersebut. 4301 entries, 0 to 4300 Data columns (total 22 columns):

TABEL I
DATA SET JM1

LOC	V(G)	EV(G)	IV(G)	N	…
11.0	2.0	1.0	2.0	20.0	….
14.0	2.0	1.0	1.0	21.0	…
10.0	2.0	1.0	2.0	15.0	….
5.0	1.0	1.0	1.0	11.0	
10.0	3.0	3.0	1.0	21.0	….
..	….	…	….	….	

B. Eksplorasi dan Visualisasi Data

Eksplorasi Data dengan Histogram:

Sebelum menerapkan klasifikasi Naive Bayes, pemahaman distribusi data perlu diperoleh. Histogram digunakan untuk mengidentifikasi pola distribusi setiap fitur dalam dataset JM1, memungkinkan penentuan adanya outlier atau distribusi yang tidak merata yang dapat memengaruhi kinerja model.

Pengukuran Hubungan antar Variabel dengan Covariance: Untuk mengetahui sejauh mana setiap fitur berkolerasi satu sama lain dalam dataset JM1, covariance digunakan. Hubungan arah antara dua variabel dapat dianalisis melalui covariance. Ketika nilai covariance negatif ditemui, itu menunjukkan hubungan yang berlawanan, sementara nilai covariance positif mengindikasikan kecenderungan fitur-fitur tersebut untuk bergerak bersamaan.

Visualisasi Korelasi dengan Heatmap:

Efektivitas visualisasi matriks korelasi antar fitur pada dataset ditingkatkan melalui penggunaan heatmap. Dengan heatmap, hubungan antar fitur dapat diperhatikan dengan jelas, memunculkan pola korelasi yang berperan dalam proses seleksi fitur atau pemahaman data yang lebih mendalam.

Representasi Data dengan Scatter Plot:

Kelebihan cara scatter plot dalam melihat hubungan langsung antara dua variabel dieksplorasi pada dataset JM1.

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 2, Juni 2024, Pp. 782-791

Scatter plot membantu mengidentifikasi pola hubungan antara pasangan fitur tertentu, memfasilitasi penentuan pola linier atau non-linier yang dapat memengaruhi hasil prediksi menggunakan model Naive Bayes.

C. Pemrosesan Data

Data awal yang diberikan adalah sebagai berikut: loc = [11.0, 14.0, 10.0, 5.0, 10.0]. Langkah-langkah normalisasi menggunakan metode Min-Max Scaling dilakukan sebagai berikut: Pertama, kita menghitung nilai minimum (min_loc) dan nilai maksimum (max_loc) dari fitur "loc", di mana nilai terkecilnya adalah 5.0 dan nilai terbesarnya adalah 14.0. Kedua, kita normalisasi setiap nilai dalam fitur "loc" menggunakan rumus: normalized_loc = (loc - min_loc) / (max_loc - min_loc). Ketiga, kita menghitung nilai normalisasi untuk setiap data dalam fitur "loc" menggunakan rumus tersebut, yang menghasilkan nilai normalisasi sebagai berikut: normalized_loc = [(11.0 - 5.0) / (14.0 - 5.0), (14.0 - 5.0) / (14.0 - 5.0), (10.0 - 5.0) / (14.0 - 5.0), (5.0 - 5.0) / (14.0 - 5.0), (10.0 - 5.0) / (14.0 - 5.0)]. Sehingga, hasil normalisasi dari data awal adalah: normalized_loc = [0.6, 1.0, 0.5, 0.0, 0.5]. Dengan demikian, langkah-langkah normalisasi telah dilakukan dengan sukses, dan data telah dinormalisasi sesuai dengan metode Min-Max Scaling.

D. Ekstraksi Data

Ekstraksi fitur dari Dataset Prediksi Cacat Perangkat Lunak (Software Defect Prediction Dataset) melibatkan pemahaman dan analisis mendalam terhadap atribut-atribut yang ada (Hardoni, 2021). Dataset ini disediakan dengan tujuan untuk mendukung pengembangan model prediksi perangkat lunak yang dapat diulang, diverifikasi, diperdebatkan, dan ditingkatkan. Dalam konteks analisis, ekstraksi fitur adalah tahap awal yang penting dalam memahami karakteristik perangkat lunak yang akan dianalisis.

Berikut adalah beberapa atribut kunci dalam dataset dan cara ekstraksi fitur dari setiap atribut:

1) *loc (McCabe's Line Count of Code):* Atribut ini mengukur jumlah baris kode dalam perangkat lunak. Dalam ekstraksi fitur, ini dapat digunakan untuk mengukur kompleksitas kode dengan mengidentifikasi jumlah baris yang terlibat dalam perangkat lunak.
2) v(g) (McCabe "Cyclomatic Complexity"): Atribut ini mengukur kompleksitas perangkat lunak berdasarkan metrik McCabe. Dalam ekstraksi fitur, kompleksitas ini dapat digunakan sebagai indikator untuk tingkat kesulitan dalam memahami dan mengelola perangkat lunak.
3) ev(g) (McCabe "Essential Complexity"): Essential complexity adalah tingkat kompleksitas yang tetap dalam perangkat lunak. Ekstraksi fitur dari atribut ini dapat membantu dalam memahami kompleksitas inti perangkat lunak.
4) iv(g) (McCabe "Design Complexity"): Atribut ini mengukur kompleksitas desain perangkat lunak. Ekstraksi fitur dapat membantu dalam menganalisis sejauh mana desain memengaruhi kompleksitas perangkat lunak.
5) n (Halstead Total Operators + Operands): Atribut ini mencakup jumlah total operator dan operand dalam perangkat lunak. Ekstraksi fitur dari atribut ini dapat digunakan untuk mengukur jumlah entitas yang terlibat dalam perangkat lunak.
6) v (Halstead "Volume"): Halstead volume mengukur ukuran perangkat lunak. Dalam ekstraksi fitur, atribut ini dapat digunakan untuk memahami ukuran perangkat lunak yang dianalisis.
7) (Halstead "Program Length"): Program length adalah ukuran perangkat lunak berdasarkan metrik Halstead. Ekstraksi fitur dari atribut ini dapat membantu dalam mengidentifikasi panjang kode.
8) d (Halstead "Difficulty"): Atribut ini mengukur tingkat kesulitan dalam memahami perangkat lunak. Ekstraksi fitur dari atribut ini dapat memberikan wawasan tentang tingkat kompleksitas perangkat lunak.
9) i (Halstead "Intelligence"): Intelligence dalam konteks Halstead mengacu pada tingkat "kecerdasan" kode. Ekstraksi fitur dapat membantu dalam menganalisis sejauh mana kode tersebut dianggap cerdas.
10) *e (Halstead "Effort")*: Atribut ini mengukur upaya yang diperlukan untuk mengembangkan perangkat lunak. Ekstraksi fitur dari atribut ini dapat memberikan gambaran tentang seberapa besar usaha yang diperlukan dalam pengembangan.

E. *Data Normalization (Min-Max Normalization)*

Berikun contoh perhitungan dari proses normalisasi Min-Max.

Data Awal:
v : [3, 6, 9, 12]
b : [15, 18, 21, 24]

Normalisasi Min-Max untuk Kolom 'v'

$$v_{scaled} = \frac{v - \min(v)}{\max(v) - \min(v)} \quad (1)$$

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 2, Juni 2024, Pp. 782-791

Hitungan:

$$v_1 = \frac{3 - \min(3)}{\max(12) - \min(3)} = \frac{0}{9} = 0$$
$$v_2 = \frac{6 - \min(3)}{\max(12) - \min(3)} = \frac{3}{9} = \frac{1}{3}$$
$$v_3 = \frac{9 - \min(3)}{\max(12) - \min(3)} = \frac{6}{9} = \frac{2}{3}$$
$$v_4 = \frac{12 - \min(3)}{\max(12) - \min(3)} = \frac{9}{9} = 1$$

Jadi, hasil normalisasi Min-Max untuk kolom 'v':

$$v_{scaled} = \left[0, \frac{1}{3}, \frac{2}{3}, 1\right] \quad (2)$$

Normalisasi Min-Max untuk Kolom 'b'

$$b_{scaled} = \frac{b - \min(b)}{\max(b) - \min(b)}$$

Hitungan:

$$b_1 = \frac{15 - \min(15)}{\max(24) - \min(15)} = \frac{0}{9} = 0$$
$$b = \frac{18 - \min(15)}{\max(24) - \min(15)} = \frac{3}{9} = \frac{1}{3}$$
$$v_3 = \frac{21 - \min(15)}{\max(24) - \min(15)} = \frac{6}{9} = \frac{2}{3}$$
$$v_4 = \frac{24 - \min(15)}{\max(24) - \min(15)} = \frac{9}{9} = 1$$

Jadi, hasil normalisasi Min-Max untuk kolom 'v':

$$b_{scaled} = \left[0, \frac{1}{3}, \frac{2}{3}, 1\right]$$

F. Implementasi Model Prediksi

Selanjutnya dimasukkan keproses pelatihan dilakukan menggunakan data yang telah dinormalisasi sebelumnya, termasuk fitur-fitur 'v' dan 'b', serta target klasifikasi cacat perangkat lunak yang ditentukan sebagai 0 untuk tidak cacat dan 1 untuk cacat.

1) Model Naive Bayes dapat dijelaskan dengan fungsi keberhasilan (likelihood) dan posterior sebagai berikut:

$$P(Cacat|Data) = \frac{P(Data|cacat)x\,P(cacat)}{P(data)}$$

2) Prediksi Tingkat Cacat dengan Regresi Linier
Untuk memprediksi tingkat cacat perangkat lunak, model Regresi Linier telah diterapkan. Model ini menggunakan data tambahan mengenai tingkat cacat aktual dan fitur-fitur 'v' serta 'b'. Persamaan Regresi Linier dapat dituliskan sebagai berikut:

Cacat=β0+β1×$v$scaled+β2× B_{scaled}

Koefisien (β0, β1, β2) dan \(\beta_2\) pada model Regresi Linier telah ditentukan melalui proses pelatihan menggunakan data yang telah dinormalisasi. Model ini dapat digunakan untuk memprediksi tingkat cacat berdasarkan data baru yang diberikan.

G. Pengujian Model

Dalam pengujian model, akurasi model diukur dengan metrik Mean Squared Error (MSE) dan Root Mean Squared Error (RMSE). Tujuannya adalah mengevaluasi sejauh mana nilai prediksi mendekati nilai sebenarnya. Misalkan hasil prediksi ($\hat{y}$) dari model untuk tingkat cacat perangkat lunak dan nilai sebenarnya (y) sebagai berikut:

$\hat{y}$: [8, 12, 18, 22]

y : [10, 15, 20, 25]

1) Mean Squared Error (MSE)
MSE dihitung dengan menjumlahkan kuadrat selisih antara setiap prediksi dan nilai sebenarnya, kemudian dibagi dengan jumlah observasi.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 2, Juni 2024, Pp. 782-791

Hitungannya:

$MSE=41((8-10)^2+(12-15)^2+(18-20)^2+(22-25)^2)$

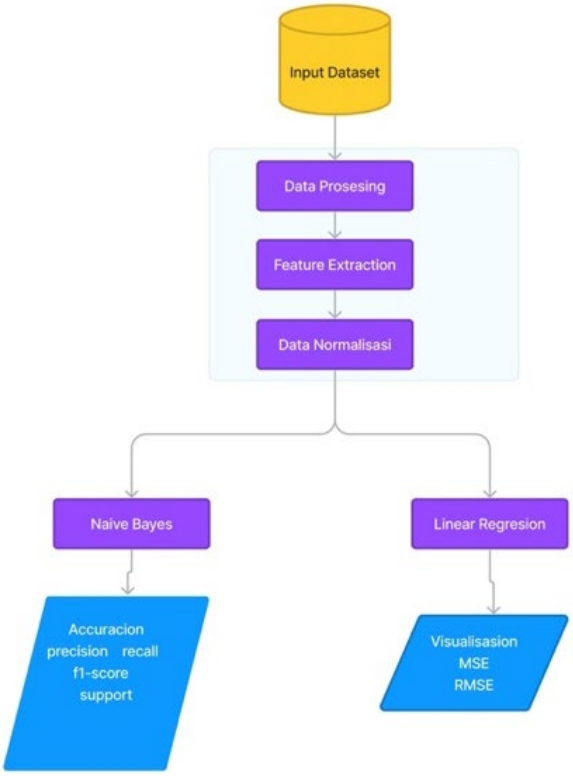
$MSE=41(4+9+4+9)=6.5$

2) Root Mean Squared Error (RMSE)
RMSE adalah akar kuadrat dari MSE, memberikan gambaran tentang seberapa besar kesalahan rata-rata.

$RMSE=\sqrt{MSE}$

Hitungannya: $RMSE=6.5\approx2.55$

Alur Penelitian

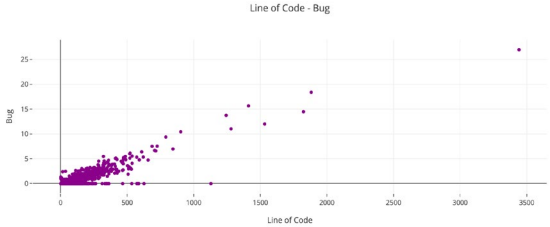


Gambar. 1. Alur Penelitian

Tahapan ini merupakan inti dari penelitian ini dan menghasilkan wawasan berharga tentang prediksi cacat perangkat lunak. Penelitian ini memberikan pandangan menyeluruh tentang pelaksanaan dari pengolahan data hingga pengujian model. Setiap tahap dalam metodologi ini memiliki peran penting dalam mencapai tujuan penelitian, yaitu menganalisis prediksi cacat perangkat lunak dengan metode Naive Bayes dan Regresi Linier serta mengukur akurasi hasil prediksi.

III. HASIL DAN PEMBAHASAN

Dari hasil klasifikasi dengan naïve bayes dapat di visualisakan dengan plot seperti gambar 3 berikut



Gambar 2. Visualisasi Plot Naive bayes

Untuk hasil dari metode Regresi linier dalam memprediksi tingkat cacat perangkat lunak, yang menghasilkan nilai Intercept dan Coef.

pengembang, atau kompleksitas kode) memiliki nilai nol. Dalam kasus ini, nilai "Intercept" yang negatif menunjukkan bahwa tanpa adanya faktor lain yang mempengaruhi, kita dapat mengharapkan prediksi cacat perangkat lunak yang relatif rendah atau mendekati nol.

Nilai "Coef" ([0.00761893]) menunjukkan seberapa besar pengaruh setiap unit peningkatan dalam variabel independen akan berkontribusi terhadap prediksi cacat perangkat lunak. Dalam konteks ini, misalnya, jika kita mempertimbangkan variabel independen seperti ukuran kode yang lebih besar, nilai "Coef" yang positif menunjukkan bahwa peningkatan ukuran kode akan berkontribusi pada peningkatan prediksi cacat perangkat lunak. Namun, penting untuk diingat bahwa nilai "Coef" ini harus dinilai dalam konteks spesifik dari variabel independen yang dipertimbangkan dalam model regresi.

Dengan demikian, dalam praktiknya, pemahaman nilai "Intercept" dan "Coef" dalam model regresi linear dapat membantu para profesional dalam pengembangan perangkat lunak untuk mengantisipasi dan mengelola risiko cacat dengan lebih efektif. Dengan mengetahui titik awal (Intercept) dan seberapa besar pengaruh setiap variabel (Coef), mereka dapat membuat keputusan yang lebih informasional dan proaktif dalam proses pengembangan perangkat lunak untuk mengurangi risiko cacat yang mungkin timbul.

A. *Pembahasan*

Kombinasi Nilai True or False

Eksperimen ini bertujuan untuk menguji berbagai kombinasi persentase nilai "*True*" dan "*False*" dalam data pelatihan dan mengamati pengaruhnya terhadap akurasi model yang digunakan. Eksperimen ini menggunakan algoritma Naive Bayes (NB) sebagai model prediktif.

TABEL II
DATABASE JM1

Persentase		Accuracy NB	Pengujian Akurasi	
False	True		RMSE	MSE
10%	90%	0.954	0.478	0.228
25%	75%	0.967	0.372	0.138
40%	60%	0.975	0.383	0.147
75%	25%	0.980	0.395	0.198

Dapat diamati bahwa komposisi data pelatihan memiliki pengaruh signifikan terhadap akurasi, menjelaskan bahwa kombinasi tertentu menghasilkan tingkat akurasi yang unggul karena pola atau faktor mendasar tertentu yang memengaruhi kinerja model. Misalnya, contoh di mana hasil optimal dicapai dengan 75% salah dan 35% benar dapat dijelaskan sebagai berikut.

Pertama, distribusi yang tidak merata antara kategori "*True*" dan "*False*" dalam kumpulan data pelatihan dapat secara signifikan mempengaruhi kemanjuran model. Dalam skenario seperti itu, proporsi 75% salah dan 35% benar dapat menandakan skenario di mana kategori “Salah” lebih lazim dalam kumpulan data, memungkinkan model untuk lebih memahami pola dan atribut kelas “Salah”, akibatnya menghasilkan akurasi yang lebih tinggi untuk kelas tertentu. Faktor penting lain yang memengaruhi hasil adalah signifikansi kedua kelas dalam konteks aplikasi. Misalnya, jika kategori “*False*” memiliki konsekuensi yang lebih substantif atau memberikan pengaruh yang lebih besar daripada kelas “*True*” dalam aplikasi tertentu, model cenderung memprioritaskan identifikasi akurat dari kelas “*False*”, sehingga meningkatkan akurasi yang terkait dengan kategori ini. Selanjutnya, campuran nilai "*True*" dan "*False*" yang mengarah ke akurasi yang meningkat juga dapat dipengaruhi oleh fitur atau variabel independen yang digunakan dalam model. Berbekal wawasan ini, strategi yang lebih efisien dan tepat untuk pengembangan model dapat diimplementasikan di berbagai skenario aplikasi.

Hasil eksperimen ini dapat digunakan sebagai panduan dalam memilih persentase yang paling sesuai untuk data pelatihan dalam model Naive Bayes.

Hasil dari Eksperimen Model Evaluasi Metrik yang dilakukan dalam beragam studi yang berkaitan dengan algoritma yang berbeda digambarkan di sini. Hasil evaluasi tersebut berasal dari matriks kebingungan yang mencakup metrik penting termasuk Recall, F1-Measure, Precision, dan Best Accuracy. Tabel berikutnya menggambarkan penjajaran kemanjuran algoritma yang berbeda yang didasarkan pada metrik ini, memberikan perspektif berharga untuk memahami efisiensi dan keandalan setiap model dalam lingkup evaluasi yang dijalankan, ditujukan di tabel 4.

TABEL IV
MATRIK PERBANDINGAN LANGSUNG DENGAN PENELITIAN SEBELUMNYA

Author	Algoritma	Recall	F1-measure:	Precision	Best Accuracy
[26]	Random Forest	-	-	-	0.55
[28]	Decision Tree	0.81	0.89		0.82
[25]	SVM-RBF	-	-	-	0.88
[28]	Random Forest	0.83	0.89	-	0.90
[27]	SVM	-	-	-	0.93
Penelitian Kami	Metode Yang di-usulkan Naïve Bayes	0.96	0.92	0.88	0.98

IV. KESIMPULAN

Berdasarkan hasil dari pengujian terhadap seluruh dataset yang dilakukan dapat disimpulkan bahwa metode naïve bayes menghasilkan klasifikasi dengan akurasi yang dihasilkan dengan nilai 0,98. Dan untuk hasil dari regresi liner ditemukan bahwa model ini memiliki nilai "Intercept" (intersepsi) sebesar -0.09359968647139849 dan koefisien "Coef" sebesar 0.00761893. Nilai "Intercept" mewakili titik awal atau baseline dari prediksi dalam konteks model ini, sementara koefisien "Coef" menunjukkan seberapa besar pengaruh perubahan dalam variabel independen terhadap prediksi variabel dependen.

DAFTAR PUSTAKA

[1] S. A. Putri, "Prediksi Cacat Software Dengan Teknik Sampel Dan Seleksi Fitur Pada Bayesian Network," *Jur. Kajian Il.*, vol. 19, no. 1, hlm. 17, Jan 2019, doi: 10.31599/jki.v19i1.314.
[2] M. Banga dan A. Bansal, "Proposed software faults detection using hybrid approach," *Security and Privacy*, vol. 6, no. 4, hlm. e103, Jul 2023, doi: 10.1002/spy2.103.
[3] H. Herfandi, M. T. A. Zaen, Y. Yuliadi, M. Julkarnain, dan F. Hamdani, "Application of Information Gain to Select Attributes in Improving Naïve Bayes Accuracy in Predicting Customer's Payment Capability," *JISA*, vol. 4, no. 2, hlm. 155–163, Des 2021, doi: 10.31326/jisa.v4i2.1044.
[4] N. Ichsan, H. Fatah, E. Ermawati, I. Indriyanti, dan T. Wahyuni, "Integrasi Distribution Based Balance dan Teknik Ensemble Bagging Naive Bayes Untuk Prediksi Cacat Software," *MJI*, vol. 14, no. 2, hlm. 79, Des 2022, doi: 10.35194/mji.v14i2.2623.
[5] N. Hidayati, J. Suntoro, dan G. G. Setiaji, "Perbandingan Algoritma Klasifikasi untuk Prediksi Cacat Software dengan Pendekatan CRISP-DM," *JSI*, vol. 7, no. 2, hlm. 117–126, Nov 2021, doi: 10.34128/jsi.v7i2.313.
[6] A. Muzaki dan A. Witanti, "SENTIMENT ANALYSIS OF THE COMMUNITY IN THE TWITTER TO THE 2020 ELECTION IN PANDEMIC COVID-19 BY METHOD NAIVE BAYES CLASSIFIER," *J. Tek. Inform. (JUTIF)*, vol. 2, no. 2, hlm. 101–107, Mar 2021, doi: 10.20884/1.jutif.2021.2.2.51.
[7] E.- Mutiara, "ALGORITMA KLASIFIKASI NAIVE BAYES BERBASIS PARTICLE SWARM OPTIMIZATION UNTUK PREDIKSI PENYAKIT TUBERCULOSIS (TB)," *SWABUMI*, vol. 8, no. 1, hlm. 46–58, Mar 2020, doi: 10.31294/swabumi.v8i1.7668.
[8] R. Yuliza, "Sistem Pakar Akurasi dalam Mengidentifikasi Penyakit Gingivitis pada Gigi Manusia dengan Metode Naive Bayes," *jsisfotek*, Agu 2022, doi: 10.37034/jsisfotek.v5i1.157.
[9] K. R. Diska dan K. Budayawan, "Sistem Informasi Prediksi Kelulusan Menggunakan Metode Naive Bayes Classifer (Studi Kasus: Prodi Pendidikan Teknik Informatika)," *jptam*, vol. 7, no. 1, hlm. 936–943, Feb 2023, doi: 10.31004/jptam.v7i1.5375.
[10] D. K. Nurilahi, R. Munadi, S. Syahrial, dan A. Bahri, "Penerapan Metode Naïve Bayes pada Honeypot Dionaea dalam Mendeteksi Serangan Port Scanning," *ELKOMIKA*, vol. 10, no. 2, hlm. 309, Apr 2022, doi: 10.26760/elkomika.v10i2.309.
[11] I. N. Yulita, R. Rosadi, S. Purwani, dan R. M. Awangga, "A COMBINATION DEEP BELIEF NETWORKS AND SHALLOW CLASSIFIER FOR SLEEP STAGE CLASSIFICATION," *kursor*, hlm. 197, Okt 2017, doi: 10.28961/kursor.v8i4.97.
[12] W. McKinney, *Python for data analysis: data wrangling with pandas, NumPy, and IPython*, Second edition. Sebastopol, California: O'Reilly Media, Inc, 2018.
[13] Y. Tohma, K. Tokunaga, S. Nagase, dan Y. Murata, "Structural approach to the estimation of the number of residual software faults based on the hyper-geometric distribution," *IIEEE Trans. Software Eng.*, vol. 15, no. 3, hlm. 345–355, Mar 1989, doi: 10.1109/32.21762.
[14] M. D'Ambros, M. Lanza, dan R. Robbes, "An extensive comparison of bug prediction approaches," dalam *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, Cape Town, South Africa: IEEE, Mei 2010, hlm. 31–41. doi: 10.1109/MSR.2010.5463279.
[15] Universitas Sriwijaya, A. Hardoni, dan D. P. Rini, "Integrasi Pendekatan Level Data Pada Logistic Regression Untuk Prediksi Cacat Perangkat Lunak," *JIKO*, vol. 3, no. 2, hlm. 101–106, Agu 2020, doi: 10.33387/jiko.v3i2.1734.
[16] A. Hardoni, "Integrasi SMOTE pada Naive Bayes dan Logistic Regression Berbasis Particle Swarm Optimization untuk Prediksi Cacat Perangkat Lunak," *justin*, vol. 9, no. 2, hlm. 144, Apr 2021, doi: 10.26418/justin.v9i2.43173.
[17] H. Kaur dan A. Kaur, "An empirical study of Aging Related Bug prediction using Cross Project in Cloud Oriented Software," *IJCAI*, vol. 46, no. 8, Nov 2022, doi: 10.31449/inf.v46i8.4197.
[18] P. L. S. T. Sangeetha Yalamanchili, "Software Defect Prediction Using Machine Learning," *IJRTE*, vol. 8, no. 2S11, hlm. 1053–1057, Nov 2019, doi: 10.35940/ijrte.B1178.0982S1119.
[19] R. Agrawal dan R. Goyal, "Developing bug severity prediction models using word2vec," *International Journal of Cognitive Computing in Engineering*, vol. 2, hlm. 104–115, Jun 2021, doi: 10.1016/j.ijcce.2021.08.001.
[20] S. Wang, T. Liu, J. Nam, dan L. Tan, "Deep Semantic Feature Learning for Software Defect Prediction," *IIEEE Trans. Software Eng.*, vol. 46, no. 12, hlm. 1267–1293, Des 2020, doi: 10.1109/TSE.2018.2877612.
[21] B. Turhan dan A. Bener, "Analysis of Naive Bayes' assumptions on software fault data: An empirical study," *Data & Knowledge Engineering*, vol. 68, no. 2, hlm. 278–290, Feb 2009, doi: 10.1016/j.datak.2008.10.005.

[22] N. A. Zaidi, J. Cerquides, M. J. Carman, dan G. I. Webb, "Alleviating Naive Bayes Attribute Independence Assumption by Attribute Weighting," *Journal of Machine Learning Research*, vol. 14, no. 24, hlm. 1947–1988, 2013.
[23] W. Gata *dkk.*, "Algorithm Implementations Naïve Bayes, Random Forest. C4.5 on Online Gaming for Learning Achievement Predictions," dalam *Proceedings of the 2nd International Conference on Research of Educational Administration and Management (ICREAM 2018)*, Bandung, Indonesia: Atlantis Press, 2019. doi: 10.2991/icream-18.2019.1.
[24] S. Khadijah dan P. S. Sasongko, "The Comparison of Imbalanced Data Handling Method in Software Defect Prediction," *KINETIK*, hlm. 203–210, Agu 2020, doi: 10.22219/kinetik.v5i3.1049.
[25] S. Goyal, "Effective software defect prediction using support vector machines (SVMs)," *Int J Syst Assur Eng Manag*, vol. 13, no. 2, hlm. 681–696, Apr 2022, doi: 10.1007/s13198-021-01326-1.
[26] U. S. Bhutamapuram dan R. Sadam, "With-in-project defect prediction using bootstrap aggregation based diverse ensemble learning technique," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 10, hlm. 8675–8691, Nov 2022, doi: 10.1016/j.jksuci.2021.09.010.
[27] S. Goyal, "Handling Class-Imbalance with KNN (Neighbourhood) Under-Sampling for Software Defect Prediction," *Artif Intell Rev*, vol. 55, no. 3, hlm. 2023–2064, Mar 2022, doi: 10.1007/s10462-021-10044-w.
[28] M. J. Hernández-Molinos, A. J. Sánchez-García, R. E. Barrientos-Martínez, J. C. Pérez-Arriaga, dan J. O. Ocharán-Hernández, "Software Defect Prediction with Bayesian Approaches," *Mathematics*, vol. 11, no. 11, hlm. 2524, Mei 2023, doi: 10.3390/math11112524.