TUGAS AKHIR

Penerapan *ICLL* Dalam Mengatasi Ketidakseimbangan Kelas Pada Prediksi Cacat Perangkat Lunak

Oleh:

Muhammad Fadli 2024250015

PROGRAM STUDI INFORMATIKA
FAKULTAS ILMU KOMPUTER DAN REKAYASA
UNIVERSITAS MULTI DATA PALEMBANG
PALEMBANG
2024

Fakultas Ilmu Komputer dan Rekayasa
Universitas Multi Data Palembang

Program Studi Informatika
Tugas Akhir Sarjana Komputer
Semester Genap Tahun 2023/2024

Penerapan ICLL Dalam Mengatasi Ketidakseimbangan Kelas Pada Prediksi Cacat Perangkat Lunak

Muhammad Fadli 2024250015

Abstrak

Ketidakseimbangan *dataset* merupakan masalah yang sering ditemui dalam prediksi cacat perangkat lunak. Salah satu teknik yang umum digunakan untuk mengatasi masalah ini adalah *oversampling* dengan metode *SMOTE*. Namun, *SMOTE* memiliki kelemahan, seperti rentan menghasilkan *data noise* dan meningkatkan risiko *overfitting* akibat penggandaan *data* dalam *dataset*. Selain teknik *oversampling* seperti *SMOTE*, terdapat juga metode yang disebut *ICLL*. *ICLL* mampu mengatasi ketidakseimbangan kelas tanpa melakukan *resampling*, melainkan melalui analisis pengelompokan hirarki dalam dua tahapan. Dengan tidak menggandakan *dataset*, *ICLL* mengurangi kemungkinan terjadinya *overfitting*. Penelitian ini akan membandingkan kinerja antara *SMOTE* dan *ICLL* pada *classifier SVM* dengan *dataset NASA*, yaitu CM1 dan JM1, pembagian *dataset* menggunakan rasio 70% *data* latih dan 30% *data* uji. Hasil dari penelitian ini membuktikan bahwa evaluasi model *SVM* dengan *SMOTE* berhasil mengungguli ICLL pada *dataset* CM1 menggunakan kernel *rbf* dengan akurasi sebesar 70%, *AUC* sebesar 65.59 %, *F1-Score* kelas *Non-Defect* sebesar 81.16%, *F1-Score* kelas *Defect* sebesar 26.23%. *SMOTE* juga mengungguli *ICLL* pada *dataset* JM1 menggunakan kernel *rbf* dengan akurasi sebesar 72.67%, *AUC* sebesar 66.91%, *F1-Score* kelas *Non-Defect* sebesar 81.83%, *F1-Score* kelas *Defect* sebesar 44.87%. Namun *ICLL* mengungguli *SMOTE* pada waktu *training* dan *predict* model, dengan rata-rata 0.018 detik pada dataset CM1, sementara *SMOTE* memiliki rata-rata 0.033 detik. *ICLL* juga mengungguli *SMOTE* pada waktu *training* dan *predict* model dataset JM1 dengan rata-rata 7.74 detik pada, sementara SMOTE memiliki rata-rata 9.72 detik.

Kata kunci: SMOTE; ICLL; SVM; Cacat Perangkat Lunak; Ketidakseimbangan Dataset

BAB 1
PENDAHULUAN

1.1 Latar Belakang Masalah

Perangkat lunak merupakan sebuah bagian dari komputer yang tidak dapat dilihat secara langsung, tetapi dapat dirasakan manfaatnya oleh pengguna komputer. Menurut Nuriani et al. (2022) perangkat lunak memiliki bentuk berupa *data digital* yang tidak dapat dilihat secara fisik, tetapi bisa digunakan dan diambil manfaatnya oleh para pengguna komputer. Menurut Nur (2019) perangkat lunak atau biasa disebut dengan *software* merupakan istilah yang digunakan secara khusus untuk *data* yang diformat, serta disimpan secara *digital*, termasuk juga program komputer, dokumentasi, dan berbagai informasi yang bisa dibaca, atau ditulis oleh komputer. Terdapat beberapa jenis perangkat lunak, diantaranya adalah Sistem Operasi, Aplikasi, ataupun Pemrograman (Sisma, 2023).

Perangkat lunak pertama kali diperkenalkan secara luas pada tahun 1935 (Sisma, 2023). Perangkat lunak memiliki banyak fungsi dan keberadaannya sangatlah penting bagi suatu komputer. Perangkat lunak berfungsi sebagai penghubung antara perangkat keras yang ada pada komputer, perangkat lunak juga berperan sebagai penyedia dari fungsi-fungsi dasar dari komputer, sehingga komputer tersebut dapat berjalan dengan baik (Sisma, 2023)

Banyak sekali perangkat lunak yang bisa dimanfaatkan oleh pengguna komputer. Salah satu contoh perangkat lunak yang memberikan bantuan yang

sangat besar kepada pengguna komputer adalah Microsoft Office (Nurzaman dkk., 2023). Tidak hanya itu, perangkat lunak atau aplikasi yang dikhususkan pada bidang tertentu juga dapat membantu pekerjaan yang terkait dengan bidang tersebut, misalnya pada bidang pendidikan dimana terdapat begitu banyak siswa, sebuah perangkat lunak yang dapat membantu pendidik atau pengajar dalam menghitung nilai setiap siswa sangatlah diperlukan (Widodo dkk., 2019). Perangkat lunak telah banyak membantu manusia dalam berbagai hal. Namun, apabila terdapat *error* atau cacat pada suatu perangkat lunak terutama cacat pada fitur yang sangat penting, maka kerugian yang muncul juga akan sangat besar.

Pada tahun 1983 terdapat kegagalan sistem *software EWS* (*Early Warning System*) yang hampir menyebabkan perang antara Uni Soviet dan Amerika Serikat, 27 September 1983 sistem peringatan milik Uni Soviet menampilkan sebuah peringatan bahwa Amerika telah meluncurkan 5 rudal balistik menuju ke Uni Soviet, beruntungnya saat itu salah seorang petugas Uni Soviet menyadari bahwa peringatan tersebut merupakan sebuah peringatan palsu (Hasan, 2017). Tahun 1996, terjadi sebuah tragedi dimana penerbangan pertama roket Ariane 5 mengalami kegagalan setelah meluncur selama sekitar 37 detik, meledaknya roket Ariane 5 diakibatkan oleh kesalahan perangkat lunak yang gagal mengkonversi *data*, perubahan *format* dari 64 *bit* menjadi 16 *bit* menyebabkan kelebihan kapasitas (*overflow*) dalam perhitungan nilai *variabel* terkait kecepatan *horizontal*, sehingga sistem kontrol penerbangan gagal untuk mengoperasikan roket dengan benar (Ferry, 2023).

Kemudian, pada tahun 2012, perusahaan Knight Capital Group mengalami kerugian sebesar 440 juta USD atau sekitar 4,2 Triliun Rupiah hanya dalam hitungan menit saja akibat dari kesalahan teknis pada aplikasi mereka, *software trading* Knight Capital Group mengirimkan *trade* palsu secara bertubi-tubi selama 45 menit dan mengakibatkan kerugian besar pada saham yang dibeli dengan harga mahal (Detikfinance, 2012). Tahun 2019, terjadi kecelakaan pesawat Max 8 Lion Air dan Ethiopian Airlines akibat dari kegagalan sensor *AOA* (*Angle Of Attack)* yang memberi petunjuk keliru pada *MCAS* (*Maneuvering Characteristic Augmentation System*) sehingga mengakibatkan kecelakaan terjadi (Hakim, 2020).

Tidak hanya itu, masih banyak lagi tragedi lain yang disebabkan karena kegagalan suatu perangkat lunak. Kasus *game* bernama Aliens: Colonial Marines yang dikembangkan oleh Gearbox Software, dimana terdapat *bug* yang tidak diketahui sehingga menurunkan tingkat penjualan permainan. Kasus hancurnya satelit bernama *Mars Climate Orbiter* pada tahun 1998, yang mengakibatkan hilangnya proyek senilai 327.6 juta USD disebabkan oleh perangkat lunak pada satelit yang salah melakukan perhitungan. Kasus *Y2K* yang juga dikenal sebagai *Bug Millenium* yang terjadi pada tahun 2000, akibat salah perhitungan oleh komputer saat kalender berganti dari tahun 1999 menjadi tahun 2000 dan masih banyak kasus lainnya (Suherman dkk., 2018).

Kerugian yang diakibatkan oleh kecacatan perangkat lunak tidak hanya dapat mempengaruhi finansial, tetapi juga dapat merenggut nyawa seseorang. Karena itulah sangat penting sekali untuk melakukan proses pengujian perangkat lunak terlebih dahulu sebelum meluncurkannya. Menurut Suherman et al. (2018)

melaksanakan pengujian perangkat lunak merupakan hal yang wajib dilakukan pada setiap pengembangan perangkat lunak. Terdapat banyak teknik atau metode yang bisa dilakukan untuk menguji perangkat lunak.

Sebelumnya, telah dilakukan riset lapangan untuk bertemu dengan narasumber dari berbagai instansi, masing-masing dari para narasumber memiliki standar atau metodenya tersendiri untuk melakukan pengujian perangkat lunak. Salah seorang narasumber mengatakan bahwa mereka melakukan pengujian menggunakan teknik *Blackbox Testing*, ada juga narasumber yang melakukan pengujian berdasarkan skema yang telah dipersiapkan terlebih dahulu, tidak hanya itu ada juga narasumber yang mengatakan bahwa melakukan pengujian perangkat lunak sesuai dengan standar atau metode dari *Quality Assurance* (*QA*).

Narasumber menjelaskan kepada peneliti, bahwa konsumsi sumber daya baik itu dana maupun waktu untuk pengujian perangkat lunak biasanya akan bergantung pada skala *project* tersebut. Salah seorang narasumber menyatakan bahwa pengujian perangkat lunak paling cepat bisa memakan waktu selama 1 bulan, sementara untuk perangkat lunak yang berskala besar bisa memakan waktu selama 6 bulan hingga 1 tahun. Namun jika skala dari *project* nya tidak terlalu besar atau *project* kecil, bisa juga memakan waktu hanya beberapa hari meskipun tetap disarankan untuk dilakukan selama paling tidak 1 bulan. Selain konsumsi waktu, proses pengujian perangkat lunak sendiri juga cukup memakan biaya, narasumber menginformasikan bahwa dana yang dialokasikan bisa mencapai 5 hingga 10 persen, dan bahkan bisa mencapai 40 persen jika menggunakan metode *QA*.

Menurut Wahono (2015), kecacatan pada perangkat lunak akan memakan biaya paling tinggi, biaya dalam mengatasi perangkat lunak bisa saja meningkat seiring dengan langkah pengembangan perangkat lunak. Selama tahap pemrograman biaya mengidentifikasi serta memperbaiki cacat bisa mencapai 977 USD untuk setiap cacat, dan meningkat menjadi 7.136 USD pada tahap pengujian, bahkan bisa meningkat lagi mencapai 14.102 USD pada tahap pemeliharaan.

Berdasarkan permasalahan yang telah dibahas sebelumnya, bisa diartikan bahwa pengujian perangkat lunak sangatlah penting, karena apabila terdapat kecacatan dari suatu perangkat lunak maka dampak yang diakibatkan bisa menjadi sangat fatal kepada para penggunanya. Namun, proses pengujian perangkat lunak tidak terlepas dari pengorbanan sumber daya dan juga waktu. Karena itulah solusi disarankan adalah membangun sebuah *tool* yang dapat membantu memprediksi sebuah kecacatan pada perangkat lunak.

Terdapat berbagai cara untuk membangun *tool* tersebut, salah satunya adalah dengan menggunakan *dataset* terkait yang disediakan secara umum, dan melakukan klasifikasi untuk menentukan apakah suatu perangkat lunak diprediksi cacat atau tidaknya, tetapi penggunaan *dataset* umum tidaklah maksimal, hal ini dikarenakan *dataset* yang tersedia secara umum terkadang memiliki kendala tersendiri seperti ketidakseimbangan kelas seperti pada *dataset NASA* (*National Aeronautics and Space Administration*) (Hardoni dkk., 2021).

Ketidakseimbangan kelas (*imbalance class*) merupakan suatu kondisi dimana jumlah data pada setiap kelas yang ada pada *dataset* tidak sama banyak dengan rasio yang cukup besar. Menurut Hairani et al. (2020) ketidakseimbangan kelas adalah

keadaan ketika jumlah *instance* kelas minoritas jauh lebih sedikit dibandingkan dengan jumlah *instance* kelas mayoritas. Ketidakseimbangan kelas menjadi salah satu faktor yang dapat memberikan dampak negatif pada hasil kinerja model. Menurut Hardoni et al. (2021) ketidakseimbangan kelas pada *dataset* menjadi salah satu masalah pada *dataset* dan dapat mempengaruhi hasil kinerja model klasifikasi.

Ketidakseimbangan kelas pada *dataset* akan mengakibatkan kelas mayoritas lebih sering diklasifikasikan daripada kelas minoritas. Masalah ketidakseimbangan kelas akan menyebabkan sebuah model klasifikasi menjadi lebih mudah mengklasifikasikan kelas mayoritas dibandingkan dengan kelas minoritas (Hairani dkk., 2020). Ketidakseimbangan kelas akan membuat keputusan dari model *classifier* menjadi keliru saat melakukan klasifikasi karena lebih memilih kelas mayoritas dibandingkan kelas minoritas (Sir & Soepranoto, 2022).

Sebelumnya telah muncul beberapa penelitian terkait prediksi cacat perangkat lunak dengan kasus *dataset* yang tidak seimbang. Prasetyo et al. (2021) melakukan penelitian terkait prediksi cacat perangkat lunak menggunakan *classifier Random Forest* dan *Logistic Regression*, *dataset* yang mereka gunakan berasal dari PROMISE repositori, dan mereka menggunakan *Random Undersampling* untuk menyeimbangkan *dataset* yang tidak seimbang, hingga akhirnya mendapatkan hasil akurasi tertinggi sebesar 95.582% dengan *classifier Random Forest* setelah dilakukan *resampling*.

Kemudian Hardoni et al. (2021) pernah melakukan penelitian terkait dengan prediksi cacat perangkat lunak, uji coba dilakukan pada 8 *dataset NASA*, mereka mengatasi ketidakseimbangan kelas menggunakan teknik *oversampling* yang

disebut sebagai *SMOTE* (*Synthetic Minority Over-sampling Technique*), Hardoni juga mengatasi *noise attribute* pada *dataset* dengan menggunakan teknik PSO (*Particle Swarm Optimization*), kemudian melakukan klasifikasi menggunakan *Naive Bayes* dan *Logistic Regression*, setelah itu mereka mengukur kinerja model dengan menghitung nilai *AUC* (*Area Under Curve*), hasilnya mereka membuktikan bahwa penggunaan *SMOTE* dan *PSO* dapat meningkatkan hasil kinerja model, serta mereka mendapatkan nilai *AUC* tertinggi pada *classifier Logistic Regression* sebesar 0,89.

Sementara itu Putri & Wahono (2015) melakukan penelitian untuk memprediksi cacat pada perangkat lunak, 9 *dataset* yang mereka gunakan berasal dari *NASA*, dan untuk mengatasi ketidakseimbangan kelas pada *dataset NASA* mereka menggunakan *SMOTE*, kemudian mereka juga menggunakan teknik yang disebut sebagai *Information Gain* untuk memilih atribut demi mencegah kemungkinan dari *noise attribute*, dengan *classifier Naive Bayes*, mereka membuktikan bahwa kombinasi dari *SMOTE* dan *Information Gain* yang diterapkan pada *Naive Bayes* yang memiliki rata-rata *AUC* sebesar 0.789 berhasil mengungguli rata-rata *AUC* model *Naive Bayes* yang bernilai 0.753.

Selain itu, ada juga penelitian yang dilakukan oleh Alhija et al (2022) yang meneliti prediksi cacat perangkat lunak dengan menggunakan *Support Vector Machine* (*SVM*) dan berbagai *dataset* dari PROMISE, penelitian yang mereka lakukan adalah melakukan uji coba untuk mengukur kinerja dari berbagai kernel dari *SVM* dalam memprediksi cacat perangkat lunak pada setiap *dataset* yang digunakan, penelitian mereka membuktikan bahwa kernel *Quadratic* memberikan

kinerja yang lebih baik pada *dataset* CM1, kernel *RBF* untuk KC1, kernel *Cubic* untuk PC1, dan kernel *Gaussian* untuk JM1.

Dalam mengatasi masalah ketidakseimbangan kelas pada *dataset*, penggunaan teknik *resampling* seperti *SMOTE* sudah sering digunakan. Teknik *resampling* dibagi menjadi 3 jenis yaitu, *undersampling*, *oversampling* ataupun kombinasi dari *undersampling* dan *oversampling* (Sir & Soepranoto, 2022). Teknik *resampling* memang lebih sederhana dan mudah untuk dipadukan kepada teknik algoritma pembelajaran yang lain, tetapi teknik *resampling* memiliki keterbatasannya sendiri. *Undersampling* bisa saja mengakibatkan hilangnya informasi penting pada *dataset*, sementara *oversampling* bisa meningkatkan peluang terjadinya *overfitting* (Cerqueira dkk., 2023).

SMOTE termasuk kategori teknik *oversampling*. *SMOTE* merupakan teknik yang meningkatkan jumlah *data* pada kelas minoritas secara acak agar menyamai jumlah *data* pada kelas mayoritas sehingga jumlah *data* pada setiap kelas pada *dataset* menjadi seimbang. *SMOTE* adalah teknik penyeimbang jumlah *data* pada kelas minoritas dengan melakukan seleksi pada *data* tersebut sehingga jumlahnya seimbang dengan jumlah *data* pada kelas mayoritas (Kasanah dkk., 2019). Teknik *SMOTE* memang populer untuk diterapkan dalam mengatasi ketidakseimbangan kelas (Sulistiyowati & Jajuli, 2020), tetapi penggunaan teknik *SMOTE* rentan menghasilkan *data noise* karena tidak membedakan area kelas yang tumpang tindih akibat dari pemilihan *instance* kelas minoritas secara acak untuk di-*oversample* menggunakan *uniform probability* (Hairani dkk., 2020). Penggunaan SMOTE juga

memungkinkan terjadinya *overfitting* akibat dari kemungkinan adanya *data* latih yang sama pada kelas minoritas (Kasanah dkk., 2019).

Penelitian ini akan mencoba untuk membangun sebuah *tool* yang dapat membantu memprediksi cacat perangkat lunak, *dataset* yang digunakan juga akan diambil dari *NASA*. Berdasarkan beberapa penelitian sebelumnya yang telah dibahas, *dataset NASA* memiliki kendala seperti ketidakseimbangan kelas, kelas dalam hal ini adalah klasifikasi pada *dataset*. Namun, penelitian ini tidak akan mengatasi ketidakseimbangan kelas *dataset* menggunakan teknik *resampling* seperti *SMOTE* melainkan teknik yang disebut sebagai *ICLL* (*Imbalanced Classification via Layered Learning*).

ICLL merupakan sebuah metode yang mampu mengatasi masalah ketidakseimbangan kelas tanpa melakukan *resampling*, *ICLL* menggunakan analisis pengelompokan hirarki dalam 2 tahapan, tahap pertama *ICLL* adalah memisahkan *data* yang sudah jelas dari *data* yang berada pada garis batas kelas. Kemudian pada tahap kedua *ICLL* akan membuang atau mengabaikan *data* jelas dari kelas mayoritas yang telah dipisahkan sebelumnya dan membangun model klasifikasi menggunakan *data* yang tersisa. Masing-masing tahapan ini akan mengurangi masalah ketidakseimbangan kelas sehingga pemodelan akan menjadi lebih sederhana (Cerqueira dkk., 2023).

Tujuan utama dari penelitian ini adalah untuk membandingkan efisiensi dan keakuratan *classifier Support Vector Machine* (*SVM*) dalam memprediksi cacat perangkat lunak, setelah ketidakseimbangan distribusi kelas *data* pada *dataset* diatasi menggunakan metode *ICLL* dan *SMOTE*. Selain penggunaan teknik *ICLL*

dalam mengatasi ketidakseimbangan kelas pada *dataset* dan *SVM* sebagai *classifier*, penelitian ini akan menggunakan beberapa teknik tambahan *seperti Min-Max* sebagai teknik normalisasi, serta metode *SelectKBest* yang menggunakan statistik pengukuran *chi-square* sebagai teknik *feature selection*. Sementara pengukuran kinerja model pada penelitian ini akan menggunakan *f1-score* melalui *confusion matrix* pada setiap hasil model, serta menghitung waktu pelatihan yang dibutuhkan oleh model untuk membandingkan efisiensi antara *SMOTE* dan *ICLL*.

Teknik normalisasi *Min-Max* merupakan sebuah langkah yang dilakukan untuk membuat setiap *attribute* pada masing-masing *data* memiliki rentang yang sama, demi mencegah suatu *attribute* mendominasi *attribute* lainnya. *Min-Max* merupakan salah satu metode normalisasi yang melakukan transformasi *linear* kepada *data* asli dan membuatnya memiliki nilai yang setara antara *data* sebelum dan sesudah proses (Henderi dkk., 2021). *Min-Max normalization* adalah metode yang menggunakan nilai minimum dan maksimum dalam melakukan transformasi *linear* untuk membuat *data* berada pada rentang yang sama dan menghasilkan keseimbangan antar *data* (Suryanegara dkk., 2021).

Selain itu, ada juga penggunaan teknik *feature selection* yang disebut sebagai *SelectKBest* pada penelitian ini. *SelectKBest* merupakan teknik yang memilih sejumlah fitur yang memiliki nilai terbaik serta pengaruh terbesar terhadap model klasifikasi. *SelectKBest* adalah salah satu teknik yang tersedia pada paket *sci-kit learn* dari *python*, *SelectKBest* merupakan metode yang menilai fitur-fitur pada *dataset* dan mempertahankan beberapa fitur yang memiliki nilai teratas (Kumar dkk., 2021). Statistik pengukuran yang digunakan dalam penggunaan SelectKBest

adalah *chi-square*. *Chi-square* bekerja dengan cara menghitung bobot dari masing-masing fitur dan mengurutkannya berdasarkan tingkat kepentingan fitur tersebut. (Almutiri & Saeed, 2019)

Kemudian *confusion matrix* merupakan metode untuk mengukur performa model pada permasalahan klasifikasi. *Confusion matrix* adalah alat yang dapat digunakan untuk mengukur kinerja suatu model klasifikasi untuk keluaran dua kelas ataupun lebih (Steffany dkk., 2023). Penelitian ini akan memanfaatkan metode *confusion matrix* untuk mendapatkan hasil kinerja dari *classifier* pada percobaan *SMOTE* dan juga *ICLL*, kemudian hasil dari kinerja tersebut akan dihitung kembali untuk mendapatkan nilai *f1-score* dari setiap percobaan untuk menjadi pembanding kinerja dari masing-masing percobaan.

Pengukuran kinerja model menggunakan *f1-score* dilakukan karena penelitian ini menggunakan *dataset* yang tidak seimbang jumlah *data* kelasnya. Penggunaan *f1-score* sangat baik untuk mengukur kinerja model dengan *dataset* yang tidak seimbang, hal ini disebabkan karena *f1-score* akan mengkombinasikan *recall* dan *precision* untuk memberikan metrik yang efektif dalam mencari kembali informasi pada *data* yang mengandung ketidakseimbangan (Kasanah dkk., 2019). Selain pengukuran kinerja model, penelitian ini juga akan menghitung lamanya waktu pelatihan dari *SVM* pada klasifikasi menggunakan *SMOTE* dan klasifikasi yang menggunakan *ICLL*, untuk membandingkan efisiensi dari kedua teknik tersebut.

1.2 Rumusan Masalah

Berdasarkan latar belakang permasalahan yang telah dibahas sebelumnya, dapat disimpulkan bahwa proses pengujian perangkat lunak tidak luput dari konsumsi dana maupun waktu, sehingga akan sangat membantu sekali jika terdapat sebuah alat yang dapat membantu untuk memprediksi kecacatan pada suatu perangkat lunak.

Membangun sebuah alat pembantu prediksi kecacatan pada suatu perangkat lunak juga bukanlah hal yang mudah, salah satu masalah yang harus diperhatikan adalah adanya kondisi ketidakseimbangan kelas pada *dataset* yang digunakan untuk membangun model, walaupun teknik penyeimbang *dataset* seperti *SMOTE* sudah sering digunakan, tetapi *SMOTE* sendiri memiliki kelemahan. Karena itulah penelitian ini akan mencoba untuk mengatasi masalah ketidakseimbangan *dataset* dengan menggunakan teknik *ICLL*.

1.3 Analisis Terhadap Batasan

1.3.1 Analisis dari Aspek Ekonomis

Dalam rangka menganalisis aspek ekonomis, lima narasumber yang merupakan individu berpengalaman di bidang perangkat lunak telah ditemui, yaitu pemilik dari Smart Integrated Systems, kepala UPT SI Universitas Multi Data Palembang, *web develover* dari *startup* TaniYuk, CEO dari Phylot, dan direktur dari CV Kito Group.

Berdasarkan hasil wawancara yang dilakukan kepada para narasumber, narasumber memberikan respon yang positif dan mengatakan bahwa *tool* yang

mampu memprediksi cacat perangkat lunak akan membantu dalam pekerjaan mereka. Para narasumber telah memberikan masukan bahwa *tool* ini bisa saja diberikan harga hingga Rp. 500.000 hingga Rp. 1.000.000 per bulannya, dan juga baiknya jika terdapat uji coba gratis untuk pengujian versi beta.

1.3.2 Analisis dari Aspek Manufakturabilitas

Tabel 1.1 berikut ini akan memperlihatkan hasil dari analisis terhadap aspek manufakturabilitas untuk penelitian ini.

Tabel 1.1 Analisis Aspek Manufakturabilitas dalam Sudut Pandang Pengguna.

Aspek	Smart Integrated Systems	UPT SI MDP	Startup TaniYuk	Phylot	CV Kito Group
Kemudahan mendeteksi kecacatan perangkat lunak (1 bulan)	OK	OK	OK	OK	OK
Kemampuan mendeteksi cacat perangkat lunak (1 bulan)	OK	OK	OK	OK	OK
Mendeteksi kecacatan perangkat lunak berdasarkan input angka (1 bulan)	Bagaimana cara mendapatkan angkanya?	Angka apa yang dimaksudkan ini?	Apakah tidak bisa menginputkan aplikasinya saja atau projectnya?	Bagaimana project yang dikerjakan bisa terhubung dengan angka-	Mungkin lebih sederhana jika bisa langsung menginputkan aplikasinya saja.

				angka ini?	
Total: 3 bulan					

1.3.3 Analisis dari Aspek Sustainibilitas

Tabel 1.2 berikut ini akan memperlihatkan hasil dari analisis terhadap aspek sustainibilitas untuk penelitian ini.

Tabel 1.2 Analisis Aspek Sustainibilitas dalam Sudut Pandang Pengguna

Aspek	Smart Integrated Systems	UPT SI MDP	Startup TaniYuk	Phylot	CV Kito Group
Tool mampu memprediksi kecacatan pada perangkat lunak dalam 1 detik	OK	OK	OK	OK	OK

1.4 Analisis Terhadap Karakteristik Solusi

Berikut ini Tabel 1.3 yang akan memperlihatkan hasil dari analisis terhadap karakteristik solusi untuk penelitian ini.

Tabel 1.3 Analisis Karakteristik Solusi

No.	Masalah	Fungsi
1.	Proses pengujian perangkat lunak tidak luput dari konsumsi sumber daya dan waktu.	Mengurangi konsumsi sumber daya dan pemakaian waktu untuk memprediksi cacat perangkat lunak.
2.	Faktor sumber daya manusia yang dapat mempengaruhi proses pengujian perangkat lunak.	Kinerja AI tidak sebaik kinerja manusia, tetapi bisa menjadi alternatif yang lebih baik, jika terdapat hambatan dari faktor sumber daya manusia.
3.	Kecacatan yang tidak sengaja terlewat sehingga memberikan dampak fatal pada penggunanya.	Bagaikan lapis kedua, AI prediksi cacat perangkat lunak bisa menjadi alat untuk memastikan kembali kecacatan yang terdapat pada perangkat lunak.

1.5 Pemilihan Solusi

Masalah ketidakseimbangan *dataset* seringkali ditemui oleh para peneliti, terutama pada kasus deteksi cacat perangkat lunak. Seperti halnya pada penelitian Prasetyo et al. (2021) yang telah dibahas sebelumnya di latar belakang masalah, dimana mereka menggunakan *dataset* yang berasal dari PROMISE repositori, dan menggunakan *Random Undersampling* untuk menyeimbangkan dataset yang tidak seimbang. Hardoni et al. (2021) yang melakukan penelitian menggunakan *dataset NASA* dan *SMOTE* untuk mengatasi masalah ketidakseimbangan kelas pada *dataset*. Putri & Wahono (2015) yang juga melakukan penelitian menggunakan *dataset NASA* dan teknik *SMOTE* untuk menyeimbangkan kelas pada *dataset*.

Sudah cukup umum bahwa teknik *resampling* seperti *SMOTE* digunakan untuk mengatasi ketidakseimbangan kelas pada *dataset*. Namun, teknik *SMOTE* memiliki kemungkinan untuk menghasilkan *overfitting* (Kasanah dkk., 2019), teknik *SMOTE* juga bisa saja meningkatkan beratnya biaya komputasi dan lamanya waktu pelatihan model. Karena itulah solusi yang ditawarkan pada penelitian ini adalah mengatasi ketidakseimbangan *dataset* menggunakan teknik *ICLL*.

Seperti yang telah dibahas melalui latar belakang masalah sebelumnya, *ICLL* mampu mengatasi ketidakseimbangan kelas tanpa *resampling*, dengan menggunakan analisis pengelompokan hirarki dalam 2 tahapan, dimana tahap pertama *ICLL* adalah memisahkan *data* yang sudah jelas dari *data* yang berada pada garis batas kelas. Kemudian pada tahap kedua *ICLL* akan membuang *data* jelas dari kelas mayoritas yang telah dipisahkan sebelumnya dan membangun model klasifikasi menggunakan *data* yang tersisa (Cerqueira dkk., 2023).

1.6 Skenario Pemanfaatan Produk oleh Pengguna

Tool yang akan dibuat merupakan sebuah *form* menggunakan bahasa *python* dengan memanfaatkan *library gradio* dan *Jupyter Notebook* sebagai *IDE*. Ketika *tool* ini dijalankan, pengguna akan diminta untuk memasukan beberapa *data* angka pada *form* yang disediakan, dan kemudian *tool* akan memprediksi kecacatan perangkat lunak berdasarkan angka yang dimasukan oleh pengguna.

1.7 Tujuan

Berikut merupakan beberapa tujuan dari pembuatan *Capstone Project* ini.

1. Membangun sebuah *tool* atau alat yang dapat membantu untuk memprediksi kecacatan pada perangkat lunak.
2. Memperkenalkan *ICLL* sebagai salah satu metode yang dapat membantu dalam mengatasi ketidakseimbangan distribusi kelas *data* pada *dataset*.
3. Membandingkan efisiensi dan keakuratan *classifier Support Vector Machine* (SVM) dalam memprediksi cacat perangkat lunak, setelah ketidakseimbangan distribusi kelas *data* pada *dataset* diatasi menggunakan metode *ICLL* dan *SMOTE*.

REFERENSI

Alfonsius, E., Sukardi, & Astawa, I. M. N. V. (2023). Sistem Informasi Pelaporan Pekerjaan Proyek Berbasis SDLC Modelling (Studi Kasus: PT Vertikal Tiara Manunggal). *Journal of Artificial Intelligence And Technology Information (JAITI)*, *1*(2), 50–58.

Alhija, H. A., Azzeh, M., & Almasalha, F. (2022). Software Defect Prediction Using Support Vector Machine. *arXiv preprint arXiv:2209.14299*. https://doi.org/0.6977/IJoSI.202206_7(2).0003

Almutiri, T., & Saeed, F. (2019). Chi Square and Support Vector Machine with Recursive Feature Elimination for Gene Expression Data Classification. *2019 1st International Conference of Intelligent Computing and Engineering: Toward Intelligent Solutions for Developing and Empowering our Societies, ICOICE 2019*. https://doi.org/10.1109/ICOICE48418.2019.9035165

Andriaskiton, M., & Fahdian, E. (2022). Analisis Pengaruh Pengetahuan Kewirausahaan Dan Motivasi Usaha Terhadap Keberhasilan Usaha (Studi Kasus Pada Pedagang Jalan Selat Panjang Medan). *Management Studies and Entrepreneurship Journal*, *3*(6), 3906–3914.

Bellinger, C., Branco, P., & Torgo, L. (2019). The CURE for class imbalance. *Discovery Science: 22nd International Conference, DS 2019, Split, Croatia, October 28–30, 2019, Proceedings 22*, 3–17.

Butler, C. W. (2021). Metric Based Evaluation and Improvement of Software Designs. *Journal of Software Engineering and Applications*, *14*(8), 389–399. https://doi.org/10.4236/jsea.2021.148023

Butler, C. W., & McCabe, T. J. (2021). Cyclomatic Complexity-Based Encapsulation, Data Hiding, and Separation of Concerns. *Journal of Software Engineering and Applications*, *14*(1), 44–66. https://doi.org/10.4236/jsea.2021.141004

Cerqueira, V., Torgo, L., Branco, P., & Bellinger, C. (2023). Automated Imbalanced Classification via Layered Learning. *Machine Learning*, *112*(6), 2083–2104. https://doi.org/10.1007/s10994-022-06282-w

Detikfinance. (2012). *Rp 4,2 Triliun Lenyap Gara-gara Trading Error, Knight Capital "Sekarat."* detikfinance. https://finance.detik.com/bursa-dan-valas/d-1982623/rp-42-triliun-lenyap-gara-gara-trading-error-knight-capital-sekarat

Ferry, M. (2023). *Catatan Sejarah 4 Juni: Penerbangan Pertama Ariane 5 Meledak.* bertuahpos.com. https://bertuahpos.com/insight/catatan-sejarah-4-juni-penerbangan-pertama-ariane-5-meledak.html

Gurning, U. R., Octavia, S. F., Andriyani, D. R., Nurainun, & Permana, I. (2024). Prediksi Risiko Stunting pada Keluarga Menggunakan Naïve Bayes Classifier dan Chi-Square. *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, *4*(1), 172–180. https://doi.org/10.57152/malcom.v4i1.1074

Hairani, Saputro, K. E., & Fadli, S. (2020). K-means-SMOTE untuk menangani ketidakseimbangan kelas dalam klasifikasi penyakit diabetes dengan C4.5, SVM, dan naive Bayes. *Jurnal Teknologi dan Sistem Komputer*, *8*(2), 89–93. https://doi.org/10.14710/jtsiskom.8.2.2020.89-93

Hakim, C. (2020). *Boeing 737 Max 8 dan Penjelasan Penyebab Jatuhnya Lion Air JT-610 dan Ethiopian Airlines Flight 302*. KOMPAS.com. https://www.kompas.com/tren/read/2020/11/28/174929865/boeing-737-max-8-dan-penjelasan-penyebab-jatuhnya-lion-air-jt-610-dan

Hardoni, A., Rini, D. P., & Sukemi. (2021). Integrasi SMOTE pada Naive Bayes dan Logistic Regression Berbasis Particle Swarm Optimization untuk Prediksi Cacat Perangkat Lunak. *Jurnal Media Informatika Budidarma*, *5*(1), 233–241. https://doi.org/10.30865/mib.v5i1.2616

Hasan, A. M. (2017). *Perang Dunia III antara AS dan Soviet Gagal Karena Petrov*. tirto.id. https://tirto.id/perang-dunia-iii-antara-as-dan-soviet-gagal-karena-petrov-cwZ5

Henderi, Wahyuningsih, T., & Rahwanto, E. (2021). Comparison of Min-Max normalization and Z-Score Normalization in the K-nearest neighbor (kNN) Algorithm to Test the Accuracy of Types of Breast Cancer. *IJIIS: International Journal of Informatics and Information Systems*, *4*(1), 13–20. https://doi.org/10.47738/ijiis.v4i1.73

Istikhomah, N., Rohaetin, S., & Barbara, B. (2023). Pengaruh Pembelajaran Ekonomi Terhadap Minat Berinvestasi Siswa Sma Negeri 1 Pangkalan Bun. *JURNAL KEGURUAN DAN ILMU PENDIDIKAN*, *1*(4), 304–313. https://doi.org/10.61116/jkip.v1i4.200

Kasanah, A. N., Muladi, & Pujianto, U. (2019). Penerapan Teknik SMOTE untuk Mengatasi Imbalance Class dalam Klasifikasi Objektivitas Berita Online Menggunakan Algoritma KNN. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, *3*(2), 196–201. https://doi.org/10.29207/resti.v3i2.945

Kumar, P., Praveen, S., Maheshwari, R., & Gowda, S. D. (2021). *A Comparative Study of Machine Learning Techniques in Heart Disease Detection*. *4*(10), 264–272. https://doi.org/10.5281/zenodo.4515583

Maina, N. K., Muketha, G. M., & Wambugu, G. M. (2022). A Literature Survey of Complexity Metrics for Object-Oriented Programs. *International Journal of Science and Engineering Applications*, *11*(5), 66–71. https://doi.org/10.7753/IJSEA1105.1003

Nugraha, W., Risdiansyah, D., Purwaningtias, D., Hidayatulloh, T., & Suhada, S. (2022). Kombinasi Tomek-Link Dan Smote Untuk Mengatasi Ketidakseimbangan Kelas Pada Credit Card Fraud. *Jurnal Larik: Ladang Artikel Ilmu Komputer*, *2*(2), 32–40. https://doi.org/10.31294/larik.v2i2.1789

Nur, M. (2019). *Tutorial Instalasi Software* (Vol. 11). MiftaChun Nur.

Nuriani, Abdurahman, D., Nugroho, A., Aziz, H. S. Al, Yosua, A., Hasibuan, M. S., Adha, F. F., Alex, Firmansyah, R., & Noer, F. I. (2022). Pengenalan Software Dan Hardware Komputer Kepada Siswa Madrasah Tsanawiyah Raudhatussa'Adah. *Abdi Jurnal Publikasi*, *1*(2), 80–84. https://jurnal.portalpublikasi.id/index.php/AJP/index

Nurzaman, F., Yuliani, N., Suwartane, I. G. A., Gustina, D., Basry, A., & Marnis. (2023). Peningkatan Belajar Perangkat Lunak Microsoft Office untuk Siswa SMP dan Pemanfaatan Teknologi Internet Sebagai Media Belajar. *Media Abdimas*, *3*(2), 8–14. https://doi.org/10.37817/mediaabdimas.v3i2.2759

Parlika, R., Wijaya, D. C. M., Khariono, H., & Fernanda, R. A. (2020). Studi literatur perbandingan antara metode LOC, COCOMO, FPA dalam ranah software metric. *Jurnal Pendidikan Informatika dan Sains*, *9*(1), 66–74. https://doi.org/10.31571/saintek.v9i1.1697

Pranatawijaya, V. H., Widiatry, W., Priskila, R., & Putra, P. B. A. A. (2019). Penerapan skala Likert dan skala dikotomi pada kuesioner online. *Jurnal Sains Dan Informatika*, *5*(2), 128–137.

Prasetyo, R., Nawawi, I., Fauzi, A., & Ginabila. (2021). Komparasi Algoritma Logistic Regression dan Random Forest pada Prediksi Cacat Software. *Jurnal Teknik Informatika UNIKA Santo Thomas*, *06*(2), 275–281. https://doi.org/10.54367/jtiust.v6i2.1522

Putri, S. A., & Wahono, R. S. (2015). Integrasi SMOTE dan Information Gain pada Naive Bayes untuk Prediksi Cacat Software. *Journal of Software Engineering*, *1*(2), 86–91.

Ramdani, M., Faclah, & Saifudin, A. (2023). Pengujian Sistem Pemberkasan Pada PT Flexofast Menggunakan Metode Black Box. *Jurnal Manajemen, Ekonomi, Hukum, Kewirausahaan, Kesehatan, Pendidikan dan Informatika (MANEKIN)*, *1*(4), 219–224.

Sanusi, B. A., Olabiyisi, S. O., Afolabi, A. O., & Olowoye, A. O. (2020). Development of an Enhanced Automated Software Complexity Measurement System. *Journal of Advances in Computational Intelligence Theory*, *1*(3), 1–11. https://doi.org/10.5281/zenodo.3597630

Sir, Y. A., & Soepranoto, A. H. H. (2022). Pendekatan Resampling Data Untuk Menangani Masalah Ketidakseimbangan Kelas. *Jurnal Komputer dan Informatika*, *10*(1), 31–38. https://doi.org/10.35508/jicon.v10i1.6554

Sisma, A. F. (2023). *Perangkat Lunak Komputer, Pengertian, Fungsi, dan Jenisnya*. katadata.co.id. https://katadata.co.id/agung/lifestyle/64c11f6c69262/perangkat-lunak-komputer-pengertian-fungsi-dan-jenisnya

Steffany, Purnajaya, A. R., Jelita, R., Tesvara, E., Nestelrody, M., & Irwansyah, J. (2023). Penerapan Metode Radial Basis Function (RBF) dalam Mengklasifikasikan Penyakit Demam Berdarah. *Journal of Digital Ecosystem for Natural Sustainability (JoDENS)*, *3*(1), 1–4.

Suherman, I. C., Effendy, G. Y., Adianto, A. I., Handayani, F., Wujdi, M. A., & Aini, N. (2018). Pentingnya Menelaah Macam-Macam Kesalahan Rekayasa Perangkat Lunak. *Journal Information Engineering and Educational Technology*, *2*(2), 60–64. https://doi.org/10.26740/jieet.v2n2.p60-64

Sulistiyowati, N., & Jajuli, M. (2020). Integrasi Naive Bayes Dengan Teknik Sampling Smote Untuk Menangani Data Tidak Seimbang. *Nuansa Informatika*, *14*(1), 34–37. https://doi.org/10.25134/nuansa.v14i1.2411

Suryanegara, G. A. B., Adiwijaya, & Purbolaksono, M. D. (2021). Peningkatan Hasil Klasifikasi pada Algoritma Random Forest untuk Deteksi Pasien Penderita Diabetes Menggunakan Metode Normalisasi. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, *5*(1), 114–122. https://doi.org/10.29207/resti.v5i1.2880

Wahono, R. S. (2015). A Systematic Literature Review of Software Defect Prediction: Research Trends, Datasets, Methods and Frameworks. *Journal of Software Engineering*, *1*(1), 1–16. https://doi.org/10.21067/jpm.v4i2.3636

Widodo, R. B., Subianto, M., & Imelda, G. (2019). Peningkatan Efisiensi Kerja Guru Melalui Pembuatan Aplikasi Rapor Berbasis Komputer. *JPM (Jurnal Pemberdayaan Masyarakat)*, *4*(2), 363–370. https://doi.org/10.21067/jpm.v4i2.3636