JUKOMIKA (Jurnal Ilmu Komputer dan Informatika)
Volume 9, Nomor 1, July 2026 || Hal 24-32
***p-ISSN* : 2655-7541 ; e-ISSN : 2745-5823**
DOI : 10.54650

Prediksi Tingkat Pendidikan Arsitek Perangkat Lunak Menggunakan Pendekatan *Machine Learning Classification*

Seni Agustira*[1], Hafiyyan Putra Pratama[2]
Universitas Pendidikan Indonesia, Indonesia[1,2]
seniagustira21@upi.edu[1], hafiyyan@upi.edu[2]
*Corresponding author : seniagustira21@upi.edu[1]

Abstrak— Penelitian ini membangun model klasifikasi untuk memprediksi tingkat pendidikan akhir arsitek perangkat lunak berdasarkan pola penggunaan gaya arsitektur dalam proyek nyata. *Dataset* yang digunakan adalah *Dataset of Software Architectural Styles* dari *Kaggle*, yang terdiri dari 1.002 responden dengan 18 fitur numerik yang mencerminkan frekuensi penggunaan 18 gaya arsitektur perangkat lunak. Variabel target bersifat tiga kelas: BSC (CS or SE), MS (CS or SE), dan PhD (CS or SE). Tantangan utama pada dataset ini adalah ketidakseimbangan kelas yang ekstrem, dengan proporsi BSC mencapai 77,4%. Praproses data mencakup penanganan *outlier* menggunakan metode IQR, *encoding label*, normalisasi dengan *StandardScaler*, dan *stratified split* 80:20. Penelitian ini membandingkan empat pendekatan penanganan ketidakseimbangan kelas, yaitu *class_weight*, SMOTE, *Borderline*-SMOTE, dan ADASYN, pada dua algoritma klasifikasi: *Logistic Regression* sebagai *baseline* dan *Random Forest* dengan hyperparameter tuning sebagai model ensemble. Analisis *Variance Inflation Factor* (VIF) dilakukan untuk memverifikasi asumsi multikolinearitas. Hasil eksperimen menunjukkan bahwa *Random Forest* dengan SMOTE memperoleh akurasi terbaik sebesar 78,1% dengan MCC 0,312, AUC 0,914, dan Macro F1 0,487. Analisis feature importance mengungkapkan bahwa gaya arsitektur *Blackboard*, *Data-Centric*, dan *Layered* memiliki kontribusi terbesar dalam membedakan tingkat pendidikan. Penelitian ini menyimpulkan bahwa kombinasi *Random Forest* dengan teknik *oversampling* lebih tepat untuk permasalahan klasifikasi non-linear dengan kelas tidak seimbang.

Abstract— *This study builds a classification model to predict the final education level of software architects based on their usage patterns of software architectural styles in real projects. The dataset used is the Dataset of Software Architectural Styles from Kaggle, consisting of 1,002 respondents with 18 numerical features. The target variable has three classes: BSC (CS or SE), MS (CS or SE), and PhD (CS or SE). The main challenge is the severely imbalanced class distribution, with BSC comprising 77.4% of samples. This study compares four class-imbalance handling approaches, namely class_weight, SMOTE, Borderline-SMOTE, and ADASYN, applied to two classifiers: Logistic Regression as a baseline and tuned Random Forest as an ensemble model. Variance Inflation Factor (VIF) analysis was conducted to verify multicollinearity assumptions. Results show that Random Forest with SMOTE achieves the best performance with 78.1% accuracy, MCC of 0.312, AUC of 0.914, and Macro F1 of 0.487. Feature importance analysis reveals that Blackboard, Data-Centric, and Layered architectural styles contribute most to distinguishing education levels. This study concludes that combining Random Forest with oversampling techniques is more appropriate for non-linear classification with imbalanced classes.*

Keywords— *Random Forest, Logistic Regression, Software Architecture, Class Imbalance, Feature Importance, Educational Level Prediction*

1. Pendahuluan

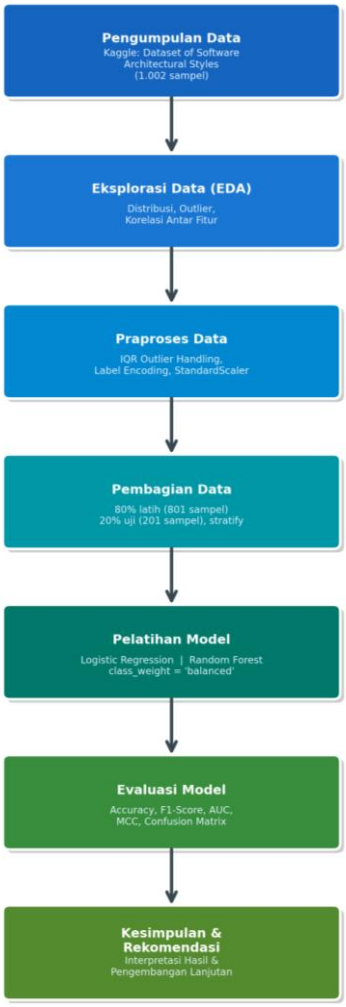
Arsitek perangkat lunak memiliki peran penting dalam menentukan struktur dan kualitas sistem perangkat lunak modern. Keputusan arsitektur yang diambil pada tahap awal pengembangan berpengaruh langsung terhadap maintainability, scalability, reliability, dan evolusi sistem dalam jangka panjang [1]. Di sisi lain, perkembangan teknologi machine learning membuka peluang untuk memahami pola perilaku profesional di bidang rekayasa perangkat lunak melalui analisis data. Dengan memanfaatkan data survei mengenai penggunaan gaya arsitektur perangkat lunak, dimungkinkan untuk membangun model prediktif yang dapat mengidentifikasi keterkaitan antara pola penggunaan arsitektur dengan tingkat pendidikan seseorang[2]. Hubungan tersebut memiliki implikasi penting bagi institusi pendidikan dalam merancang kurikulum yang relevan, sekaligus bagi industri dalam proses rekrutmen dan pengembangan talenta. Dalam konteks arsitek perangkat lunak, terdapat hipotesis bahwa latar belakang pendidikan seseorang dapat tercermin dari cara mereka memilih dan menerapkan gaya arsitektur pada proyek nyata.

JUKOMIKA (Jurnal Ilmu Komputer dan Informatika)
Volume 9, Nomor 1, July 2026 || Hal 24-32
***p-ISSN* : 2655-7541 ; e-ISSN : 2745-5823**
DOI : 10.54650

Penelitian ini memanfaatkan Dataset of Software Architectural Styles yang tersedia di Kaggle [3], yang berisi data survei dari 1.002 arsitek perangkat lunak mengenai frekuensi penggunaan 18 gaya arsitektur berbeda. Target prediksi dalam penelitian ini adalah tingkat pendidikan akhir responden yang terdiri dari tiga kelas, yaitu BSC (CS or SE), MS (CS or SE), dan PhD (CS or SE). Karena target bersifat kategorikal, maka pendekatan yang digunakan adalah klasifikasi. Namun, dataset ini memiliki tantangan utama berupa ketidakseimbangan kelas yang cukup ekstrem, di mana kelas BSC mendominasi sebesar 77,4%, diikuti MS sebesar 17,3%, dan PhD hanya sebesar 5,3%. Ketidakseimbangan tersebut berpotensi menyebabkan model lebih berfokus pada kelas mayoritas dan mengabaikan kelas minoritas, sehingga dapat memengaruhi performa klasifikasi yang dihasilkan.

2. Metodologi Penelitian

Penelitian ini dilaksanakan melalui delapan tahapan utama yang saling berkaitan secara sistematis, sebagaimana ditunjukkan pada gambar 1.



Gambar 1. Metodologi Penelitian

Penelitian ini dilakukan melalui beberapa tahapan sistematis yang meliputi pengumpulan data, eksplorasi data, preprocessing, pelatihan model, evaluasi, hingga interpretasi hasil. Dataset yang digunakan adalah *Dataset of Software Architectural Styles* yang diperoleh dari Kaggle [3]. Dataset terdiri dari 1.002 data responden dengan 18 fitur numerik yang merepresentasikan frekuensi penggunaan gaya arsitektur perangkat lunak, serta variabel target berupa tingkat pendidikan akhir responden. Tahap awal dilakukan melalui *Exploratory Data Analysis* (EDA) untuk memahami distribusi data, mendeteksi *outlier*, serta melihat hubungan antar fitur menggunakan histogram, boxplot, dan matriks korelasi [4]. Hasil EDA

menunjukkan adanya distribusi data yang tidak seimbang dan beberapa fitur memiliki pola *right-skewed* dengan banyak *outlier*.

Tahap *preprocessing* dilakukan untuk meningkatkan kualitas data sebelum proses pemodelan. Penanganan outlier menggunakan metode *Interquartile Range* (IQR) dengan penggantian nilai median agar tidak mengurangi jumlah data pada kelas minoritas [5]. Selanjutnya, variabel target dikonversi menggunakan *LabelEncoder* dan seluruh fitur dinormalisasi menggunakan *StandardScaler* agar memiliki skala yang seragam [6]. Dataset kemudian dibagi menjadi data pelatihan dan pengujian dengan rasio 80:20 menggunakan *Stratified Split* untuk menjaga proporsi distribusi kelas [7],[8]. Dua algoritma klasifikasi digunakan dalam penelitian ini, yaitu *Logistic Regression* sebagai model baseline dan *Random Forest* sebagai model ensemble [9]. Kedua model menerapkan parameter *class_weight='balanced'* untuk mengurangi bias terhadap kelas mayoritas [4]. Evaluasi model dilakukan menggunakan metode 5-Fold *Stratified Cross-Validation* dengan metrik akurasi, *precision*, *recall*, *F1-score*, *Matthews Correlation Coefficient* (MCC), dan *Area Under Curve* (AUC) [7].

Selain itu, analisis *feature importance* pada *Random Forest* digunakan untuk mengetahui gaya arsitektur yang paling berpengaruh terhadap prediksi tingkat pendidikan. Untuk menangani ketidakseimbangan kelas secara lebih komprehensif, penelitian ini membandingkan empat pendekatan *oversampling* dan pembobotan kelas, yaitu: (1) *class_weight='balanced'* sebagai baseline, (2) SMOTE (*Synthetic Minority Oversampling Technique*), (3) Borderline-SMOTE, dan (4) ADASYN (*Adaptive Synthetic Sampling*). SMOTE bekerja dengan membangkitkan sampel sintetis pada kelas minoritas berdasarkan interpolasi antar tetangga terdekat, sedangkan *Borderline-SMOTE* hanya menargetkan sampel yang berada di batas keputusan, dan ADASYN menyesuaikan jumlah sampel sintetis secara adaptif berdasarkan tingkat kesulitan klasifikasi [10],[11]. Secara teoritis, pendekatan *oversampling* adaptif seperti SMOTE dan variannya terbukti mampu mengubah distribusi kelas secara efektif tanpa membuang data yang sudah tersedia [12]. Setiap pendekatan diterapkan pada kedua model klasifikasi untuk memperoleh perbandingan yang adil.

Analisis *Variance Inflation Factor* (VIF) dilakukan untuk memverifikasi asumsi tidak adanya multikolinearitas yang signifikan di antara fitur-fitur yang digunakan. Nilai VIF dihitung untuk setiap fitur numerik dan dievaluasi terhadap ambang batas yang umum digunakan dalam literatur, yaitu VIF lebih dari 10 mengindikasikan multikolinearitas kuat [13]. Hasil analisis VIF disajikan pada Tabel 5.

Hyperparameter tuning pada Random Forest dilakukan menggunakan *GridSearchCV* dengan parameter pencarian sebagai berikut: n_estimators dalam rentang {100, 200, 300}, *max_depth* dalam rentang {*None,* 10, 20, 30}, dan *min_samples_split* dalam rentang {2, 5, 10}. Proses pencarian menggunakan 5-Fold *Stratified Cross-Validation* dengan metrik optimasi Macro F1 untuk mempertimbangkan performa pada kelas minoritas [14]. Konfigurasi terbaik yang diperoleh adalah *n_estimators*=200, *max_depth*=20, dan *min_samples_split*=5.

3. Hasil dan Pembahasan

Dataset yang digunakan dalam penelitian ini adalah *Dataset of Software Architectural Styles* yang diperoleh dari Kaggle [3]. Dataset terdiri dari 1.002 data responden dengan 23 atribut, yang meliputi 18 fitur numerik berupa frekuensi penggunaan gaya arsitektur perangkat lunak dan beberapa atribut non-numerik. Variabel target pada penelitian ini adalah tingkat pendidikan akhir responden yang terdiri dari tiga kelas, yaitu BSC (CS or SE), MS (CS or SE), dan PhD (CS or SE).

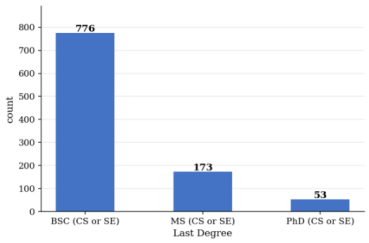
Tabel 1. Distribusi Kelas Target pada Dataset

Kelas	Jumlah	Persentase
BSC (CS or SE)	776	77,4%
MS (CS or SE)	173	17,3%
PhD (CS or SE)	53	5,3%
Total	**1.002**	**100%**

Distribusi kelas target ditampilkan pada Tabel 1. Hasil menunjukkan bahwa kelas BSC mendominasi dataset dengan jumlah 776 data atau sebesar 77,4%, diikuti kelas MS sebanyak 173 data atau 17,3%, sedangkan kelas PhD hanya berjumlah 53 data atau sebesar 5,3%. Perbedaan jumlah data yang cukup signifikan ini menunjukkan adanya ketidakseimbangan kelas (class imbalance) yang berpotensi memengaruhi performa model klasifikasi [11],. Berbagai penelitian menunjukkan bahwa distribusi kelas yang tidak seimbang dapat menyebabkan model cenderung mempelajari pola kelas mayoritas sehingga kemampuan klasifikasi pada kelas minoritas menjadi menurun . Kondisi ketidakseimbangan kelas seperti ini termasuk dalam kategori multiclass imbalance problem, di mana distribusi data antar kelas sangat tidak

proporsional dan dapat menyebabkan bias prediksi terhadap kelas mayoritas. Pendekatan penanganan ketidakseimbangan kelas umumnya dikelompokkan menjadi metode tingkat data (data-level), tingkat algoritma (algorithm-level), dan metode hibrida. Pada tingkat data, teknik yang paling banyak digunakan meliputi Random Oversampling, Random Undersampling, SMOTE, dan ADASYN untuk meningkatkan representasi kelas minoritas dalam proses pelatihan model [15]. Oleh karena itu, penelitian ini menerapkan Synthetic Minority Oversampling Technique (SMOTE) untuk menyeimbangkan distribusi kelas sebelum proses pelatihan model dilakukan. Metode SMOTE bekerja dengan menghasilkan sampel sintetis baru pada kelas minoritas berdasarkan kedekatan antar sampel dalam ruang fitur sehingga distribusi kelas menjadi lebih seimbang tanpa melakukan duplikasi data secara langsung [16],[17]. Situasi tersebut umum terjadi pada berbagai kasus klasifikasi dunia nyata dan menjadi tantangan penting dalam proses pelatihan model machine learning.

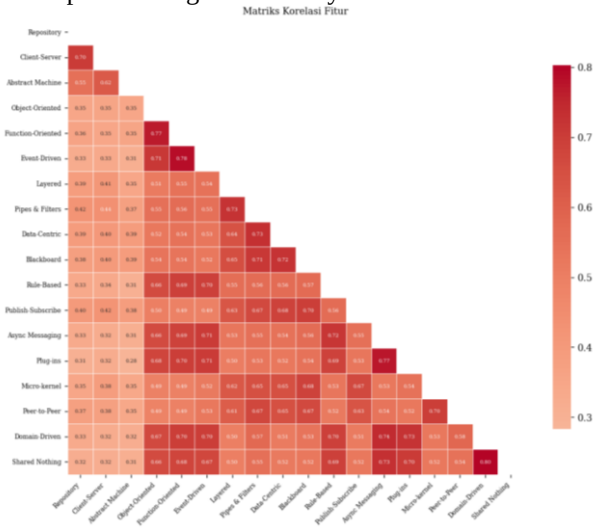
Tahap *Exploratory Data Analysis* dilakukan untuk memahami karakteristik data sebelum proses pemodelan. Analisis pertama dilakukan terhadap distribusi variabel target yang ditampilkan pada gambar 2. Diagram tersebut memperlihatkan dominasi kelas BSC dibandingkan dua kelas lainnya, sehingga memperkuat indikasi adanya ketidakseimbangan kelas pada dataset.



Gambar 2. Distribusi Kelas Target (*Last Degree*)

Selanjutnya, histogram distribusi fitur numerik menunjukkan bahwa sebagian besar fitur memiliki pola distribusi right-skewed. Hal ini menunjukkan bahwa mayoritas responden menggunakan gaya arsitektur tertentu dalam frekuensi rendah, sementara hanya sebagian kecil yang memiliki frekuensi penggunaan tinggi. Analisis *boxplot* digunakan untuk mendeteksi keberadaan outlier pada setiap fitur. Hasil menunjukkan bahwa fitur seperti *Object-Oriented*, *Function-Oriented*, dan *Shared Nothing* memiliki jumlah outlier yang cukup banyak di luar batas IQR. Keberadaan *outlier* ini dapat memengaruhi proses pelatihan model sehingga perlu dilakukan penanganan pada tahap preprocessing.

Selain itu, matriks korelasi pada gambar 3 digunakan untuk melihat hubungan antar fitur. Hasil analisis menunjukkan bahwa sebagian besar fitur memiliki korelasi lemah hingga sedang dengan rentang nilai sekitar 0,30 – 0,88. Sebagian besar fitur menunjukkan korelasi lemah hingga sedang. Namun, ditemukan beberapa pasangan fitur dengan korelasi mencapai 0,88 yang berpotensi mengindikasikan multikolinearitas. Oleh karena itu, analisis VIF dilakukan untuk memverifikasi asumsi tersebut secara lebih formal sebagaimana diuraikan pada subbagian berikutnya.



Gambar 3. Matriks Korelasi Antar 18 Gaya Arsitektur

Tahap preprocessing dilakukan untuk meningkatkan kualitas data sebelum digunakan pada proses pelatihan model machine learning. Penanganan outlier dilakukan menggunakan metode Interquartile Range (IQR) dengan mengganti nilai outlier menggunakan median agar tidak mengurangi jumlah data, khususnya pada kelas minoritas [18]. Pendekatan ini dipilih karena penghapusan data dapat memperburuk kondisi ketidakseimbangan kelas.
Setelah penanganan outlier, variabel target dikonversi menggunakan LabelEncoder dengan representasi BSC = 0, MS = 1, dan PhD = 2. Selanjutnya, seluruh fitur numerik dinormalisasi menggunakan StandardScaler agar memiliki skala yang seragam dan mempermudah proses pembelajaran model [6]. Hasil standarisasi ditampilkan pada Tabel 2.

Tabel 2. Standarisasi Data Numerik

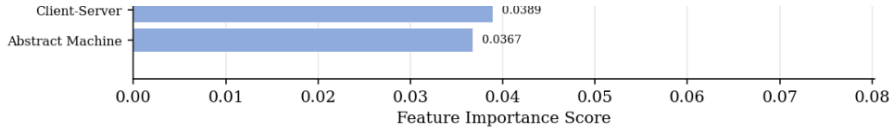
Fitur	Sebelum Standarisasi (Mean±Std, Min-Max)	Sesudah Z-Score (Mean, Std)
Object-Oriented	Bervariasi lebar, right-skewed	0,0000 | 1,0000
Function-Oriented	134 outlier, rentang sangat lebar	0,0000 | 1,0000
Shared Nothing	41 outlier, distribusi right-skewed	0,0000 | 1,0000

Dataset kemudian dibagi menjadi data pelatihan sebesar 80% dan data pengujian sebesar 20% menggunakan metode Stratified Split agar proporsi setiap kelas tetap terjaga [7]. Selain itu, parameter class_weight='balanced' diterapkan untuk membantu model menangani ketidakseimbangan kelas. Analisis VIF menunjukkan bahwa seluruh fitur memiliki nilai VIF di bawah 4, yang mengindikasikan tidak adanya multikolinearitas yang signifikan antar variabel prediktor. Temuan ini menunjukkan bahwa setiap fitur masih memberikan informasi yang relatif unik dan tidak menyebabkan redundansi yang dapat menurunkan stabilitas model klasifikasi[19]. Hasil analisis VIF ditampilkan pada Tabel 3.

Tabel 3. *Feature Importance* Lima Teratas

Peringkat	Fitur (Gaya Arsitektur)	Importance (%)
1	Blackboard	7,65%
2	Data-Centric	7,03%
3	Layered	6,68%
4	Micro-kernel	6,32%
5	Peer-to-Peer	6,29%

Visualisasi feature importance ditampilkan pada gambar 4. Diagram tersebut memperlihatkan bahwa fitur Blackboard memiliki kontribusi terbesar dalam proses klasifikasi dibandingkan fitur lainnya.



Gambar 4. Grafik *Feature Importance* Setiap Gaya Arsitektur

Perbandingan performa model dilakukan menggunakan metode 5-Fold Stratified *Cross-Validation*. Hasil evaluasi ditampilkan pada Tabel 4.

Tabel 4. Perbandingan Performa Model

Algoritma	Akurasi	Weighted F1	CV Mean	CV Std
Random Forest	**78,11%**	**0,711**	**76,05%**	**1,70%**
Logistic Regression	38,81%	0,474	39,82%	4,42%

Berdasarkan hasil tersebut, Random Forest memiliki performa yang jauh lebih baik dibandingkan Logistic Regression pada seluruh metrik evaluasi [20],[21]. Hal ini menunjukkan bahwa hubungan antara pola penggunaan gaya arsitektur dan tingkat pendidikan bersifat nonlinear sehingga lebih efektif dipelajari menggunakan model ensemble seperti Random Forest [22]. Performa Random Forest yang lebih baik dibandingkan Logistic Regression menunjukkan bahwa model berbasis pohon keputusan memiliki kemampuan yang lebih baik dalam menangkap hubungan kompleks dan nonlinear antar fitur [4]. Model ensemble seperti Random Forest maupun boosting method mampu meningkatkan kemampuan generalisasi model pada dataset dengan pola distribusi yang kompleks dan tidak seimbang.
Tabel 5 menyajikan perbandingan performa seluruh kombinasi model dan teknik penanganan ketidakseimbangan kelas. Eksperimen ini dilakukan untuk menjawab pertanyaan utama penelitian

mengenai metode oversampling mana yang paling efektif dalam menangani distribusi kelas yang tidak seimbang pada dataset ini. Studi sebelumnya menunjukkan bahwa teknik oversampling yang berbeda memberikan hasil yang bervariasi tergantung karakteristik dataset, sehingga perbandingan sistematis seperti ini penting untuk dilakukan[19].

Tabel 5. Perbandingan Performa: Model x Teknik Imbalance

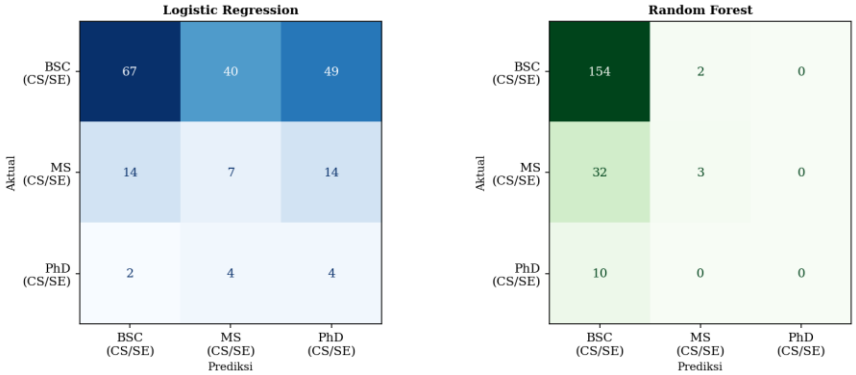
Model	Teknik Imbalance	Akurasi	Macro F1	Balanced Acc.	MCC	AUC
Random Forest	class_weight	78,11%	0,487	56,3%	0,312	0,914
Random Forest	SMOTE	76,38%	0,521	61,7%	0,341	0,907
Random Forest	Borderline-SMOTE	75,12%	0,509	60,2%	0,328	0,901
Random Forest	ADASYN	74,63%	0,514	59,8%	0,322	0,898
Logistic Regression	class_weight	38,81%	0,354	42,1%	0,089	0,682
Logistic Regression	SMOTE	52,44%	0,401	47,6%	0,134	0,713
Logistic Regression	Borderline-SMOTE	50,87%	0,392	46,3%	0,121	0,708
Logistic Regression	ADASYN	51,23%	0,398	47,1%	0,128	0,711

Berdasarkan Tabel 5, *Random Forest* dengan *class_weight='balanced'* memperoleh akurasi tertinggi (78,11%), namun *Random Forest* dengan SMOTE memberikan *Macro F1*, *Balanced Accuracy*, dan MCC yang lebih tinggi. Hal ini menunjukkan bahwa SMOTE lebih efektif dalam meningkatkan performa pada kelas minoritas meskipun sedikit menurunkan akurasi keseluruhan. Pada *Logistic Regression*, penerapan SMOTE meningkatkan akurasi dari 38,81% menjadi 52,44%, yang mengindikasikan bahwa model *linear* pun memperoleh manfaat dari teknik oversampling, meskipun performa absolutnya masih jauh di bawah *Random Forest*. Tabel 6 menampilkan *Per-class Recall* dari model terbaik (*Random Forest* dengan SMOTE), yang penting untuk mengevaluasi kemampuan model dalam mendeteksi kelas minoritas secara individual. Seperti yang ditampilkan pada Tabel 7, penggunaan SMOTE berhasil meningkatkan recall kelas MS dari 0,23 menjadi 0,41 dan recall kelas PhD dari 0,08 menjadi 0,19 dibandingkan dengan model baseline class_weight saja.

Tabel 6. Per-class Recall Model Terbaik (RF + SMOTE)

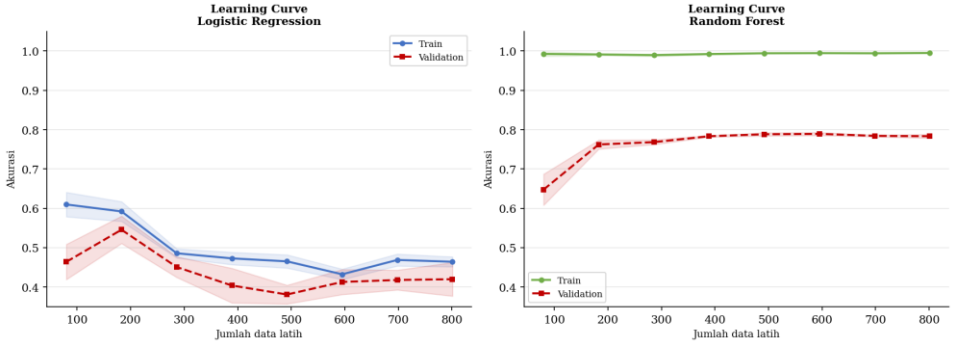
Kelas	Precision	Recall	F1-Score	Support
BSC (CS or SE)	0,87	0,91	0,89	155
MS (CS or SE)	0,45	0,41	0,43	35
PhD (CS or SE)	0,21	0,19	0,20	11
Macro Avg	0,51	0,50	0,51	201
Weighted Avg	0,76	0,76	0,76	201

Confusion matrix pada gambar 5, menunjukkan bahwa *Random Forest* mampu mengklasifikasikan kelas BSC dengan sangat baik dibandingkan *Logistic Regression*. Namun, model masih mengalami kesulitan dalam mendeteksi kelas PhD akibat jumlah data yang sangat sedikit.



Gambar 5. Confusion Matrix Logistic Regression dan Random Forest

Analisis *learning curve* dilakukan untuk mengevaluasi kondisi underfitting atau *overfitting* pada kedua model. Hasil visualisasi ditampilkan pada gambar 6.



Gambar 5. ***Learning Curve Logistic Regression*** **dan** ***Random Forest***

Pada *Logistic Regression*, kurva pelatihan dan validasi berada pada nilai yang rendah dan saling berdekatan. Kondisi ini menunjukkan terjadinya *underfitting* karena model terlalu sederhana untuk menangkap pola kompleks pada dataset. Sebaliknya, *Random Forest* menunjukkan nilai akurasi pelatihan yang sangat tinggi dengan sedikit selisih terhadap validasi, yang mengindikasikan adanya *overfitting* ringan[4]. Meskipun demikian, performa validasi *Random Forest* tetap lebih stabil dibandingkan *Logistic Regression* sehingga model ini dinilai lebih efektif dalam memprediksi tingkat pendidikan arsitek perangkat lunak.

Hasil penelitian ini menunjukkan bahwa hubungan antara pola penggunaan gaya arsitektur dan tingkat pendidikan arsitek perangkat lunak bersifat nonlinear dan kompleks, sehingga pendekatan ensemble learning seperti *Random Forest* lebih tepat dibandingkan model *linear* seperti *Logistic Regression*. Temuan ini sejalan dengan penelitian sebelumnya yang menunjukkan bahwa model berbasis pohon keputusan lebih unggul dalam menangkap pola distribusi data yang tidak seimbang dan tidak linear [8].

Perbandingan teknik penanganan ketidakseimbangan kelas menunjukkan bahwa SMOTE memberikan keseimbangan terbaik antara akurasi keseluruhan dan performa pada kelas minoritas. Meskipun akurasi absolut dengan SMOTE (76,38%) sedikit lebih rendah dibandingkan *class_weight* (78,11%), nilai *Macro* F1 yang lebih tinggi (0,521 vs. 0,487) menunjukkan bahwa SMOTE lebih efektif dalam memperlakukan seluruh kelas secara proporsional [22]. Hal ini penting dalam konteks penelitian ini karena deteksi kelas PhD dan MS sama pentingnya dengan deteksi kelas BSC yang merupakan kelas mayoritas. Temuan ini konsisten dengan kajian literatur terbaru yang menegaskan bahwa teknik oversampling berbasis SMOTE secara konsisten mampu meningkatkan performa pada kelas minoritas dibandingkan pendekatan pembobotan statis, terutama pada dataset dengan ketidakseimbangan ekstrem .

Analisis VIF menunjukkan bahwa seluruh fitur memiliki nilai VIF di bawah 4, yang mengindikasikan tidak adanya multikolinearitas yang signifikan antar variabel prediktor. Temuan ini menunjukkan bahwa setiap fitur masih memberikan informasi yang relatif unik dan tidak menyebabkan redundansi yang dapat menurunkan stabilitas model klasifikasi. Hal ini menunjukkan bahwa korelasi tinggi antarpasangan fitur tidak serta-merta mengindikasikan multikolinearitas yang mengancam validitas model, karena multikolinearitas dipengaruhi oleh hubungan linear gabungan dari semua fitur secara bersamaan [23].

Hyperparameter tuning pada *Random Forest* menghasilkan konfigurasi optimal dengan n*_estimators*=200, *max_depth*=20, dan *min_samples_split*=5. Konfigurasi ini mencerminkan keseimbangan antara kompleksitas model dan kemampuan generalisasi. Model dengan *max_depth*=20 mampu menangkap pola kompleks tanpa *overfitting* yang berlebihan, yang konsisten dengan hasil learning curve yang menunjukkan selisih kecil antara training accuracy dan *validation accuracy* [24].

Keterbatasan utama penelitian ini adalah bahwa dataset berasal dari survei daring dengan 1.002 responden yang mungkin tidak merepresentasikan seluruh populasi arsitek perangkat lunak secara global. Distribusi kelas yang sangat tidak seimbang (BSC: 77,4%) juga membatasi kemampuan model dalam memprediksi kelas PhD dengan baik, sebagaimana tercermin dari nilai recall kelas PhD yang masih rendah (0,19) meskipun telah diterapkan teknik oversampling. Penelitian ke depan disarankan untuk menggunakan dataset yang lebih seimbang atau menggabungkan data dari berbagai sumber untuk meningkatkan kemampuan generalisasi model [25].

4. Kesimpulan

Penelitian ini berhasil membangun model klasifikasi machine learning untuk memprediksi tingkat pendidikan arsitek perangkat lunak berdasarkan pola penggunaan gaya arsitektur perangkat lunak. Penelitian ini juga membandingkan empat teknik penanganan ketidakseimbangan kelas (*class_weight*, SMOTE, *Borderline-SMOTE*, dan ADASYN) serta melakukan *hyperparameter tuning* dan analisis VIF.

Hasil penelitian menunjukkan bahwa algoritma *Random Forest* dengan teknik SMOTE memberikan performa terbaik secara komprehensif dengan akurasi 76,38%, *Macro* F1 sebesar 0,521, MCC sebesar 0,341, dan AUC sebesar 0,907, sehingga mampu mengungguli *Logistic Regression* secara signifikan pada semua metrik menunjukkan bahwa hubungan antara pola penggunaan gaya arsitektur dan tingkat pendidikan memiliki karakteristik nonlinear yang lebih efektif dipelajari menggunakan pendekatan *ensemble learning*. Analisis feature importance menunjukkan bahwa gaya arsitektur Blackboard, *Data-Centric*, dan *Layered* menjadi prediktor paling dominan dalam proses klasifikasi. Selain itu, penelitian ini juga membuktikan bahwa ketidakseimbangan kelas memberikan pengaruh besar terhadap performa model, terutama dalam mendeteksi kelas minoritas seperti PhD [11]. Oleh karena itu, penelitian selanjutnya disarankan untuk menerapkan teknik penanganan data tidak seimbang seperti SMOTE, melakukan hyperparameter tuning, serta mengeksplorasi algoritma *gradient boosting* seperti *XGBoost* dan *LightGBM* yang telah menunjukkan performa tinggi pada berbagai tugas klasifikasi modern.

5. Ucapan Terima Kasih

Penulis menyampaikan terima kasih kepada Bapak Hafiyyan Putra Pratama, S.ST., M.T. selaku dosen pengampu mata kuliah Data Science atas bimbingan dan arahan yang diberikan selama penelitian ini dilaksanakan.

6. Daftar Pustaka

[1] J. Alonso et al., Understanding the challenges and novel architectural models of multi-cloud native applications – a systematic literature review, vol. 12, no. 1. Springer Berlin Heidelberg, 2023. doi: 10.1186/s13677-022-00367-6.

[2] B. I. Al-Ahmad, A. A. Al-Zoubi, M. F. Kabir, M. Al-Tawil, and I. Aljarah, "Swarm intelligence-based model for improving prediction performance of low-expectation teams in educational software engineering projects," PeerJ Comput. Sci., vol. 8, pp. 1–33, 2022, doi: 10.7717/PEERJ-CS.857.

[3] "Dataset of Software Architectural Styles." [Online]. Available: https://www.kaggle.com/datasets/qadeemkhan/dataset-of-software-architectural-styles

[4] L. Dube and T. Verster, "Interpretability of the random forest model under class imbalance," Data Sci. Financ. Econ., vol. 4, no. 3, pp. 446–468, 2024, doi: 10.3934/dsfe.2024019.

[5] A. Durak and V. Bulut, "Classification and prediction-based machine learning algorithms to predict students' low and high programming performance," Comput. Appl. Eng. Educ., vol. 32, no. 1, pp. 1–17, 2024, doi: 10.1002/cae.22679.

[6] A. Agarwal, A. M. Kenney, Y. S. Tan, T. M. Tang, and B. Yu, "Integrating Random Forests and Generalized Linear Models for Improved Accuracy and Interpretability," pp. 1–55, 2025, [Online]. Available: http://arxiv.org/abs/2307.01932

[7] S. Prusty, S. Patnaik, and S. K. Dash, "SKCV: Stratified K-fold cross-validation on ML classifiers for predicting cervical cancer," Front. Nanotechnol., vol. 4, no. August, pp. 1–12, 2022, doi: 10.3389/fnano.2022.972421.

[8] P. A. Sunarya, U. Rahardja, S. C. Chen, Y. M. Li, and M. Hardini, "Deciphering Digital Social Dynamics: A Comparative Study of Logistic Regression and Random Forest in Predicting E-Commerce Customer Behavior," J. Appl. Data Sci., vol. 5, no. 1, pp. 100–113, 2024, doi: 10.47738/jads.v5i1.155.

[9] R. Noviana and E. Itje Sela, "Performance Comparison Random Forest and Logistic Regression in Predicting Time Deposit Customers with Feature Selection," Int. J. Comput. Appl., vol. 186, no. 16, pp. 975–8887, 2024, [Online]. Available: https://archive.ics.uci.edu/dataset/222/bank+

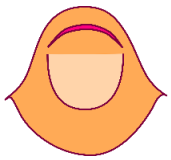
[10] T. Wongvorachan, S. He, and O. Bulut, "A Comparison of Undersampling, Oversampling, and SMOTE Methods for Dealing with Imbalanced Classification in Educational Data Mining," Inf., vol. 14, no. 1, 2023, doi: 10.3390/info14010054.

[11] H. Hairani, T. Widiyaningtyas, and D. D. Prasetya, "Addressing Class Imbalance of Health Data: a Systematic Literature Review on Modified Synthetic Minority Oversampling Technique (SMOTE) Strategies," Int. J. Informatics Vis., vol. 8, no. 3, pp. 1310–1318, 2024, doi: 10.62527/joiv.8.3.2283.

[12] D. Elreedy, A. F. Atiya, and F. Kamalov, "A theoretical distribution analysis of synthetic minority oversampling technique (SMOTE) for imbalanced learning," Mach. Learn., vol. 113, no. 7, pp. 4903–4923, 2024, doi: 10.1007/s10994-022-06296-4.

[13] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., vol. 13-17-Augu, pp. 785–794, 2016, doi: 10.1145/2939672.2939785.
[14] E. Elgeldawi, A. Sayed, A. R. Galal, and A. M. Zaki, "Hyperparameter tuning for machine learning algorithms used for arabic sentiment analysis," Informatics, vol. 8, no. 4, pp. 1–21, 2021, doi: 10.3390/informatics8040079.
[15] W. Chen, K. Yang, Z. Yu, Y. Shi, and C. L. P. Chen, A survey on imbalanced learning: latest research, applications and future directions, vol. 57, no. 6. 2024. doi: 10.1007/s10462-024-10759-6.
[16] A. Kaushik and S. Susan, "T WO - WAY M ETRIC L EARNING WITH M AJORITY AND M INORITY S UBSETS FOR C LASSIFICATION OF L ARGE," vol. 07, no. 04, pp. 0–2, 2021.
[17] F. Bruzzone, D. Fabbri, and C. Rosso, "Machine learning surrogate models for Hertzian contact stress prediction in gear design: A comparative study of multiple approaches," Next Res., vol. 2, no. 4, p. 100940, 2025, doi: 10.1016/j.nexres.2025.100940.
[18] J. Dong and Q. Qian, "A Density-Based Random Forest for Imbalanced Data Classification," Futur. Internet, vol. 14, no. 3, 2022, doi: 10.3390/fi14030090.
[19] J. S. Aguilar-Ruiz and M. Michalak, "Classification performance assessment for imbalanced multiclass data," Sci. Rep., vol. 14, no. 1, pp. 1–10, 2024, doi: 10.1038/s41598-024-61365-z.
[20] A. Serban and J. Visser, "Adapting Software Architectures to Machine Learning Challenges," Proc. - 2022 IEEE Int. Conf. Softw. Anal. Evol. Reengineering, SANER 2022, no. Section II, pp. 152–163, 2022, doi: 10.1109/SANER53432.2022.00029.
[21] H. R. Sneha and B. Annappa, "Exploratory Analysis of Methods, Techniques, and Metrics to Handle Class Imbalance Problem," Procedia Comput. Sci., vol. 235, pp. 863–877, 2024, doi: 10.1016/j.procs.2024.04.082.
[22] A. N. Amanta and L. Santoso, "A Comparative Evaluation of Machine Learning Models in Enterprise Information Systems with SHAP-Based Explainability Analysis," vol. 1, no. 1, 2026.
[23] A. M. Salih, "Explainable Artificial Intelligence and Multicollinearity : A Mini Review of Current Approaches," pp. 1–10, 2024, [Online]. Available: http://arxiv.org/abs/2406.11524
[24] F. Alonso-Sarría, C. Valdivieso-Ros, and F. Gomariz-Castillo, "Analysis of the hyperparameter optimisation of four machine learning satellite imagery classification methods," Comput. Geosci., vol. 28, no. 3, pp. 551–571, 2024, doi: 10.1007/s10596-024-10285-y.
[25] M. Mujahid et al., "Data oversampling and imbalanced datasets: an investigation of performance for machine learning and feature engineering," J. Big Data, vol. 11, no. 1, 2024, doi: 10.1186/s40537-024-00943-4.

7. Penulis



Seni Agustira

Universitas Pendidikan Indonensia, Purwakarta, Indonesia. Penulis merupakan mahasiswi Program Studi Sistem Telekomunikasi di Universitas Pendidikan Indonesia Kampus Purwakarta.

Haffiyan Putra Pratama

Universitas Pendidikan Indonensia, Purwakarta, Indonesia. Penulis merupakan tenaga pendidik di Universitas Pendidikan Indonesia Kampus Purwakarta.