ISSN: 2477-5126
e-ISSN: 2548-9356

# Penilaian Kredit Menggunakan Algoritma XGBoost dan Logistic Regression

**Ainul Yaqin [1*)], Gani Ramadhani [2]**
[1]Jurusan Informatika, Fakultas Komputer, Universitas Amikom Yogyakarta, Yogyakarta
[2]Fakultas Ilmu Komputer, Universitas Amikom Yogyakarta, Yogyakarta
[1,2]Jl. Ringroad Utara, Condongcatur, Depok, Sleman, 55283, Yogyakarta, Indonesia
email: [1]ainulyaqin@amikom.ac.id, [2]gani.ramadhani@students.amikom.ac.id

*Abstract* – **Credit Collectability is a process or system used by financial institutions or banks to assess the eligibility of a person applying for a loan. Credit Collectability is very necessary to avoid losses due to default. An efficient, fast and accurate method is needed to classify whether or not someone is eligible for a loan. The author proposes a machine learning method for classification, namely the XGBoost algorithm and logisitc regression then compares the two algorithms. After being trained and tested with stratified kfold cross validation, XGBoost produces an average accuracy of 85,51%; F1 Score 83,81%; precision 83,80% and recall 84,04% while logistic regression produces an average accuracy of 85,94%; F1 Score 85,36%; precision 80,08%; and 91,52% recalls. Both algorithms can classify whether or not a person is eligible for a loan properly, so that it can be used to help financial institutions and credit analysts.**

*Abstrak* – **Penilaian kredit merupakan suatu proses atau sistem yang digunakan oleh lembaga pembiayaan atau bank untuk menilai kelayakan seseorang yang mengajukan pinjaman. Hal ini sangat diperlukan untuk menghindari kerugian akibat gagal bayar. Menanggapi hal tersebut dibutuhkan sebuah metode yang efisien, cepat dan akurat untuk mengklasifikasikan layak atau tidaknya seseorang untuk diberikan pinjaman. Penulis mengusulkan metode machine learning untuk klasifikasi yaitu algoritma *XGBoost* dan *logisitc regression,* kemudian membandingkan kedua algoritma tersebut. Setelah dilatih dan diuji dengan *stratified kfold cross validation*, *XGBoost* menghasilkan rata-rata akurasi 85,51%; *F1 Score* 83,81%; *precision* 83,80% dan *recall* 84,04% sedangkan *logisitc regression* menghasilkan rata-rata akurasi 85,94%; *F1 Score* 85,36%; *precision* 80,08%; dan *recall* 91,52%. Kedua algoritma dapat mengklasifikasikan layak atau tidaknya seseorang untuk diberikan pinjaman dengan baik, sehingga dapat digunakan untuk membantu institusi keuangan maupun para analis kredit.**

***Kata Kunci* – penilaian kredit, klasifikasi, *XGBoost*, *logistic regression*, *data mining***

## I. PENDAHULUAN

Fungsi utama bank dan lembaga keuangan adalah untuk memberikan kredit dan pinjaman kepada nasabah [1], namun yang menjadi tantangan adalah menilai layak atau tidaknya nasabah untuk diberi pinjaman [2]. Karena hal tersebut dibutuhkan yang namanya penilaian kredit. Jika nasabah dinilai layak maka kemungkinan besar mampu membayar pinjaman dengan baik, sehingga jika pinjaman tidak disetujui maka akan menimbulkan kehilangan potensi keuntungan pada bank maupun lembaga keuangan [3]. Namun jika nasabah dinilai tidak layak maka kemungkinan nasabah tersebut tidak mampu untuk melunasi pinjaman, sehingga jika menyetujui pinjaman tersebut maka bisa menimbulkan kerugian [3].

Karena hal tersebut penilaian kredit sangat penting dilakukan. Berbagai manfaat penilaian kredit bagi lembaga keuangan adalah mengurangi resiko kredit, membuat keputusan manajerial dan meningkatkan serta mengefisiensikan arus kas, sehingga model penilaian kredit sangat berpengaruh pada profitabilitas lembaga keuangan [2]. Namun sering kali proses penilaian kredit memakan waktu yang cukup lama, akurasi sangat bergantung kepada keahlian dari para analis kredit dan memakan biaya [4]. Untuk itu dibutuhkan sebuah metode yang dapat digunakan untuk melakukan penilaian kredit secara cepat, akurat dan efisien.

Penulis memilih algoritma *XGBoost* dikarenakan algoritma ini telah memenangkan banyak kompetisi *machine learning* serta memiliki waktu kalkulasi yang cepat dan performa yang baik [7] [8]. Sedangkan penulis membandingkan dengan *logistic regression* dikarenakan algoritma ini telah banyak diimplementasikan untuk berbagai klasifikasi *binary* [9], serta *logistic regression* cukup mudah untuk diimplementasikan dan terkadang memiliki performa yang lebih baik dari algoritma *tree* dan *neural network* [7].

Metode kecerdasan buatan atau model *machine learning* dapat mengatasi permasalahan penilaian kredit manual [5]. Sebuah model harus dapat mengklasifikasikan layak atau tidaknya nasabah untuk diberi pinjaman [6]. Penulis membandingkan algoritma *machine learning* yaitu *XGBoost* dengan *logistic regression* yang nantinya akan digunakan untuk mengklasifikasikan kondisi *binary* layak atau tidaknya nasabah.

## II. PENELITIAN YANG TERKAIT

Terdapat banyak karya ilmiah yang membahas mengenai model penilaian kredit. Di Wang dan Zuoquan Zhang menggunakan algoritma *Support Vector Machine (SVM)* dan *logistic regression* yang digabungkan dengan *information fusion* menggunakan algoritma *Dempster-Shafer theory* menghasilkan akurasi sebesar 85% [10]. Hermawan dan Yoannita menggunakan algoritma *Classification and Regression Tree (CART)* kemudian dilatih dan diuji dengan *k-fold validation* menghasilkan rata-rata akurasi sebesar 85,36% [11]. Muhammad Husni Rifqo dan Ardi Wijaya menggunakan algoritma *Naïve Bayes* dan menghasilkan akurasi 80% [12].

TABEL 1 PERBANDINGAN PENELITI PENILAIAN KREDIT

*) **penulis korespondensi**: Gani Ramadhani
Email: gani.ramadhani@students.amikom.ac.id

| Metode | Akurasi |
|---|---|
| *SVM + logistic regression + Dempster-Shafer theory* [10] | 85% |
| *CART* [11] | 85,36% |
| *Naïve Bayes* [12] | 80% |

Pada penelitian ini, penulis membandingkan algoritma *XGBoost* dan *logistic regression* untuk penilaian kredit menggunakan dataset yang sama dengan penelitian yang terdapat dalam tabel 1.

## III. XGBOOST DAN LOGISTIC REGRESSION

### A. XGBoost

*XGBoost* merupakan salah satu implementasi dari *Gradient Boosting Machines* (GBM) [13]. Algoritma ini bisa digunakan untuk penyelesaian klasifikasi maupun regressi [14]. *XGBoost* mengkombinasikan prediksi dari beberapa pohon yang lemah menjadi pohon yang memiliki performa yang baik secara iteratif [15]. *XGBoost* menggunakan residual untuk mengkalibrasikan pohon sebelumnya pada setiap iterasi, proses ini disebut dengan optimisasi *loss function.*

Dalam pembuatan pohon pada *XGBoost* diperlukan menghitung nilai residual menggunakan persamaan (1).

$$residual_i = y_i - \hat{y}_i \quad (1)$$

Dimana $y_i$ merupakan nilai target dan $\hat{y}_i$ merupakan nilai prediksi probabilitas. Pada pohon pertama, $\hat{y}_i$ dapat di isi dengan nilai 0,5 yang merupakan nilai probabilitas awal. Kemudian untuk memilih *node* yang sesuai dilakukan dengan menghitung *similarity scor*e menggunakan persamaan (2).

$$similarity\ score = \frac{(\sum residual)^2}{\sum[probability_i \times (1 - probability_i)] + \lambda} \quad (2)$$

Dimana $\lambda$ merupakan nilai parameter *regularization* pada *XGBoost*. Setelah menentukan *node* diperlukan proses *pruning* atau mengecilkan ukuran pohon dengan menghilangkan beberapa cabang yang lemah. Untuk proses *pruning* perlu menghitung *gain* menggunakan persamaan (3) dan membandingkannya dengan nilai *cover* yang dihitung dengan persamaan (4).

$$gain = Left_{Similarity\ score} + Right_{Similarity\ score} - Root_{Similarity\ score} \quad (3)$$

$$cover = \sum[probability_i \times (1 - probability_i)] \quad (4)$$

Jika nilai *gain* lebih kecil daripada nilai *cover* maka cabang tersebut perlu dihapus. Untuk membuat prediksi perlu menghitung nilai *output* menggunakan persamaan (5) berikut ini.

$$output = \frac{(\sum residual_i)}{\sum\ [probability_i \times (1 - probability_i)\ + \lambda]} \quad (5)$$

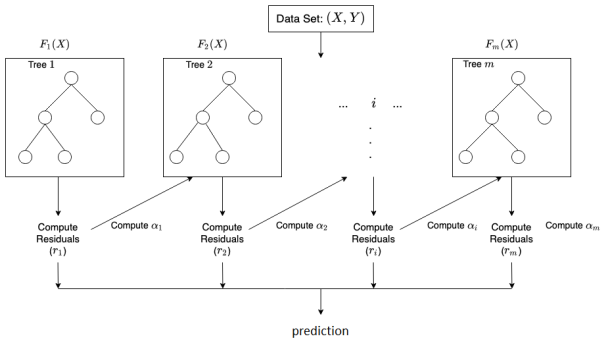
Kemudian nilai output ini perlu ditambahkan dengan fungsi *log odd* dan *learning rate* menggunakan persamaan (6).

$$\log(odds)\ prediction = \log\left(\frac{probability_i}{1 - probability_i}\right) + (\eta + output) \quad (6)$$

Dimana $\eta$ merupakan nilai *learning rate*. Nilai yang dihasilkan dari persamaan (6) diatas ini agar menjadi nilai probabilitas yang memiliki nilai antara 0 dan 1 sehingga bisa digunakan untuk klasifikasi perlu dimasukkan ke dalam fungsi *sigmoid* yang direpresentasikan dalam persamaan (7) berikut ini.

$$probability = \frac{1}{1 - e^{-(x)}} \quad (7)$$

Proses di atas dilakukan secara iteratif hingga nilai residual semakin mengecil atau jumlah iterasi maksimal sudah tercapai. Arsitektur algoritma *XGBoost* seperti yang direpresentasikan pada gambar 1 berikut.




Gbr. 1 Ilustrasi XGBoost

Algoritma XGBoost memiliki peforma yang sangat baik dan cepat terutama pada dataset yang besar. Namun seringkali algoritma ini memiliki kelemahan yaitu overfitting.

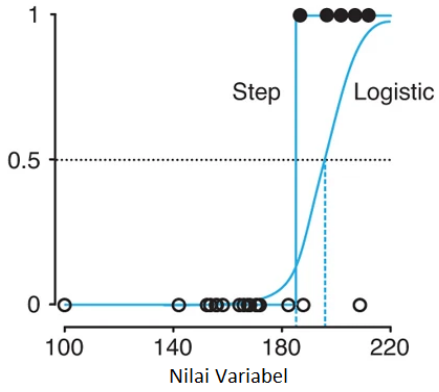
### B. Logistic Regression

*Logistic regression* adalah algoritma yang digunakan untuk mengestimasi probabilitas suatu kondisi [9]. Algoritma ini merupakan kombinasi linear dari variabel independen yang kemudian dikonversi menjadi nilai antara 0 dan 1 yang digunakan untuk klasfisikasi [16]. *Logistic regression* akan memodelkan peluang suatu kejadian berdasarakan karakteristik variabel menggunakan fungsi *logit* (8) [17].

$$P = \frac{1}{1 - e^{-(b_0 + b_1 x_1 + \cdots + b_i x_i)}} \quad (8)$$

Dimana $b_0, b_1, \ldots, b_i$ adalah koefisien regresi dan $x_1, \ldots, x_i$ adalah variabel input. Ilustrasi perhitungan fungsi *logit* (8) akan menghasilkan garis biru diagonal pada gambar 2.

ISSN: 2477-5126
e-ISSN: 2548-9356




Gbr. 2 Ilustrasi Logistic Regression

Setelah mendapatkan output antara 0 dan 1 algoritma ini bisa digunakan untuk mengklasifikan suatu kategori dengan persamaan (9).

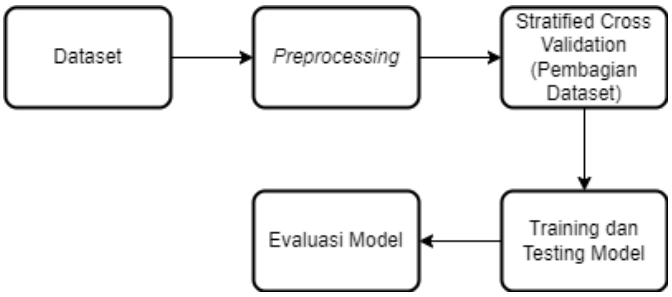
$$Y_i = \begin{cases} 1 \; if \; x \geq 0{,}5 \\ 0 \; if \; x < 0{,}5 \end{cases} \qquad (9)$$

Jika hasil perhitungan (9) menghasilkan nilai lebih dari sama dengan 0,5 maka dianggap positif sedangkan jika hasil perhitungan dibawah 0,5 maka dianggap negatif.

Logistic regression memiliki akurasi yang baik terutama pada dataset yang tidak terlalu besar dan simpel. Algoritma ini sangat efektif dan jarang overfitting, namun seringkali tidak memiliki peforma yang baik pada dataset yang besar dan kompleks.

## IV. METODE PENELITIAN

Langkah-langkah penelitian direpresentasikan pada gambar 3 di bawah ini.

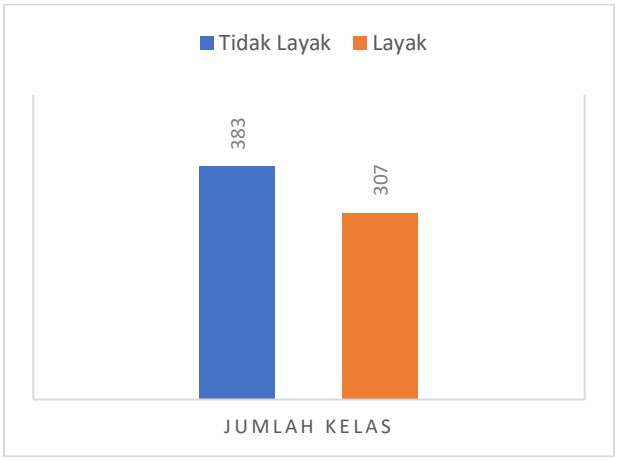



Gbr. 3 Alur Penelitian

Penelitian ini melalui beberapa langkah dari melakukan proses *preprocessing* pada dataset, pembagian dataset menggunakan metode *cross validation*, melatih dan menguji model kemudian mengevaluasi hasil peforma kedua algoritma.

### *A. Dataset*

Dalam penelitian ini membutuhkan sebuah dataset yang akan digunakan untuk melatih dan menguji model. Dataset yang penulis gunakan adalah data penilaian kredit Australia. Dataset ini didapatkan dari situs *UCI Machine Learning Repository*. Nama kolom dan nilai pada dataset telah disamarkan untuk melindungi kerahasiaan data asli serta diubah menjadi data numerik.




Gbr. 4 Proporsi Kelas Dataset

Dataset ini berisi data riwayat pengajuan kredit yang terdiri dari 15 kolom dan 690 baris dengan proporsi kelas seperti pada gambar 4.

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 22.08 | 11.46 | 2 | 4 | 4 | 1.585 | 0 | 0 | 0 | 1 | 2 | 100 | 1213 | 0 |
| 0 | 22.67 | 7.00 | 2 | 8 | 4 | 0.165 | 0 | 0 | 0 | 0 | 2 | 160 | 1 | 0 |
| 0 | 29.58 | 1.75 | 1 | 4 | 4 | 1.250 | 0 | 0 | 0 | 1 | 2 | 280 | 1 | 0 |
| 0 | 21.67 | 11.50 | 1 | 5 | 3 | 0.000 | 1 | 1 | 11 | 1 | 2 | 0 | 1 | 1 |
| 1 | 20.17 | 8.17 | 2 | 6 | 4 | 1.960 | 1 | 1 | 14 | 0 | 2 | 60 | 159 | 1 |

Gbr. 5 Sampel Dataset

Sampel dataset yang penulis gunakan dapat dilihat pada gambar 5 di atas.

### *B. Preprocessing*

Pada tahap ini mencakup beberapa hal yang dilakukan untuk mengolah data sehingga dapat digunakan untuk melatih model dengan baik.

1. Menambahkan Nama Kolom

   Dataset yang penulis gunakan tidak memiliki nama kolom dikarenakan nama kolom sengaja dihilangkan untuk melindungi kerahasiaan data, sehingga perlu ditambahkan nama kolom secara manual agar algoritma dapat mempelajari data dengan baik. Penulis menambahkan kolom dengan nama A1 hingga A15, dimana A15 merupakan kolom label. Hasil dataset yang sudah siap digunakan seperti pada gambar 6 di bawah.

| A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | A11 | A12 | A13 | A14 | A15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 22.08 | 11.46 | 2 | 4 | 4 | 1.585 | 0 | 0 | 0 | 1 | 2 | 100 | 1213 | 0 |
| 0 | 22.67 | 7.00 | 2 | 8 | 4 | 0.165 | 0 | 0 | 0 | 0 | 2 | 160 | 1 | 0 |
| 0 | 29.58 | 1.75 | 1 | 4 | 4 | 1.250 | 0 | 0 | 0 | 1 | 2 | 280 | 1 | 0 |
| 0 | 21.67 | 11.50 | 1 | 5 | 3 | 0.000 | 1 | 1 | 11 | 1 | 2 | 0 | 1 | 1 |
| 1 | 20.17 | 8.17 | 2 | 6 | 4 | 1.960 | 1 | 1 | 14 | 0 | 2 | 60 | 159 | 1 |

Gbr. 6 Sampel Dataset Dengan Kolom

2. Normalisasi Data

   Dikarenakan nilai-nilai pada dataset memiliki rentang yang cukup bervariasi antar variabelnya sehingga perlu

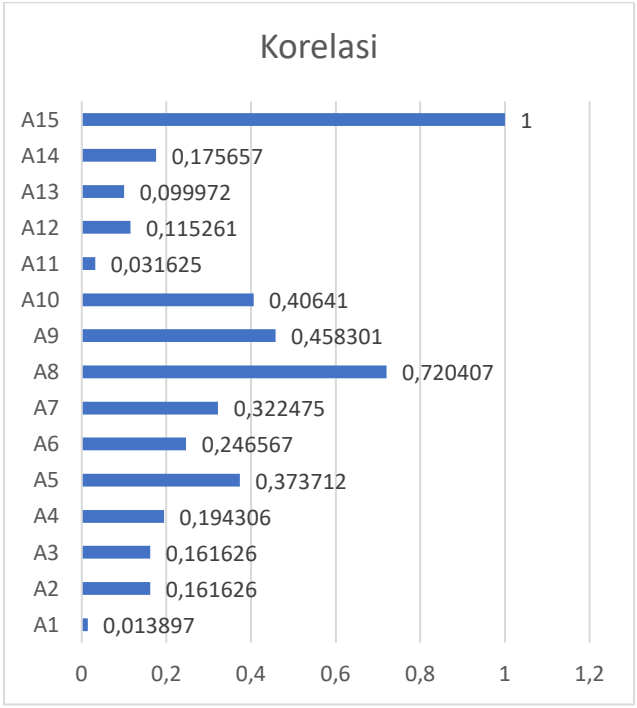
dilakukan normalisasi data agar memiliki rentang yang sama. Hal ini bertujuan agar data yang digunakan tidak memiliki penyimpangan yang besar dan tidak ada lagi satu variabel yang mendominasi variabel data lainnya. Dalam normalisasi data ini penulis menggunakan metode *Min Max Scaler*. Metode ini memungkinkan semua nilai pada dataset memiliki rentang nilai antara 0 dan 1. Hasil setelah dilakukan normalisasi data dapat direpresentasi pada gambar 7 berikut.

| A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | A11 | A12 | A13 | A14 | A15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.0 | 0.125 | 0.409 | 0.5 | 0.231 | 0.375 | 0.056 | 0.0 | 0.0 | 0.000 | 1.0 | 0.5 | 0.05 | 0.012 | 0.0 |
| 0.0 | 0.134 | 0.250 | 0.5 | 0.538 | 0.375 | 0.006 | 0.0 | 0.0 | 0.000 | 0.0 | 0.5 | 0.08 | 0.000 | 0.0 |
| 0.0 | 0.238 | 0.062 | 0.0 | 0.231 | 0.375 | 0.044 | 0.0 | 0.0 | 0.000 | 1.0 | 0.5 | 0.14 | 0.000 | 0.0 |
| 0.0 | 0.119 | 0.411 | 0.0 | 0.308 | 0.250 | 0.000 | 1.0 | 1.0 | 0.164 | 1.0 | 0.5 | 0.00 | 0.000 | 1.0 |
| 1.0 | 0.097 | 0.292 | 0.5 | 0.385 | 0.375 | 0.069 | 1.0 | 1.0 | 0.209 | 0.0 | 0.5 | 0.03 | 0.002 | 1.0 |

Gbr. 7 Dataset Setelah Normaliasi

3. Feature Selection

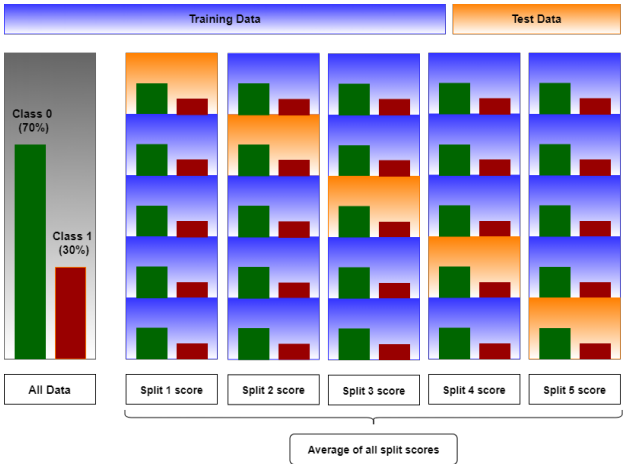
*Feature selection* merupakan proses memilih variabel yang paling relevan. Hal ini dapat meningkatkan peforma model serta mempercepat waktu *training*. Proses *feature selection* dilakukan dengan membandingkan korelasi antara variabel *input* dan variabel *output*. Variabel *input* yang memiliki korelasi rendah akan dihilangkan, sehingga akan tersisa beberapa variabel yang berkualitas. Pada dataset yang penulis gunakan, korelasi antar variabel direpresentasikan pada gambar 8. Korelasi yang sangat kuat ditandai dengan nilai lebih dari 0,5. Namun dikarenakan pada dataset ini variabel yang memiliki nilai korelasi di atas 0,5 hanya ada satu variabel, maka penulis memutuskan hanya akan menggunakan variabel mulai dari yang memiliki tingkat korelasi sedang yaitu dengan nilai di atas 0,3. Sehingga nantinya variabel yang akan digunakan hanya tersisa 5 variabel yaitu A5, A7, A8, A9 dan A10.




Gbr. 8 Korelasi Variabel

*C. Pembagian Dataset*

Dataset dibagi menggunakan *stratified k-fold cross validation* menjadi 5 pola pembagian data *training* dan data *testing* seperti pada gambar 9. Hal ini bertujuan supaya proses pelatihan model dapat dilakukan secara merata sehingga bisa mengurangi bias dan *overfitting* serta dengan menggunakan metode ini dapat merepresentasikan peforma model dengan baik.




Gbr. 9 Ilustrasi Stratified Kfold Cross Validation

*D. Training dan Testing Model*

Setelah dilakukan pembagian dataset menjadi beberapa pola pembagian, data *training* tersebut digunakan untuk melatih kedua model yaitu algoritma *XGBoost* dan algoritma *logistic regression.* Dalam penelitian ini parameter kedua model hanya dilakukan sedikit modifikasi seperti yang tertera pada tabel 2 berikut ini, selebihnya menggunakan parameter sesuai dengan bawaan *library sklearn* pada *python.*

TABEL 2 NILAI PARAMETER

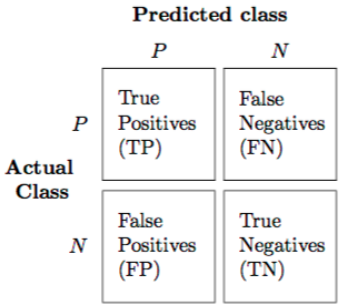
| Algoritma | Parameter |
|---|---|
| *XGBoost* | *Scale_pos_weight* = 1.25 |
| *Logistic Regression* | *random_state* = 11<br>*max_iter* = 10000<br>*class_weight* = *'balanced'* |

Diperlukan melakukan *tuning* pada parameter pembobotan kelas agar model dapat memiliki peforma yang baik pada dataset dengan rasio kelas yang tidak seimbang [18] [19]. Berdasarkan dokumentasi *XGBoost* dan *logistic regression*, pada *scale_pos_weigth* diisi dengan nilai rasio kelas negatif terhadap kelas positif dan *class_weight* diisi dengan nilai '*balanced*' sehingga model mampu mempelajari dataset yang memiliki rasio kelas yang tidak seimbang dengan baik. Agar model *logistic regression* dapat konvergen, maka maksimal iterasinya perlu dinaikkan. Pada dataset ini model dapat konvergen setelah maksimal iterasinya dinaikkan hingga 10000 pada parameter *max_iter.*

Setelah model dilatih dengan data *training*, kedua model tersebut digunakan untuk memprediksi data *testing* dan dibandingkan dengan label aslinya. Kemudian jumlah benar dan salah dibandingkan dengan *confusion matrix* seperti gambar 10 dan didapatkan perhitungan indikator berikut.

1. *accuracy* $= \frac{TP+TN}{TP+FP+TN+FN}$ (10)
2. *precision* $= \frac{TP}{TP+FP}$ (11)
3. *recall* $= \frac{TP}{TP+FN}$ (12)
4. *F1 Score* $= 2 \times \frac{precision \times recall}{precision + recall}$ (13)

Dikarenakan terdapat 5 pola pembagian data *training* dan data *testing* sehingga diambil hasil rata-ratanya.



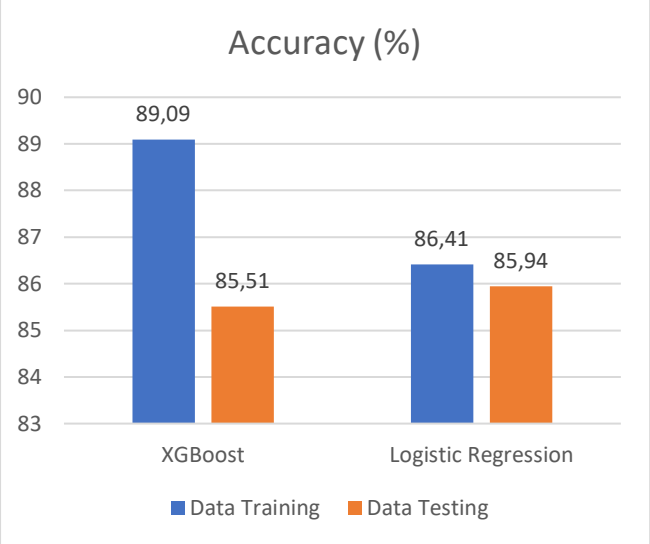

Gbr. 10 Confusion Matrix

Evaluasi model dilakukan dengan melihat perbandingan rata-rata perhitungan indikator *accuracy*, *precision, recall* dan *F1 Score* kedua model. Kemudian dilihat juga selisih perhitungan indikator tersebut pada data *training* dan data *testing* untuk menentukan apakah model tersebut *goodfitting* atau *overfitting*.

## V. HASIL DAN PEMBAHASAN

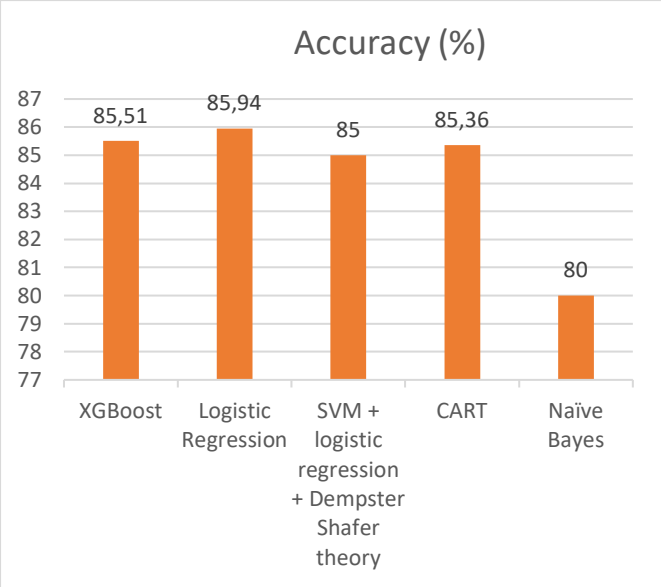
Setelah dilakukan pengujian kemudian menghasilkan rata-rata perhitungan indikator *accuracy, precission, recall* dan *F1 Score* berikut ini.

### *A. Accuracy*




Gbr. 11 *Accuracy*

Akurasi merepresentasikan berapa yang benar dari seluruh data yang diprediksi. Pada gambar 11 terlihat bahwa algoritma *logistic regression* lebih unggul dengan nilai akurasi 85,94%. Sedangkan *XGBoost* sedikit lebih rendah dengan nilai 85,51%. Selain itu selisih peforma antara data *testing* dan data *training* pada *XGBoost* terlihat lebih besar dari pada *logistic regression* yaitu sebesar 3,58%. Sedangkan pada *logistic regression* hanya memiliki selisih 0,47%.

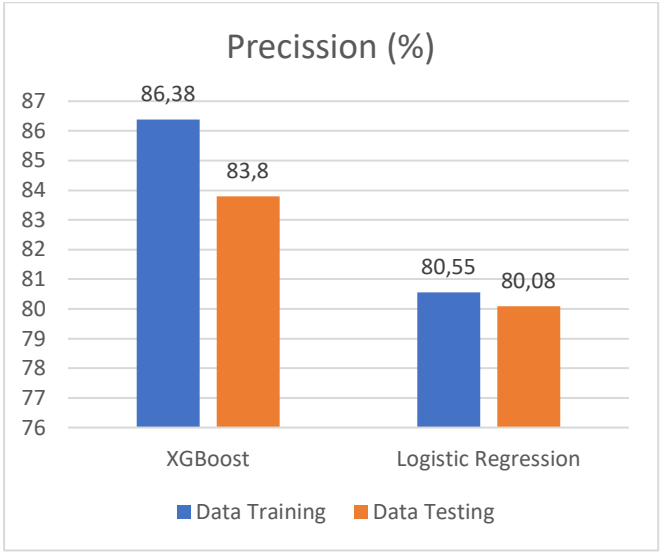



Gbr. 12 Perbandingan Metode

Selain perbandingan dengan *XGBoost*, seperti yang direpresentasikan pada gambar 12 terlihat bahwa *logistic regression* juga lebih unggul di antara metode-metode yang dilatih dan diuji menggunakan dataset yang sama seperti yang penulis gunakan. Pada pengukuran akurasi, *logistic regression* lebih baik daripada semua algoritma.

### *B. Precission*

ISSN: 2477-5126
e-ISSN: 2548-9356

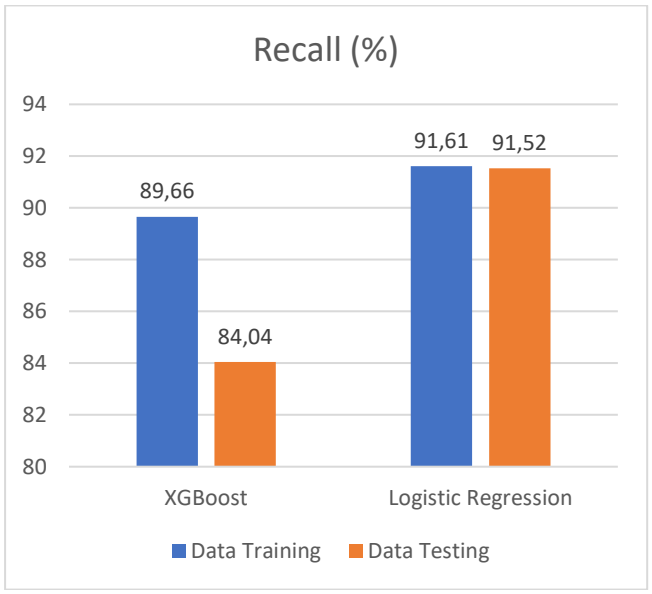



Gbr. 13 *Precission*

*Precission* digunakan untuk melihat berapa nasabah yang benar diprediksi layak dari keseluruhan nasabah yang diprediksi layak. *Precission* bisa dijadikan pertimbangan ketika lebih ingin terhindar dari resiko gagal bayar yang besar.

Pada gambar 13 terlihat bahwa *XGBoost* lebih unggul dengan nilai 83,80%. Sedangkan *logisitc regression* memiliki nilai 80,08%. Namun selisih antara peforma pada data *training* dan data *testing* pada *XGBoost* lebih besar dengan nilai selisih 2,58% dibandingkan dengan *logistic regression* yang memiliki nilai selisih 0,47%. Pada indikator *precission, XGBoost* memiliki peforma yang lebih baik dari pada *logistic regression*

*C. Recall*

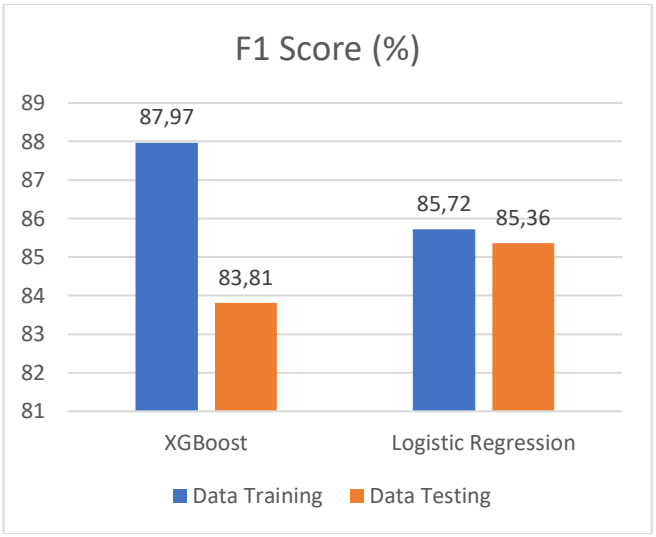



Gbr. 14 *Recall*

*Recall* merepresentasikan berapa nasabah yang benar diprediksi layak dari keseluruhan nasabah yang layak. *Recall* bisa dijadikan pertimbangan bila tidak ingin melewatkan banyak nasabah yang layak, hal ini dapat memberikan keuntungan yang lebih besar serta perputaran arus kas yang cepat.

Pada gambar 14 di atas memperlihatkan bahwa algoritma *logistic regression* lebih unggul dengan nilai 91,52%. Sedangkan *XGBoost* memiliki nilai 84,04%. Nilai selisih peforma antara data *training* dan data *testing* pada *logistic regression* juga lebih kecil dengan nilai 1,39% sedangkan *XGBoost* memiliki selisih 5,62%. Pada indikator *recall, logistic regression* memiliki peforma yang lebih baik daripada *XGBoost*

*D. F1 Score*




Gbr. 15 *F1 Score*

*F1 Score* merupakan perpaduan antara *recall* dan *precission*. *F1 Score* bisa digunakan untuk melihat keseimbangan antara *recall* dan *precission*.

Pada gambar 15 memperlihatkan bahwa nilai *logistic regression* sedikit lebih besar yaitu 85,36% dan memiliki selisih peforma antara data *training* dengan data *testing* hanya sebesar 0,36%. Sedangkan *XGBoost* memiliki nilai 83,81% dengan selisih peforma antara data *training* dan data *testing* sebesar 4,16%. Pada indikator *F1 Score, logistic regression* memiliki peforma lebih baik daripada *XGBoost*

## VI. KESIMPULAN

Berdasarkan penelitian yang telah dilakukan, didapatkan kesimpulan bahwa algoritma *logistic regression* menghasilkan *accuracy* 85,94%; *precission* 80,08%; *recall* 91,52% dan *F1 Score* 85,36% dengan selisih peforma pada data *training* dan data *testing* memiliki nilai *accuracy* 0,47%; *precission* 0,47%; *recall* 1,39%; *F1 Score* 0,36%. Sedangkan *XGBoost* menghasilkan *accuracy* 85,51%; *precission* 83,80%; *recall* 84,04% dan *F1 Score* 83,81% dengan selisih peforma pada data *training* dan data *testing* memiliki nilai *accuracy* 3,58%; *precission* 2,58%; *recall* 5,62%; *F1 Score* 4,16%.

*Logistic regression* lebih unggul pada indikator *accuracy, recall* dan *F1 Score* sedangkan *XGBoost* hanya unggul pada indikator *precission*. Selain itu *XGBoost* lebih *overfitting* daripada *logistic regression*. Berdasarkan hal-hal tersebut dapat diambil kesimpulan bahwa algoritma terbaik untuk penilaian kredit adalah *logistic regression*.

## VII. SARAN

Berdasarkan penelitian yang telah dilakukan, penulis memiliki saran-saran berikut ini.

1. Menggunakan metode lain atau menggabungkan dengan beberapa metode dengan harapan dapat memberikan peforma yang lebih baik lagi.

2. Mencoba beberapa variasi *hyperparameter tunning*.
3. Melakukan *error analysis.*
4. Penelitian ini dapat dijadikan sebagai referensi bagi peneliti berikutnya maupun institusi keuangan dalam menilai kelayakan nasabah untuk diberikan pinjaman.

## UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada pihak yang membantu ataupun memberikan dukungan terkait dengan penelitian yang dilakukan.

## DAFTAR PUSTAKA

*Journal Article*

[1] F. Yang, Y. Qiao, C. Huang, S. Wang, and X. Wang, "An Automatic Credit Scoring Strategy (ACSS) using memetic evolutionary algorithm and neural architecture search," *Appl. Soft Comput.*, vol. 113, 2021, doi: 10.1016/j.asoc.2021.107871.

[2] D. Tripathi, D. R. Edla, R. Cheruku, and V. Kuppili, "A novel hybrid credit scoring model based on ensemble feature selection and multilayer ensemble classification," *Comput. Intell.*, vol. 35, no. 2, pp. 371–394, 2019, doi: 10.1111/coin.12200.

[3] D. Boughaci, A. A. K. Alkhawaldeh, J. J. Jaber, and N. Hamadneh, "Classification with segmentation for credit scoring and bankruptcy prediction," *Empir. Econ.*, vol. 61, no. 3, pp. 1281–1309, 2021, doi: 10.1007/s00181-020-01901-8.

[4] J. P. Barddal, L. Loezer, F. Enembreck, and R. Lanzuolo, "Lessons learned from data stream classification applied to credit scoring," *Expert Syst. Appl.*, vol. 162, no. July, p. 113899, 2020, doi: 10.1016/j.eswa.2020.113899.

[5] V. Kuppili, D. Tripathi, and D. Reddy Edla, "Credit score classification using spiking extreme learning machine," *Comput. Intell.*, vol. 36, no. 2, pp. 402–426, 2020, doi: 10.1111/coin.12242.

[6] D. Şen, C. Ç. Dönmez, and U. M. Yıldırım, "A Hybrid Bi-level Metaheuristic for Credit Scoring," *Inf. Syst. Front.*, vol. 22, no. 5, pp. 1009–1019, 2020, doi: 10.1007/s10796-020-10037-0.

[7] Y. Wang and X. S. Ni, "A XGBoost risk model via feature selection and Bayesian hyper-parameter optimization," *Int. J. Database Manag. Syst.*, vol. 11, no. 1, pp. 1–17, Jan. 2019, doi: https://doi.org/10.48550/arXiv.1901.08433.

[8] J. Chen, F. Zhao, Y. Sun, and Y. Yin, "Improved XGBoost model based on genetic algorithm," *Int. J. Comput. Appl. Technol.*, vol. 62, no. 3, p. 240, 2020, doi: 10.1504/IJCAT.2020.106571.

[9] D. J. Foster, S. Kale, H. Luo, M. Mohri, and K. Sridharan, "Logistic Regression: The Importance of Being Improper," *Proc. Mach. Learn. Res.*, vol. 75, pp. 1–42, Mar. 2018, [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sam.11428?af=R.

[10] D. Wang and Z. Zhang, "Credit Scoring Using Information Fusion Technique," in *2018 7th International Conference on Digital Home (ICDH)*, Nov. 2018, pp. 154–159, doi: 10.1109/ICDH.2018.00036.

[11] Hermawan and Yoannita, "Komparasi Metode Evaluasi Pada Credit Scoring Data Mining," *Jtksi*, vol. 01, no. 02, pp. 22–25, 2018.

[12] M. H. Rifqo and A. Wijaya, "Implementasi Algoritma Naive Bayes Dalam Penentuan Pemberian Kredit," *Pseudocode*, vol. 4, no. 2, pp. 120–128, 2017, doi: 10.33369/pseudocode.4.2.120-128.

[13] J. Li, H. Liu, Z. Yang, and L. Han, "A Credit Risk Model with Small Sample Data Based on G-XGBoost," *Appl. Artif. Intell.*, vol. 35, no. 15, pp. 1550–1566, 2021, doi: 10.1080/08839514.2021.1987707.

[14] A. Ibrahem Ahmed Osman, A. Najah Ahmed, M. F. Chow, Y. Feng Huang, and A. El-Shafie, "Extreme gradient boosting (Xgboost) model to predict the groundwater levels in Selangor Malaysia," *Ain Shams Eng. J.*, vol. 12, no. 2, pp. 1545–1556, 2021, doi: 10.1016/j.asej.2020.11.011.

[15] J. Zhou, Y. Qiu, S. Zhu, D. J. Armaghani, M. Khandelwal, and E. T. Mohamad, "Estimation of the TBM advance rate under hard rock conditions using XGBoost and Bayesian optimization," *Undergr. Sp.*, vol. 6, no. 5, pp. 506–515, 2021, doi: 10.1016/j.undsp.2020.05.008.

[16] A. S. Hess and J. R. Hess, "Logistic regression," *Transfusion*, vol. 59, no. 7, pp. 2197–2198, Jul. 2019, doi: 10.1111/trf.15406.

[17] S. Sperandei, "Understanding logistic regression analysis," *Biochem. Medica*, vol. 24, no. 1, pp. 12–18, 2014, doi: 10.11613/BM.2014.003.

[18] I. Savin *et al.*, "Healthcare-associated ventriculitis and meningitis in a neuro-ICU: Incidence and risk factors selected by machine learning approach," *J. Crit. Care*, vol. 45, pp. 95–104, 2018, doi: 10.1016/j.jcrc.2018.01.022.

[19] D. Paper, *Hands-on Scikit-Learn for Machine Learning Applications*. 2020.