ANALISA TERJADINYA *OVERFITTING* DAN *UNDERFITTING* PADA ALGORITMA *NAIVE BAYES* DAN *DECISION TREE* DENGAN TEKNIK *CROSS VALIDATION*

Wildan Attariq Firmansyach, Umi Hayati, Yudhistira Arie Wijaya
Program Studi Teknik Informatika, STMIK IKMI Cirebon
Jl. Perjuangan No. 10B, Karyamulya Cirebon, Indonesia
wildanattariq@gmail.com

ABSTRAK

Perkembangan digital dapat meringankan aktivitas manusia dalam menggunakan, serta mengelola informasi. Suatu informasi berdasarkan kumpulan data dengan dilakukannya proses data mining. Banyak masalah yang terjadi pada data mining dikarenakan hasil yang diperoleh dipengaruhi oleh adanya ketidak seimbangan data sehingga terjadinya *overfitting* dan *underfitting* biasanya disebabkan oleh data yang berlebihan atau data yang kurang sehingga model yang dihasilkan tidak bisa menangkap pola dari data yang digunaan. Tujuan dari penelitian ini mendapatkan hasil akurasi terbaik melalui perbandingan yang diperoleh supaya tidak terjadinya *overfitting* dan *underfitting* dalam hasil akurasi, untuk mengimplementasikan hasil akurasi dari *machine learning* dalam meningkatkan kepastian dan menunjukan hasil keputusan, menggunakan sampel data Status Kelulusan Siswa SMA Jurusan IPA Jalur SNMPTN Di Indonesia yang dihimpun dari periode tahun 2022, data berjumlah 4.030 data dan 10 atribut. Metode yang digunakan untuk mendapatkan hasil evaluasi terbaik menggunakan algoritma *Naïve Bayes* dan *Decision Tree* dengan teknik *Cross Validation.* Hasil dari penelitian ini yaitu algoritma *Naive Bayes* mendapatkan accuracy score sebesar 63%, serta algoritma *Decision Tree* mendapatkan accuracy score sebesar 96%, dengan hasil accuracy score yang diperoleh algoritma terbaik yaitu *Decision Tree* dengan memakai teknik *Cross Validation.*

Kata kunci: *Naïve Bayes, Decision Tree, Overfitting, Underfitting, Cross Validation, Phyton*

1. PENDAHULUAN

Data mining mempunyai kemampuan untuk menemukan suatu informasi yang menarik dalam data terpilih dengan menggunakan *teknik* dan metode tertentu. Terdapat berbagai macam teknik, metode, dan algoritma yang digunakan dalam data mining, dan pemilihan yang tepat [1]. Masalah pada dataset yang tidak seimbang dalam *Data Mining* juga sering terjadi diberbagai aplikasi seperti pada prediksi perangkat lunak [2]. Akurasi *Machine learning* ditentukan oleh metrik kinerja dan keakuratan metrik yang biasa digunakan untuk mengukur kinerja klasifikasi. Nilai target yang diprediksi untuk kasus data tertentu seperti yang diperkirakan dan pengukuran kinerja model *Machine Learning* pada kumpulan data pengujian [3]. Masalah utama dalam mengklasifikasikan kumpulan data dengan kelas yang tidak seimbang adalah menentukan metrik kinerja yang paling tepat. Ketidakseimbangan data dapat secara signifikan memengaruhi pada nilai akurasi dalam pemodelan klasifikasi [4]. Salah satu algoritma *Classification Machine Learning* yang paling populer yaitu *Naive Bayes* yang mengandalkan teori probabilitas serta statistik dan *Decision Tree* mengarah ke kelas data untuk membuat pohon keputusan untuk memilih variabel yang memungkinkan untuk diterapkan proses klasifikasi [5].

Penelitian *terdahulu* yang berkaitan dengan Algoritma *Naive Bayes* dan *Decision Tree* telah banyak penelian yang dilakukan. seperti pada penelitian yang telah diuraikan oleh Ayudhitama & Pujianto yang komparasi algoritma yang menggunakan 4 algoritma data mining menggunakan aplikasi *rapidminer* yaitu *Naïve Bayes, K-Nearest Neighbor* (KNN), *Decision Tree* dan *Neural Network*. *Dataset* yang digunakan yaitu *Indian Liver Patient Dataset* (ILPD) dari website *UCI Machine Learning Repository*. Penelitian yang telah dilakukan tersebut mendapatkan hasil algoritma Naïve Bayes memiliki akurasi 55,42%, algoritma *K-Nearest Neigbor* memiliki akurasi 66,03%, algoritma *Decision Tree* memiliki akurasi 72,74%, dan algoritma *Neural Network* memiliki akurasi 69,64% [6]. Penelitian terkait lainnya dilakukan oleh Nurajijah & Riana tetang klasifikasi data histori pinjaman nasabah koperasi syariah menggunakan algoritma *Naïve Bayes, Decision Tree* dan *Support Vector Machine* (SVM) dengan tujuan memprediksi kualitas calon nasabah selanjutnya. Hasil penelitian menunjukkan bahwa akurasi algoritma *Naïve Bayes* adalah 77,29%, *Decision Tree* adalah 89,02%, dan akurasi yang tertinggi dicapai oleh *Support Vector Machine* (SVM) yaitu sebesar 89,86%. [7].

Hasil evaluasi yang dihasilkan tidak memvalidasi akurasi algoritma dengan meningkatkan dan mengubah masing-masing pembagian data dalam *algoritma Machine Learning* (ML). Proses ini sangat penting dalam setiap algoritma *Machine Learning* (ML) untuk menghindari *Underfitting* dan *Overfitting* [8]. Pada penelitian ini dilakukan pembagian *fold* mengimplementasikan teknik *K-Fold Cross Validation* untuk membuat pola pengambilan data

dengan aplikasi *Google Colaboratory* dan menggunakan bahasa pemrograman *phyton* untuk mengetahui penelitian yang dihasilkan tersebut terhidar dari permasalahan *Underfitting* atau *Overfitting*. Teknik *K-Fold Cross Validation* akan melatih model dengan memakai *fold* pertama sebagai *train data* dan sisa *fold* menjadi data uji, proses ini akan diulang sebanyak *fold* yang sudah ditentukan yang berjumlah lima atau sepuluh kali, dengan setiap *fold* digunakan sekali sebagai data uji. Setelah itu, hasil dari masing-masing iterasi akan dihitung rata-ratanya untuk memberikan nilai performa model yang dihasilkan.

2. TINJAUAN PUSTAKA

2.1. Algoritma *Naïve Bayes*

Naïve Bayes adalah salah satu algoritma pembelajaran mesin dan *data mining* yang paling efektif dan efisien. Meskipun asumsi bahwa atribut dalam data adalah independen, kinerja klasifikasi *Naïve Bayes* tetap cukup tinggi. Meskipun asumsi independensi atribut jarang terjadi dalam data yang sebenarnya, jika dilanggar, algoritma tetap dapat menghasilkan hasil klasifikasi yang baik. [9]. *Naive Bayes* merupakan implementasi Algoritma pembelajaran mesin probabilistik menggunakan teknik perhitungan teorema bayes [10]. Berikut ini adalah bentuk umum dari teorema Bayes [11]:

$$P(H|X)^{\wedge} = \frac{p(H|X).P(H)}{P(X)} \quad . \quad (1)$$

Keterangan rumus:
X = Data class belum validasi
H = Data class spesifik
P(H|X)^= Probabilitas berdasar kondisi
P(H) = Probabilitas
P(X|H) = Probabilitas kondisi pada hipotesis
P(X) = Probabilitas H

2.2. Algoritma *Decision Tree*

Decision Tree merupakan salah satu metode yang banyak digunakan dalam *Machine Learning*, *Image Processing*, dan *Pattern Identification*. Model ini terdiri dari serangkaian tes dasar yang dilakukan secara berurutan dan efisien, di mana fitur numerik dibandingkan dengan nilai ambang pada setiap pengujian. Aturan konseptual pada *Decision Tree* lebih mudah dibangun daripada menggunakan bobot numerik dalam jaringan saraf, yang menghubungkan antara node. Model *Decision Tree* terdiri dari node dan cabang, dimana setiap simpul merepresentasikan fitur dalam kategori yang akan diklasifikasikan, dan setiap subset mendefinisikan nilai yang dapat diambil oleh *node* [12]. Perhitungan nilai *entropy* menjadi perhitungan awal karena hasil perhitungan tersebut akan digunakan untuk perhitungan selanjutnya. Untuk mendapatkan nilai *entropy*, dihitung jumlah data *True* dan *False* pada data training. Semakin tinggi nilai *entropy*, semakin besar potensi untuk meningkatkan proses klasifikasi. Setelah melakukan perhitungan information gain untuk setiap fitur, nilai information gain tertinggi akan dipilih sebagai akar simpul *Decision Tree*. [13]. Penggunaan rumus *entropy*, seperti yang terdapat dalam rumus 2, memungkinkan kita untuk menghitung nilai entropy dan information gain, yang pada gilirannya dapat membantu menentukan seluruh struktur dari pohon keputusan [14]. Dan rumus *information gain* seperti pada rumus 3 [15].

$$Entropy\ (S) = \sum_{i=1}^{n} pi * log_2 pi \quad (3)$$

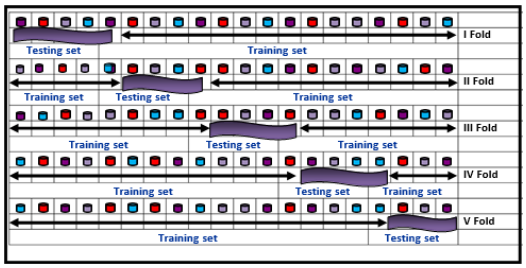
Keterangan Rumus:
S = Himpunan kasus
A = Atribut
N = Jumlah dari patisi S
Pi = Proporsi dari Si terhadap S

$$Gain(S, A) = Entropy(S) \sum_{k=0}^{n} \frac{|Si|}{|S|} + Entropy(Si) \quad (4)$$

Dengan keterangan:
S = himpunan permasalahan
A = Atribut
n = jumlah sebagian atribut A
|Si| = jumlah masalah pada partisi ke-i
|S| = jumlah masalah dalam S

2.3. *Cross Validation*

Metode *Cross Validation* digunakan untuk mempercepat waktu komputasi dalam membangun model, sambil tetap memastikan bahwa hasil estimasi model yang diperoleh tetap akurat. *Cross validation*, atau rotasi estimasi, adalah suatu teknik validasi yang digunakan untuk mengevaluasi hasil analisis. Teknik cross validation umumnya digunakan untuk memprediksi tingkat keakuratan suatu model prediktif, dan salah satu teknik *cross validation*, di mana data dibagi menjadi K bagian set data dengan jumlah data yang sama [16]. Metode *K-Fold Cross Validation* digunakan untuk memperkirakan kesalahan prediksi dan mengevaluasi model dengan melakukan beberapa pengujian pada data yang diminta oleh mesin [17]. Dari pembagian *fold* ini bisa dibagikan menjadi 5 sampai dengan 10 fold untuk dijadikan sebagai *data testing* dan *data train* sebagai model evaluasi dari setiap *fold* yang dihasilkan oleh metode *K-Fold Cross Validation* Adapun pembagian menjadi beberapa *fold* bisa dilihat pada gambar 1.



Gambar 1. Teknik Pembagian Fold

2.4. *Overfitting*

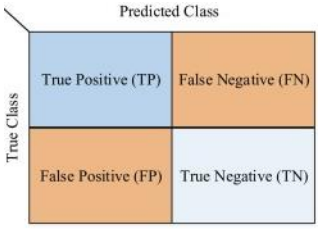
Sementara metode *Machine Learning* dengan karakteristik data yang diamati, kompleksitas yang berlebihan dalam model tersebut dapat menyebabkan *overfitting*. Sesuai dengan namanya, *overfitting* terjadi ketika algoritma peramalan berakhir dengan menggambarkan terlalu baik data sampel, modelnya nanti cuman "menghafal" pola yang dianalisis dari masa lalu, dan dengan demikian tidak akan benar-benar "mempelajari" pola [18].

2.5. *Underfitting*

Pada kasus ini *Underfitting* model memiliki kondisi ciri-ciri *False* yang berlebihan dan akurasi rendah. Underfitting berlawanan dengan *overfitting* karena model underfitting tidak cukup rumit dan terlalu sedikit fokus pada data pelatihan. Akibatnya, membuat pola yang sangat buruk pada saat menggunakan data pelatihan [19].

2.6. *Evaluation*

Evaluation yaitu hasil kinerja dari suatu keseluruhan model dibagi menjadi 2 data [20]. Hasil *evaluation* juga didasari pada *Confusion Matrix* seperti yang ada digambar 2 [21]. *Evaluation* dibutuhkan untuk melihat hasil dengan cara menampilkan *accuracy, precision, recall,* dan *f1_score.* Adapun rumus *accuracy* berada dirumus 4 [22]. rumus *precision, recall*, serta *f1-score* berada dirumus 5,6,7 [23].



Gambar 2. Confusion Matrix

$$Accuracy = \frac{TN + TP}{TN + TP + FP + FN} \quad (4)$$

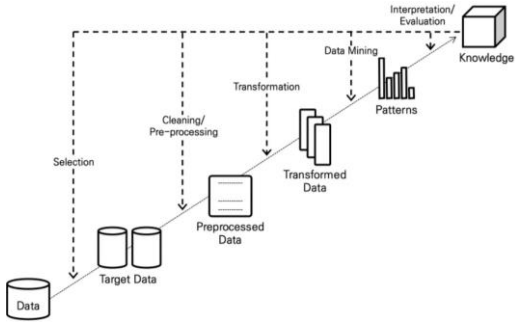
$$precision = \frac{TP}{TP + FP} \quad (5)$$

$$Recall = \frac{TP}{TP + FN} \quad (6)$$

$$f1 - score = 2\frac{Precision\ x\ Recall}{Precision + Recall} \quad (7)$$

3. METODE PENELITIAN

Penelitian ini mengaplikasikan beberapa metode dalam metodologi yang digunakan. Metode penelitian eksperimen diterapkan dalam tahapannya untuk mencapai tingkat akurasi terbaik, sedangkan metode *Knowledge Discovery in Database* (KDD) dipergunakan dalam analisis data. Detail dari tahapan analisis data tertera pada gambar 3 [24].



Gambar 3. Tahapan Knowledge discovery in Database

3.1. *Selection*

Tahap *Selection* merupakan tahap pertama dalam metode analisa knowledge *discovery in database*, dilakukan proses seleksi dan mengolah data yang relevan dari 4.230 *record* data Status Penerimaan Siswa SMA Jurusan IPA Jalur SNMPTN pada tahun 2022. Data tersebut berisi 10 atribut, seperti nama, nilai Bahasa Indonesia, Bahasa Inggris, matematika, kimia, fisika, biologi, sertifikat prestasi, akreditasi sekolah, dan keterangan, data didapat dari hasil kesepakatan dengan komunitas GapaiPTN di Jawa Barat. Pengolahan data dilakukan dengan *Microsoft Excel* dan akan dimasukkan ke *Google Colaboratory* menggunakan *library phyton pandas*, serta diolah dengan *library seaborn* dan *matplotlib.*

3.2. *Preprocessing*

Pada tahap *Preprocessing* menggunakan aplikasi Google Colab *dan library Python.drop (),.info(),* dan. isna*().sum()* untuk memastikan bahwa data yang akan dianalisis memenuhi syarat validitas. Tipe atribut data akan dievaluasi dan transformasi akan dilakukan jika diperlukan demi mencapai hasil yang optimal dalam proses pengklasifikasi machine learning, Attribute yang dihilang yaitu *Attribute* "Nama*".*

3.3. *Transformation*

Pada tahap *Transformation*, tipe atribut data yang bertipe kategori akan dikonversi menjadi tipe numerik. Klasifikasi data mining hanya dapat dilakukan dengan menggunakan tipe atribut numerik, sehingga tipe atribut kategori harus dikonversi menjadi tipe numerik sebelum melangkah ke tahap selanjutnya dalam data mining. Proses transformasi pada data Status Penerimaan Siswa SMA Jurusan IPA Jalur SNMPTN

pada tahun 2022 dilakukan dengan menggunakan source code phyton yaitu *.cat.codes.* Source code menggunakan *.cat.codes* digunakan untuk merubah atribut tipe kategori menjadi tipe numeric. Atribut yang diubah yaitu Keterangan, Akreditasi Sekolah, dan Sertifikat Prestasi, hasil pada transformasi ini seperti pada ditabel 1 *Library Cat Codes Numeric*. Merubah nama atribut yang telah diubah menjadi tipe *numeric*, hasil pada transformasi ini seperti pada ditabel 2 Library *Cat Codes Attribute*.

Tabel 1. Transformation Library Cat Codes Numeric

Nama Atribut	Nama Record	Tranformasi
Serifikat_Prestasi	Tidak Ada	2
	Nasional	1
	Internasional	0
Akreditasi_Sekolah	A	0
	B	1
	C	2
Keterangan	Lulus	0
	Tidak Lulus	1

Tabel 2. Transformation Library Cat Codes Attribute

Nama Atribut	Transformation
Sertifikat_Prestasi	Sertifikat
Akreditasi_Sekolah	Akreditasi
Keterangan	Status

3.4. *Data Mining*

Pada tahap *Data Mining*, analisis dan perhitungan akan dilakukan menggunakan algoritma seperti *Naive Bayes* dan *Decision Tree*. Dalam tahap ini, dataset akan dibagi menjadi 70% dan 30% menggunakan implemtasi dalam bahasa pemrograman phyton yaitu *library Python* sklearn.*model_selection*. Kemudian, dilanjutkan dengan menerapkan library Python untuk algoritma *Naive Bayes*, Decision Tree, dan K-Fold Cross Validation untuk melakukan analisis dan perhitungan pada dataset.

3.5. *Evaluation*

Pada tahap *Evaluation*, akan dilakukan penilaian terhadap aturan yang sudah diperoleh sesuai dengan pola dan tahap yang telah ditentukan sebelumnya. Dalam tahap ini, aplikasi *Google Colaboratory* akan digunakan untuk menentukan hasil *evaluation* yang diperoleh melalui algoritma *Naive Bayes dan Decision Tree*. Hasil evaluasi ini akan ditentukan dengan mengimplementasikan menggunakan *library Python Confusion Matrix* dan teknik *Cross Validation*.

4. HASIL DAN PEMBAHASAN

4.1. *Library Phyton*

Adapun beberapa penggunaan *library* pada gambar 4 untuk menganalisis data dan mencoba model *machine learning*, yaitu sebagai berikut*:*

```
#Library untuk mengolah data
import pandas as pd
#Library untuk melakukan pembagian data menjadi data training dan data testing:
from sklearn.model_selection import train_test_split
#Library untuk visualisasi data:
import seaborn as sns
import matplotlib.pyplot as plt
#Library untuk membangun model machine learning:
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
#Library untuk melakukan evaluasi model machine learning:
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import cross_val_score
from sklearn.metrics import precision_recall_fscore_support, roc_curve, auc
#Library untuk melakukan optimasi model machine learning:
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV
#Library lainnya:
import matplotlib.pyplot as plt
import time, datetime
from sklearn import datasets, model_selection, tree, preprocessing, metrics, linear_model
```

Gambar 4. Library Phyton

4.2. *Import* Data

Import data harus dilakukan untuk memasukan data kedalam Google Colabolatory berdasarkan pada Gambar 5, yaitu sebagai berikut:

```
df2 = pd.read_excel("/content/Data SMA Jurusan IPA DiIndonesia.xlsx")
df2
```

Gambar 5. Import Data

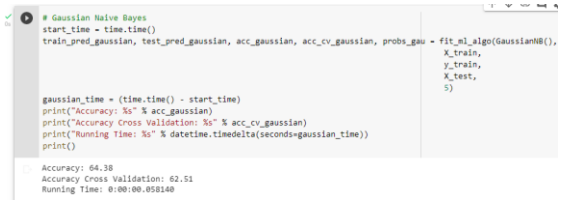
4.3. Model Algoritma

Sebelum melakukan model algoritma yang dihasilkan membuat definisi model seperti pada gambar 6 untuk menentukan data yang akan dioleh sebagai berikut:

```
def fit_ml_algo(algo, X_train, y_train, X_test, crossvalidation):
    # One Pass
    model = algo.fit(X_train, y_train)
    test_pred = model.predict(X_test)
    if (isinstance(algo, (GaussianNB,
                          DecisionTreeClassifier))):
        probs = model.predict_proba(X_test)[:,1]
    else:
        probs = "Not Available"
    acc = round(model.score(X_test, y_test) * 100, 2)
    # CV
    train_pred = model_selection.cross_val_predict(algo,
                                                   X_train,
                                                   y_train,
                                                   cv=crossvalidation,
                                                   n_jobs = -1)
    acc_cv = round(metrics.accuracy_score(y_train, train_pred) * 100, 2)
    return train_pred, test_pred, acc, acc_cv, probs
```

Gambar 6. Definisi Model Klasifikasi

Hasil simulasi model klasifikasi yang menggunakan 9 *Attribute* kecuali "nama" telah dilakukan dan dianalisis dengan menggunakan dua Algoritma utama, yaitu Algoritma *Naive Bayes* dan Algoritma *Decision Tree*. Dalam simulasi tersebut, telah dilakukan perbandingan seperti pada gambar 7 *Algorithm Naïve Bayes* dan gambar 10 *Algorithm Decision Tree.* Adapun hasil *accuracy* yang diperoleh menggunakan algoritma *Naïve Bayes* yaitu sebagai berikut :



```
# Gaussian Naive Bayes
start_time = time.time()
train_pred_gaussian, test_pred_gaussian, acc_gaussian, acc_cv_gaussian, probs_gau = fit_ml_algo(GaussianNB(),
                                                                                                X_train,
                                                                                                y_train,
                                                                                                X_test,
                                                                                                5)
gaussian_time = (time.time() - start_time)
print("Accuracy: %s" % acc_gaussian)
print("Accuracy Cross Validation: %s" % acc_cv_gaussian)
print("Running Time: %s" % datetime.timedelta(seconds=gaussian_time))
print()

Accuracy: 64.38
Accuracy Cross Validation: 62.51
Running Time: 0:00:00.058140
```

Gambar 7. Algorithm Naïve Bayes

Setelah melakukan pengujian data, dilakukan perhitungan akurasi dan perbandingan pada Algoritma *Naive Bayes* untuk mengatasi masalah *overfitting* dan *underfitting* seperti yang terlihat pada Gambar 7. Model klasifikasi Algoritma *Naive Bayes* menunjukkan tingkat akurasi yang baik, baik dengan atau tanpa penggunaan *K-Fold Cross Validation* dengan 5 kali percobaan. Tanpa menggunakan *K-Fold Cross Validation*, diperoleh tingkat akurasi sebesar 64,38%, sementara dengan menggunakan *K-Fold Cross Validation*, tingkat akurasi yang diperoleh adalah 62,51% dengan waktu pembelajaran selama 00,058 detik. Untuk Algoritma *Naive Bayes* yang tidak menggunakan *Cross Validation*, dapat dijelaskan melalui *Classification Report* sebagai berikut:

Hasil Laporan Klasifikasi naive bayes:

	precision	recall	f1-score	support
0	0.58	0.96	0.72	617
1	0.90	0.35	0.50	652
accuracy			0.64	1269
macro avg	0.74	0.65	0.61	1269
weighted avg	0.74	0.64	0.61	1269

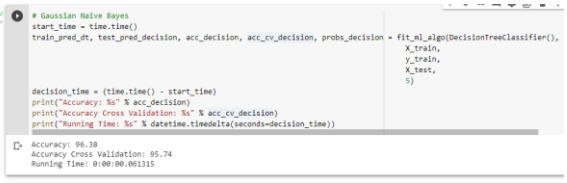
Gambar 8. Classification Report Naïve Bayes

Berdasarkan hasil *Classification Report*, diperoleh hasil perhitungan dari masing-masing model yang dapat dilihat pada Gambar 8 *Classification Report Naive Bayes*. Hasil yang diperoleh pada implementasi library memakai bahasa pemrograman *phyton* dengan memanfaatkan *library Classification Report* dengan mengimplementasikan metode algoritma *Naïve Bayes* menggunakan *K-Fold Cross Validation*, yaitu sebagai berikut :

	precision	recall	f1-score	support
0	0.56	0.94	0.70	1409
1	0.86	0.34	0.49	1552
accuracy			0.63	2961
macro avg	0.71	0.64	0.60	2961
weighted avg	0.72	0.63	0.59	2961

Gambar 9. *Classification Report Naïve Bayes Cross Validation*

Berdasarkan hasil *Classification Report*, diperoleh hasil perhitungan *Precision*, *Recall*, dan *Accuracy* dari masing-masing model yang dapat dilihat pada Gambar 9 *Classification Report Naive Bayes Cross Validation*. Adapun hasil *accuracy* yang diperoleh menggunakan algoritma *Decision Tree* yaitu sebagai berikut :



```
# Gaussian Naive Bayes
start_time = time.time()
train_pred_dt, test_pred_decision, acc_decision, acc_cv_decision, probs_decision = fit_ml_algo(DecisionTreeClassifier(),
                                                                                               X_train,
                                                                                               y_train,
                                                                                               X_test,
                                                                                               5)
decision_time = (time.time() - start_time)
print("Accuracy: %s" % acc_decision)
print("Accuracy Cross Validation: %s" % acc_cv_decision)
print("Running Time: %s" % datetime.timedelta(seconds=decision_time))

Accuracy: 96.38
Accuracy Cross Validation: 95.74
Running Time: 0:00:00.061315
```

Gambar 10. *Algorithm Decision Tree*

Setelah melakukan pengujian data, dilakukan perhitungan akurasi dan perbandingan pada Algoritma *Decision Tree* untuk mengatasi masalah *overfitting* dan *underfitting* seperti yang terlihat pada Gambar 10. Model Algoritma *Decision Tree* diklasifikasikan dan menunjukkan bahwa nilai akurasi baik saat tidak menggunakan *K-Fold Cross Validation* maupun saat menggunakan *K-Fold Cross Validation* dengan 5 kali percobaan. Tanpa menggunakan *K-Fold Cross Validation*, didapatkan nilai akurasi sebesar 96,61%, sementara saat menggunakan *K-Fold Cross Validation*, nilai akurasi yang diperoleh adalah 95,71% dengan waktu pembelajaran selama 00,062 detik. Hasil *Classification Report* pada Algoritma *Decision Tree* yang tidak menggunakan *K-Fold Cross Validation* dapat dijelaskan sebagai berikut:

Hasil Laporan Klasifikasi algoritme Decision Tree:

	precision	recall	f1-score	support
0	0.98	0.95	0.96	617
1	0.95	0.98	0.97	652
accuracy			0.96	1269
macro avg	0.96	0.96	0.96	1269
weighted avg	0.96	0.96	0.96	1269

Gambar 11. *Classification Report Decision Tree*

Berdasarkan hasil *Classification Report* tanpa menggunakan *K-Fold Cross Validation*, dihasilkan perhitungan *Precision*, *Recall*, dan *Accuracy* dari setiap model yang terlihat pada Gambar 11. Sementara itu, hasil *Classification Report* menggunakan Algoritma *Decision Tree* dengan *K-Fold Cross Validation* dapat dijelaskan sebagai berikut:

	precision	recall	f1-score	support
0	0.96	0.95	0.95	1409
1	0.95	0.96	0.96	1552
accuracy			0.96	2961
macro avg	0.96	0.96	0.96	2961
weighted avg	0.96	0.96	0.96	2961

Gambar 12. *Classification Report Naïve Bayes Cross Validation*

Hasil perhitungan *Classification Report* dari setiap model yang digunakan dapat ditemukan di Gambar 12 *Classification Report Naive Bayes Cross Validation* berdasarkan hasil *K-Fold Cross Validation*.

4.4. Hasil Implementasi dengan *Confusion Matrix*

Confusion Matrix adalah sebuah tabel yang digunakan untuk mengevaluasi performa model

klasifikasi dalam machine learning yang menghasilkan output dalam bentuk 2 kelas atau lebih. Tabel ini terdiri dari 4 kombinasi nilai prediksi dan nilai aktual. Untuk mengevaluasi performa masing-masing model klasifikasi tanpa menggunakan metode *K-Fold Cross Validation*, dapat dilihat hasil evaluasinya pada *Confusion Matrix*. Contohnya, Gambar 13 menunjukkan *Confusion Matrix* untuk model *Naïve Bayes*, sementara Gambar 14 menunjukkan *Confusion Matrix* untuk model *Decision Tree*.

```
Hasil Laporan confusion matrix algoritme naive bayes:
[[592  25]
 [427 225]]
```

Gambar 13. *Confusion Matrix Naïve Bayes*

Terdapat di gambar 13 menunjukan hasil *Confusion Matrix Naïve Bayes* tanpa menggunakan *K-Fold Cross Validation* memperoleh *True Positif* adalah 592, *False Positif* adalah 25, *False Negatif* adalah 427, *True Negatif* adalah 225. Hasil evaluasi metode Naïve Bayes berdasarkan sebagai berikut:

$$Accuracy = \frac{225 + 592}{225 + 592 + 25 + 427}$$

$$Precision = \frac{592}{592 + 427}$$

$$Recall = \frac{592}{592 + 25}$$

$$f1 - score = 2\frac{0{,}58\ x\ 0{,}96}{0{,}58 + 0{,}96}$$

Hasil pada perhitungan rumus tersebut *accuracy* mendapatkan 0,64, *precision* mendapatkan 0,58, *recall* mendapatkan 0,96, *f1-score* mendapatkan 0,72.

```
Hasil Laporan confusion matrix algoritme decision tree :
[[585  32]
 [ 14 638]]
```

Gambar 14. *Confusion Matrix Decision Tree*

Pada Gambar 14 menunjukan hasil *Confusion Matrix Naïve Bayes* tanpa menggunakan *K-Fold Cross Validation* memperoleh *True Positif* adalah 585, *False Positif* adalah 32, *False Negatif* adalah 14, *True Negatif* adalah 638. *Accuracy*, *Precision* dan *Recall* dari metode *Decision Tree* adalah sebagai berikut:

$$Accuracy = \frac{638 + 585}{638 + 585 + 32 + 14}$$

$$Precision = \frac{585}{585 + 14}$$

$$Recall = \frac{592}{592 + 25}$$

$$f1 - score = 2\frac{0{,}98\ x\ 0{,}95}{0{,}98 + 0{,}95}$$

Hasil pada perhitungan rumus tersebut *accuracy* mendapatkan 0,96, *precision* mendapatkan 0,98, *recall* mendapatkan 0,95, *f1-score* mendapatkan 0,96.

Tabel 3. Perbandingan tanpa *K-Fold Cross Validation*

Metode	*Accuracy*	*Precision*	*Recall*	*F1-score*
Naïve Bayes	0.64	0.58	0.96	0.72
Decision Tree	0.96	0.98	0.95	0.96

Meskipun Tabel 3 menunjukkan hasil model klasifikasi, namun belum dapat dipastikan apakah terdapat overfitting atau *underfitting*. Oleh karena itu, untuk mendapatkan evaluasi yang lebih akurat, perlu dilakukan evaluasi untuk masing-masing model klasifikasi dengan menggunakan metode *K-Fold Cross Validation*. Hasil evaluasi tersebut dapat dilihat pada Gambar 15 untuk model *Naïve Bayes* dan pada Gambar 16 untuk *Confusion Matrix* pada model *Decision Tree.*

```
Hasil Laporan confusion matrix algoritme Naive Bayes :
[[1325   84]
 [1026  526]]
```

Gambar 15. *Confusion Matrix Naïve Bayes Cross Validation*

Pada Gambar 15 menunjukan hasil *Confusion Matrix Naïve Bayes* menggunakan *K-Fold Cross Validation* memperoleh *True Positif* adalah 1325, *False Positif* adalah 1026, *False Negatif* adalah 84, *True Negatif* adalah 526. *Accuracy*, *Precision* dan *Recall* dari metode *Naïve Bayes* adalah sebagai berikut:

$$Accuracy = \frac{526 + 1325}{526 + 1325 + 1026 + 84}$$

$$precision = \frac{1325}{1325 + 1026}$$

$$Recall = \frac{1325}{1325 + 84}$$

$$f1 - score = 2\frac{0{,}56\ x\ 0{,}94}{0{,}56 + 0{,}94}$$

Hasil pada perhitungan rumus tersebut *accuracy* mendapatkan 0,63, *precision* mendapatkan 0,56, *recall* mendapatkan 0,94, *f1-score* mendapatkan 0,70.

```
Hasil Laporan confusion matrix algoritme Decision Tree :
[[1339   70]
 [  55 1497]]
```

Gambar 16. *Confusion Matrix Decision Tree Cross Validation*

Pada Gambar 16 menunjukan hasil *Confusion Matrix Decision Tree* menggunakan *K-Fold Cross Validation* memperoleh *True Positif* adalah 1339, *False Positif* adalah 55, *False Negatif* adalah 70, *True Negatif* adalah 1497. Ada pun rumus perhitungan pada confusion matrix yaitu sebagai berikut:

$$Accuracy = \frac{1497 + 1339}{1497 + 1339 + 55 + 70}$$

$$precision = \frac{1339}{1339 + 55}$$

$$Recall = \frac{1339}{1339 + 70}$$

$$f1 - score = 2\frac{0{,}96\ x\ 0{,}95}{0{,}96 + 0{,}95}$$

Hasil pada perhitungan rumus tersebut *accuracy* mendapatkan 0,96, *precision* mendapatkan 0,96, *recall* mendapatkan 0,95, *f1-score* mendapatkan 0,95.

Tabel 4. Perbandingan menggunakan *K-Fold Cross Validation*

Metode	***Accuracy***	***Precision***	***Recall***	***F1-score***
Naïve Bayes	0.63	0.56	0.94	0.70
Decision Tree	0.96	0.96	0.95	0.95

Pada tabel 4 menggunakan *K-Fold Cross Validation* dan dibandingkan dengan tabel 3 tanpa menggunakan *K-Fold Cross* Validation menunjukan hasil penurun 0,1 terhadap setiap model yang digunakan menyebabkan penurunan, dikarenakan terjadinya *overfitting* atau *underfitting* pada setiap model yang digunakan, sehingga permasalahan *overfitting* dan *underfitting* bisa terselesaikan dengan menggunakan *K-Fold Cross Validation*.

5. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian ini dalam penerapan metode Algoritma *Naive Bayes* dan Algoritma *Decision Tree* dengan teknik *Cross Validation*. Berdasarkan uraian yang telah disampaikan, maka dapat disimpulkan, sebagai berikut: Pada Algoritma *Naive Bayes* memiliki akurasi sebesar 0,64 dan Algoritma *Decision Tree* memiliki akurasi sebesar 0,97. Sementara itu, pada implementasi teknik *Cross Validation*, Algoritma *Naive Bayes* memperoleh akurasi sebesar 0,63 dan Algoritma *Decision Tree* memperoleh akurasi sebesar 0,96. Terlihat bahwa penggunaan metode *Cross Validation* memiliki pengaruh terhadap akurasi kedua Algoritma tersebut. Berdasarkan hasil analisis, dapat diketahui bahwa implementasi metode *K-Fold Cross Validation* pada Algoritma *Decision Tree* dan *Naive Bayes* menyebabkan penurunan akurasi masing-masing sebesar 1%. Hal ini menunjukkan bahwa sebelumnya kedua Algoritma tersebut mengalami masalah *overfitting* dan *underfitting*. *Overfitting* pada *Decision Tree* disebabkan oleh model yang terlalu kompleks dan terpengaruh noise data latih, sementara underfitting pada *Naive Bayes* disebabkan oleh model yang terlalu sederhana dan tidak dapat memahami pola dari data. Namun, implementasi *K-Fold Cross Validation* dapat memperbaiki masalah tersebut.

Pada penelitian selanjutnya saran yang diperlukan untuk disampaikan adalah sebagai berikut: Klasifikasi algoritma terbaik menggunakan Algoritma *Decision Tree.* Menggunakan metode lain untuk mengatasi masalah *Overfiting* dan *Underfiting* dalam evaluasi data mining. Perbanyak percobaan terhadap algoritma selain *Naive Bayes* dan *Decision Tree* yang digunakan karena setiap algoritma unik dan mempunyai karakteristik yang berbeda-beda.

DAFTAR PUSTAKA

[1] Nanang, "Analisis Perbandingan Algoritma Data Mining Metode Decision Tree C4 . 5 Dengan Naïve Bayes dalam Penjurusan Siswa (Studi Kasus MAN 1 Kota Tangerang Selatan)," *Sci. Sacra J. Sains , Teknol. dan Masy.*, vol. 2, no. 1, pp. 44–61, 2022, [Online]. Available: http://pijarpemikiran.com/index.php/Scientia

[2] R. B. Bahaweres, F. Agustian, I. Hermadi, A. I. Suroso, and Y. Arkeman, "Software defect prediction using neural network based smote," *Int. Conf. Electr. Eng. Comput. Sci. Informatics*, vol. 2020-Octob, no. October, pp. 71–76, 2020, doi: 10.23919/EECSI50503.2020.9251874.

[3] P. Flach, "Performance Evaluation in Machine Learning: The Good, the Bad, the Ugly, and the Way Forward," *Proc. AAAI Conf. Artif. Intell.*, vol. 33, pp. 9808–9814, 2019, doi: 10.1609/aaai.v33i01.33019808.

[4] A. Luque, A. Carrasco, A. Martín, and A. de las Heras, "The impact of class imbalance in classification performance metrics based on the binary confusion matrix," *Pattern Recognit.*, vol. 91, pp. 216–231, 2019, doi: 10.1016/j.patcog.2019.02.023.

[5] S. Farhana, "Classification of Academic Performance for University Research Evaluation by Implementing Modified Naive Bayes Algorithm," *Procedia Comput. Sci.*, vol. 194, pp. 224–228, 2021, doi: 10.1016/j.procs.2021.10.077.

[6] A. P. Ayudhitama and U. Pujianto, "Analisa 4 Algoritma Dalam Klasifikasi Liver Menggunakan Rapidminer," *J. Inform. Polinema*, vol. 6, no. 2, pp. 1–9, 2020, doi: 10.33795/jip.v6i2.274.

[7] N. Nurajijah and D. Riana, "Algoritma Naïve Bayes, Decision Tree, dan SVM untuk Klasifikasi Persetujuan Pembiayaan Nasabah Koperasi Syariah," *J. Teknol. dan Sist. Komput.*, vol. 7, no. 2, pp. 77–82, 2019, doi: 10.14710/jtsiskom.7.2.2019.77-82.

[8] R. Tabares-Soto, S. Orozco-Arias, V. Romero-

Cano, V. S. Bucheli, J. L. Rodríguez-Sotelo, and C. F. Jiménez-Varón, "A comparative study of machine learning and deep learning algorithms to classify cancer types based on microarray gene expression data," *PeerJ Comput. Sci.*, vol. 2020, no. 4, pp. 1–22, 2020, doi: 10.7717/peerj-cs.270.

[9] I. Romli, T. Pardamean, S. Butsianto, T. N. Wiyatno, and E. Bin Mohamad, "Naive Bayes Algorithm Implementation Based on Particle Swarm Optimization in Analyzing the Defect Product," *J. Phys. Conf. Ser.*, vol. 1845, no. 1, 2021, doi: 10.1088/1742-6596/1845/1/012020.

[10] J. A. Josen Limbong, I. Sembiring, K. Dwi Hartomo, U. Kristen Satya Wacana, and P. Korespondensi, "Analisis Klasifikasi Sentimen Ulasan Pada E-Commerce Shopee Berbasis Word Cloud Dengan Metode Naive Bayes Dan K-Nearest Neighbor Analysis of Review Sentiment Classification on E-Commerce Shopee Word Cloud Based With Naïve Bayes and K-Nearest Neighbor Meth," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 9, no. 2, pp. 347–356, 2019, doi: 10.25126/jtiik.202294960.

[11] H. Irsyad and M. R. Pribadi, "Klasifikasi Opini Terhadap Pertanian Sawit (Palm Oil) Indonesia Menggunakan Naïve Bayes," *JATISI (Jurnal Tek. Inform. dan Sist. Informasi)*, vol. 6, no. 2, pp. 230–239, 2020, doi: 10.35957/jatisi.v6i2.182.

[12] B. Charbuty and A. Abdulazeez, "Classification Based on Decision Tree Algorithm for Machine Learning," *J. Appl. Sci. Technol. Trends*, vol. 2, no. 01, pp. 20–28, 2021, doi: 10.38094/jastt20165.

[13] L. Nurellisa and D. Fitrianah, "Analisis Rekomendasi Calon Debitur Motor pada PT.XYZ menggunakan Algortima C 4.5," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 7, no. 4, p. 673, 2020, doi: 10.25126/jtiik.2020742080.

[14] D. Sayhidin, G. Haris, and C. Juliane, "Implementasi Data Mining Tingkat Kepemimpinan Siswa dengan K- Nearest Neighbor , Decision Tree , dan Naïve Bayes," vol. 7, pp. 199–206, 2023, doi: 10.30865/mib.v7i1.5351.

[15] W. Laila, W. Widiarto, A. Wijayanto, and E. Suryani, "Rekomendasi Makanan Bagi Pasien Hiperlipidiemia Berdasarkan Hasil Klasifikasi Menggunakan Metode Naive Bayes dan Decision Tree," *JEPIN (Jurnal Edukasi dan Penelit. Inform.*, vol. 8, no. 2, pp. 328–336, 2022.

[16] M. Y. Ashari, A. C. Fauzan, H. M. Muhamat, and A. C. Fauzan, "Perbandingan Kinerja Sistem Klasifikasi Berbasis K-Fold Cross Validation pada Algoritma Decision Tree ID3 dan C5 . 0," vol. 21, pp. 44–52, 2022.

[17] L. Mardiana, D. Kusnandar, and N. Satyahadewi, "Analisis Diskriminan Dengan K Fold Cross Validation Untuk Klasifikasi Kualitas Air Di Kota Pontianak," *Bimaster*, vol. 11, no. 1, pp. 97–102, 2022.

[18] Y. Peng and M. H. Nagata, "An empirical overview of nonlinearity and overfitting in machine learning using COVID-19 data," *Chaos, Solitons and Fractals*, vol. 139, 2020, doi: 10.1016/j.chaos.2020.110055.

[19] M. Czajkowski and M. Kretowski, "Decision tree underfitting in mining of gene expression data. An evolutionary multi-test tree approach," *Expert Syst. Appl.*, vol. 137, pp. 392–404, 2019, doi: 10.1016/j.eswa.2019.07.019.

[20] L. Putelli, A. E. Gerevini, A. Lavelli, M. Olivato, and I. Serina, "Deep learning for classification of radiology reports with a hierarchical schema," *Procedia Comput. Sci.*, vol. 176, pp. 349–359, 2020, doi: 10.1016/j.procs.2020.08.045.

[21] F. Demir, "Deep autoencoder-based automated brain tumor detection from MRI data," *Artif. Intell. Brain-Computer Interface*, pp. 317–351, Jan. 2022, doi: 10.1016/B978-0-323-91197-9.00013-8.

[22] A. H. Wicaksono *et al.*, "Klasifikasi Siswa Slow Learner Untuk Mendukung Sekolah Algoritma Naïve Bayes Classification of Slow Learner Students To Support Schools in Improving Student Understanding Using the Naïve Bayes," vol. 9, no. 3, pp. 589–596, 2022, doi: 10.25126/jtiik.202295509.

[23] M. Artur, "Review the performance of the Bernoulli Naïve Bayes Classifier in Intrusion Detection Systems using Recursive Feature Elimination with Cross-validated selection of the best number of features," *Procedia Comput. Sci.*, vol. 190, no. 2019, pp. 564–570, 2021, doi: 10.1016/j.procs.2021.06.066.

[24] S. E. Choi, J. Kim, and D. Seo, "Travel patterns of free-floating e-bike-sharing users before and during COVID-19 pandemic," *Cities*, vol. 132, no. August 2022, p. 104065, 2023, doi: 10.1016/j.cities.2022.104065.