Analisa *Fraud* Pada Transaksi Kartu Kredit Menggunakan Algoritma *Random Forest*

Ravina Armiani
Program Studi Ilmu Komputer, Jurusan Sistem Informasi
Universitas Bina Darma
Palembang, Indonesia
ravinaarmiani@gmail.com

Eka Puji Agustini
Program Studi Ilmu Komputer, Jurusan Sistem Informasi
Universitas Bina Darma
Palembang, Indonesia
eka_puji@binadarma.ac.id

Abstract* — Along with the COVID-19 pandemic, the use of credit cards has also increased. Credit cards have become a widely used payment system option because of the ease of service and also the ease with which it can be accessed by anyone. Payment with a credit card can also reduce the transmission of COVID-19 through physical contact. With the increasing use of credit cards, cases of fraud using credit cards have also increased. To overcome the many cases of fraud that occur, banks need efforts or solutions to detect fraud quickly and accurately. Given these problems, a study was conducted to detect fraud using machine learning and a random forest algorithm. The results of the study show that the random forest algorithm produces an accuracy of 90.68%, a precision of 94.74%, a recall of 86.14%, and an F1 score of 90.23% because the overall classification results are between 90 and 100%, so this value is categorized as very good. With this, it can be concluded that the random forest model and algorithm developed can detect fraud that occurs in credit card transactions so as to prevent losses for both consumers and banks.

Keywords: Covid-19, Fraud, Credit Card.

***Abstrak* — Seiring dengan terjadinya pandemi Covid-19 penggunaan kartu kredit pun meningkat, kartu kredit menjadi opsi sistem pembayaran yang banyak digunakan karena kemudahan layanan dan juga kemudahan untuk diakses oleh siapa pun, pembayaran menggunakan kartu kredit juga dapat mengurangi terjadinya penularan Covid-19 melalui kontak fisik. Dengan bertambah banyaknya penggunaan kartu kredit, kasus *fraud* menggunakan kartu kredit pun meningkat. Untuk mengatasi banyaknya kasus *fraud* yang terjadi, pihak perbankan memerlukan upaya atau solusi untuk mendeteksi tindakan *fraud* secara cepat dan akurat. Dengan adanya masalah tersebut maka dilakukan penelitian untuk mendeteksi adanya *fraud* menggunakan *machine learning* dengan algoritma *random forest.* Hasil dari penelitian menunjukkan bahwa algoritma *random forest* menghasilkan akurasi sebesar 90,68%, presisi sebesar 94,74%, recall sebesar 86,14%, dan F1 score sebesar 90,23% karena hasil klasifikasi secara keseluruhan berada diantara 90-100% maka nilai ini dikategorikan sangat baik. Dengan ini dapat disimpulkan bahwa model dan algoritma *random forest* yang dikembangkan dapat mendeteksi *fraud* yang terjadi pada transaksi kartu kredit sehingga mencegah terjadinya kerugian baik bagi konsumen maupun perbankan.**

Kata Kunci*: Covid-19, Fraud,* Kartu Kredit.

PENDAHULUAN

Kartu kredit merupakan sebuah alat pembayaran yang dilakukan melalui jasa dari bank atau perusahaan untuk melakukan transaksi jual beli berupa barang atau jasa [1]. Selain itu, kartu kredit juga dapat digunakan sebagai pengganti dari uang tunai. Sehingga seiring dengan terjadinya pandemi Covid-19 penggunaan kartu kredit pun meningkat, kartu kredit menjadi opsi sistem pembayaran yang banyak digunakan karena kemudahan layanan dan juga kemudahan untuk diakses oleh siapa pun, pembayaran menggunakan kartu kredit juga dapat mengurangi terjadinya penularan Covid-19 melalui kontak fisik. Dengan bertambah banyaknya penggunaan kartu kredit, kasus *fraud* menggunakan kartu kredit pun meningkat. *Fraud* sendiri merupakan suatu perbuatan yang dilakukan secara sengaja agar mendapat keuntungan atau tujuan tertentu [2]. Suatu perbuatan dapat dikatakan *fraud* apabila perbuatan tersebut memiliki korban, dilakukan secara sengaja, korban menuruti semua kemauan pelaku, dan korban mengalami kerugian [3]. Menurut informasi dari bank Mandiri, kondisi ekonomi yang melemah menjadi salah satu faktor yang menyebabkan meningkatnya kasus *fraud*. Kasus *fraud* pada sektor *e-commerce* naik sebanyak 83%, pada sektor jasa keuangan meningkat sebanyak 60%, dan pada perkreditan naik sebanyak 40%. *Fraud* menyerang setiap pengguna digital, mulai dari nasabah, *merchant*, *payment gateway* hingga lembaga keuangan. Sehingga banyak data pribadi yang bocor membuat modus *fraud* semakin berkembang [4].

Untuk mengatasi kasus *fraud* yang muncul selama pandemi industri perbankan perlu upaya dan segera menemukan solusi yang efektif. Salah satu upaya yang dapat dilakukan adalah dengan mendeteksi *fraud* menggunakan *machine learning*. *Machine learning* adalah bagian dari kecerdasan buatan atau *artificial intelligence* yang merupakan program komputer yang mempunyai algoritma untuk mempelajari data sehingga dapat berfikir dan brertindak seperti manusia [5]. Terdapat tiga metode dalam *artificial intelligence* salah satunya adalah *machine learning* yang merupakan kecerdasan buatan yang dilengkapi dengan pemrograman untuk memungkinkan agar sebuah komputer dapat cerdas seperti perilaku manusia sehingga dapat meningkatkan pemahaman dan pengalaman yang diperoleh secara otomatis [6]. Penerapan *machine learning* dalam mendeteksi adanya *fraud* dapat dibuktikan menggunakan penelitian sebelumnya. Salah satunya adalah penelitian yang dilakukan oleh Yoga Religia, Agung Nugroho, dan Wahyu Hadiktristanto menggunakan algoritma *random forest* dan optimasi *bagging* dan GA untuk mendapat model prediksi penerimaan pengajuan pinjaman dengan tingkat akurasi yang tinggi. Dari hasil penelitian menunjukkan bahwa klasifikasi bank marketing dataset menggunakan algoritma *random forest* memperoleh akurasi sebesar 88,30%, AUC (+) sebesar 0,500 dan AUC (-) sebesar 0,000 [7]. Pada penelitian ini penggunaan algoritma *random forest* untuk klasifikasi data

DOI: https://doi.org/10/25047/jtit.v9i2.297

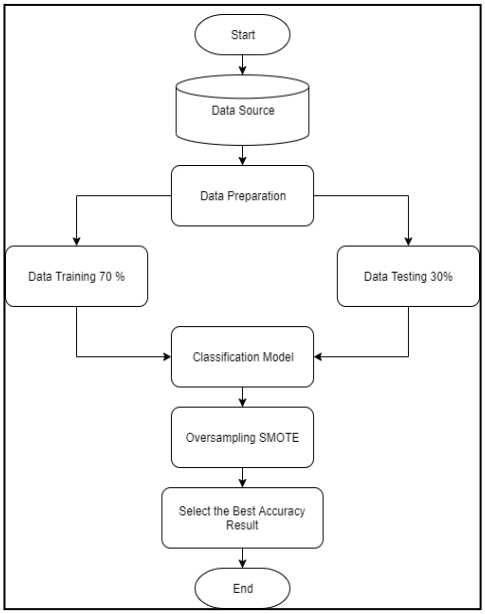
Bank Marketing tidak memerlukan optimasi GA atau *Bagging* untuk menyeimbangkan data karena tanpa GA atau *Bagging* pun hasil klasifikasi yang dilakukan pada algoritma *random forest* sendiri sudah baik dan tidak jauh berbeda dengan memakai GA atau *Bagging*.

Berdasarkan pembahasan pada paragraph sebelumnya penelitian ini akan membahas tentang algoritma *Random Forest* yang menggunakan penyeimbang data yang berbeda dari penelitian sebelumnya yaitu *oversampling SMOTE*. Dalam mendeteksi *fraud* pada transaksi menggunakan kartu kredit. Metode algoritma *random forest* ini digunakan karena menghasilkan persentase kesalahan yang kecil dan menghasilkan akurasi yang cukup tinggi untuk klasifikasi kumpulan data yang sangat besar [8]. Selain itu, sejumlah penelitian telah menunjukkan bahwa algoritma *random forest* bekerja dengan lebih baik serta waktu eksekusi yang cepat membuatnya berguna untuk mengklasifikasikan sejumlah besar data yang *imbalance* [9], [10]. .Penelitian ini dilakukan dengan tujuan untuk mendapatkan akurasi yang nantinya akan dibandingkan dengan penelitian yang sudah ada mana yang hasil akurasinya paling tinggi.

METODE PENELITIAN

Metode yang digunakan dalam project ini adalah *Random Forest*, yaitu metode dari pengembangan *decision tree* dimana pada setiap *decision tree* dilakukan sebuah pelatihan dari sampel individu [11]. *Random forest* menggunakan metode pemisahan biner rekursif untuk mencapai ke node terakhir dalam struktur pohon berbasis pohon kasifikasi dan regresi [12].

Berikut ini adalah bagan alur kegiatan proses pengerjaan *project* Analisis pada Transaksi Kartu Kredit Menggunakan Algoritma *Random Forest* :



Gambar .1. Bagan Pengerjaan Project

Berdasarkan gambar 1 merupakan langkah yang dilakukan dalam proses menganalisa data transaksi kartu kredit menggunakan algoritma *random forest*, yaitu sebagai berikut :

1. *Data Source*. Dataset yang digunakan dalam penelitian ini adalah data transaksi pada bank menggunakan kartu kredit dalam bentuk *JavaScript Object Notation* (JSON).
2. *Data Preparation* adalah proses yang digunakan untuk melakukan perbaikan masalah yang terdapat di dalam data [13]. Persiapan data yang pertama adalah dengan mengubah data yang tadinya dalam bentuk JSON menjadi *Comma Separated Values* (CSV). Kemudian dilanjutkan dengan *cleansing data* yang dilakukan dengan menghapus duplikasi data, mencari data yang tidak konsisten, dan memperbaiki kesalahan data seperti kesalahan pencetakan [14]. Selain itu, proses yang dikenal sebagai *enrichment* digunakan untuk "memperkaya" data uang ada dengan data atau informasi tambahan yang relevan dan diperlukan untuk *Knowledge Discovery in Database* (KDD), seperti data eksternal tambahan [15].
3. *Split Data Training dan Data Testing*. Algoritma dilatih menggunakan data *traning*, sedangkan data *testing* akan dipakai untuk algoritma yang sudah dilatih menggunakan data baru yang sebelumnyya ditemukan agar dapat melihat seberapa baik kinerjanya. 70% dari data dijadikan sebagai data *training* dan 30% untuk data *testing.*
4. *Classification Model*. Algoritma yang akan digunakan untuk mencari model terbaik adalah *Random Forest Classifiers* dan *Decision Tree Classifiers.*
5. *Synthetic Minority Oversampling Technique* (SMOTE). SMOTE akan menghasilkan poin data baru dari kelas minoritas menggunakan instans yang berbeda, sehingga menghasilkan sampel yang mirip dengan instans yang dimiliki tapi bukan salinan aslinya. *Random Oversampling* (ROS) merupakan SMOTE yang akan dipakai.
6. *Select the Best Accuracy Result*. Dari penelitian yang dilakukan akan dianalisa dan kemudian model yang paling akurat yang akan dipilih.

HASIL DAN PEMBAHASAN

Dalam pengerjaan *project* ini, memanfaatkan Google Colab dalam pembuatan *coding* untuk membuat *machine learning* yang menggunakan bahasa pemrograman *Python*, dimana hasil dari pendeteksian *fraud* ini akan ditampilkan dalam bentuk *dashboard* yang dibuat mengguankan Google Data Studio.

Berikut dibawah ini merupakan langkah-langkah yang dilakukan dalam proses pengerjaan *project* Analisis pada Transaksi Kartu Kredit Menggunakan Algoritma *Random Forest* :

4.1. *Data Source.*

Data yang digunakan dalam penelitian ini adalah data transaksi pada bank menggunakan kartu kredit dalam bentuk JSON. Data ini memiliki 641.914 *records* dan 29 *fields.*

DOI: https://doi.org/10/25047/jtit.v9i2.297

```
# Import Data
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# The Data is in JSON Format, so we will use Pandas to Convert to csv
df = pd.read_json(r'/content/drive/MyDrive/Data/transactions.json', lines=True)
df.to_csv (r'/content/drive/MyDrive/Data/data1.csv', index = None)
```

```
df.head()
```

	accountNumber	customerId	creditLimit	availableMoney	transactionDateTime	transactionAmount	m
0	733493772	733493772	5000	5000.00	2016-01-08T19:04:50	111.33	
1	733493772	733493772	5000	4888.67	2016-01-09T22:32:39	24.75	
2	733493772	733493772	5000	4863.92	2016-01-11T13:36:55	187.40	
3	733493772	733493772	5000	4676.52	2016-01-11T22:47:46	227.34	
4	733493772	733493772	5000	4449.18	2016-01-16T01:41:11	0.00	

5 rows × 29 columns

Gambar .2. Dataset

4.2. *Data Preparation.*

Data preparation dilakukan dengan mengubah data yang tadinya berbentuk JSON menjadi berbentuk *Comma Separated Values* (CSV)

```
# The Data is in JSON Format, so we will use Pandas to Convert to csv
df = pd.read_json(r'/content/drive/MyDrive/Data/transactions.json', lines=True)
df.to_csv (r'/content/drive/MyDrive/Data/data1.csv', index = None)
```

Gambar .3. Mengubah Data menjadi CSV

Setelah data diubah menjadi bentuk CSV, dilakukan pengecekan mengenai informasi yang terdapat dalam data.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 641914 entries, 0 to 641913
Data columns (total 29 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   accountNumber           641914 non-null  int64
 1   customerId              641914 non-null  int64
 2   creditLimit             641914 non-null  int64
 3   availableMoney          641914 non-null  float64
 4   transactionDateTime     641914 non-null  object
 5   transactionAmount       641914 non-null  float64
 6   merchantName            641914 non-null  object
 7   acqCountry              641914 non-null  object
 8   merchantCountryCode     641914 non-null  object
 9   posEntryMode            641914 non-null  object
 10  posConditionCode        641914 non-null  object
 11  merchantCategoryCode    641914 non-null  object
 12  currentExpDate          641914 non-null  object
 13  accountOpenDate         641914 non-null  object
 14  dateOfLastAddressChange 641914 non-null  object
 15  cardCVV                 641914 non-null  int64
 16  enteredCVV              641914 non-null  int64
 17  cardLast4Digits         641914 non-null  int64
 18  transactionType         641914 non-null  object
 19  isFraud                 641914 non-null  bool
 20  echoBuffer              641914 non-null  object
 21  currentBalance          641914 non-null  float64
 22  merchantCity            641914 non-null  object
```

Gambar .4. Informasi Data

Berdasarkan gambar 3 diatas terdapat data yang berbentuk boolean, yaitu isFraud, cardPresent dan expirationDataKeyInMatch. Data yang berbentuk boolean tersebut diubah menjadi Integer. Berikut di bawah ini adalah gambar 4 cara mengubah boolean menjadi Integer.

```
# convert boolean into int
df['isFraud'] = df['isFraud'].replace({True: 1, False: 0})
df['cardPresent'] = df['cardPresent'].replace({True: 1, False: 0})
df['expirationDateKeyInMatch'] = df['expirationDateKeyInMatch'].replace({True: 1, False: 0}
df.head()
```

Gambar .5. Mengubah Boolean menjadi Integer

Langkah selanjutnya adalah melakukan *cleansing data* untuk menghapus *missing value*. Berdasarkan gambar 6 di bawah ini terdapat *missing value* dan juga kolom yang nilainya kosong, sehingga harus dilakukan *cleansing data* dengan cara menghapus nilai kolom yang kosong, kolom yang terdapat *missing value*, serta kolom yang nantinya tidak akan digunakan pada pengerjaan *project.*

```
# checking nan
col = df.columns
check_nan = pd.DataFrame({'count': df.isnull().sum(), 'rate': df.isnull().sum()*100/len(df)})
check_null = check_nan.sort_values(by=['rate'],ascending=False)
round(check_null,2)
```

	count	rate
recurringAuthInd	641914	100.00
posOnPremises	641914	100.00
merchantZip	641914	100.00
merchantState	641914	100.00
merchantCity	641914	100.00
echoBuffer	641914	100.00
acqCountry	3913	0.61
posEntryMode	3345	0.52
merchantCountryCode	624	0.10
transactionType	589	0.09
posConditionCode	287	0.04
accountNumber	0	0.00
cardLast4Digits	0	0.00
cardPresent	0	0.00
currentBalance	0	0.00
isFraud	0	0.00
dateOfLastAddressChange	0	0.00
enteredCVV	0	0.00

Gambar .6. Mengecek *Missing Value*

Gambar 7 di bawah ini merupakan sintaks yang akan menghapus kolom yang terdapat *missing value*, yaitu kolom echoBuffer, merchatCity, merchantState, posOnPremises, recurringAuthInd.

```
# Delete column that have NaN values
df = df.drop(['echoBuffer', 'merchantCity', 'merchantState', 'merchantZip', 'posOnPremises', 'recurri
df.head()
```

Gambar .7. Hapus *Missing Value*

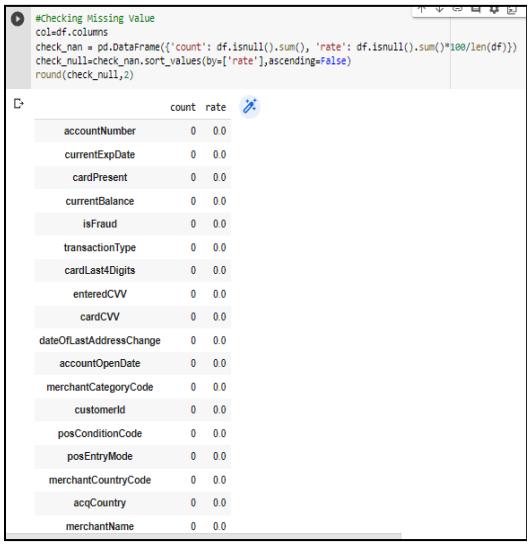
Sedangkan gambar 8 di bawah ini adalah sintaks untuk menghapus isi kolom yang tidak ada nilainya.

```
[17] # deleting empty values
     df = df.dropna()
```

Gambar .8. Hapus Nilai Kosong

Selanjutnya dilakukan pengecekan kembali apakah *missing value* telah dihapus semua, sehingga dapat dilanjutkan ke tahap selanjutnya.

DOI: https://doi.org/10/25047/jtit.v9i2.297



```
#Checking Missing Value
col=df.columns
check_nan = pd.DataFrame({'count': df.isnull().sum(), 'rate': df.isnull().sum()*100/len(df)})
check_null=check_nan.sort_values(by=['rate'],ascending=False)
round(check_null,2)
```

	count	rate
accountNumber	0	0.0
currentExpDate	0	0.0
cardPresent	0	0.0
currentBalance	0	0.0
isFraud	0	0.0
transactionType	0	0.0
cardLast4Digits	0	0.0
enteredCVV	0	0.0
cardCVV	0	0.0
dateOfLastAddressChange	0	0.0
accountOpenDate	0	0.0
merchantCategoryCode	0	0.0
customerId	0	0.0
posConditionCode	0	0.0
posEntryMode	0	0.0
merchantCountryCode	0	0.0
acqCountry	0	0.0
merchantName	0	0.0

Gambar .9. Pengecekan *Missing Value*

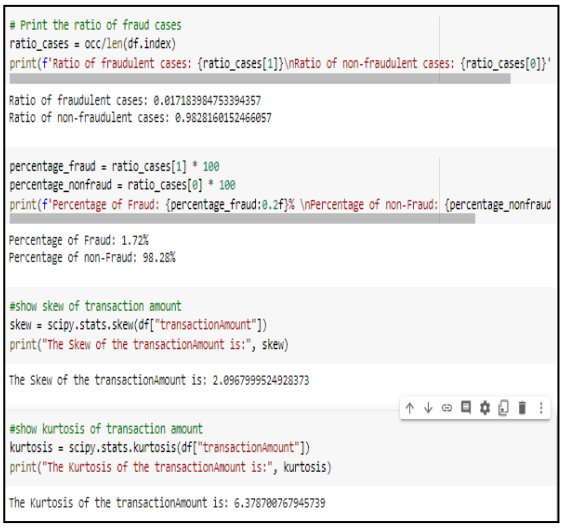
Berdasarkan gambar 9 diatas dapat disimpulakan bahwa semua *missing value* yang terdapat di dalam data telah dihapus semua. Setelah *missing value* telah dihapus, dapat dilakukan pengecekan dan juga visualisasi untuk mengetahui jumlah data transaksi yang terdapat *fraud* dan juga non *fraud*.

```
#is Fraud (variable target)
#this represents whether the transaction is fraud or not
occ = df["isFraud"].value_counts()
occ

0    622954
1     10892
Name: isFraud, dtype: int64
```

Gambar .10. Jumlah *Fraud*

Berdasarkan gambar 10 diatas, dari *output* yang didapat bahwa transaksi berlabel non *fraud* lebih banyak yaitu 622.956 dibandingkan dengan transaksi berlabel *fraud* yaitu 10.892.

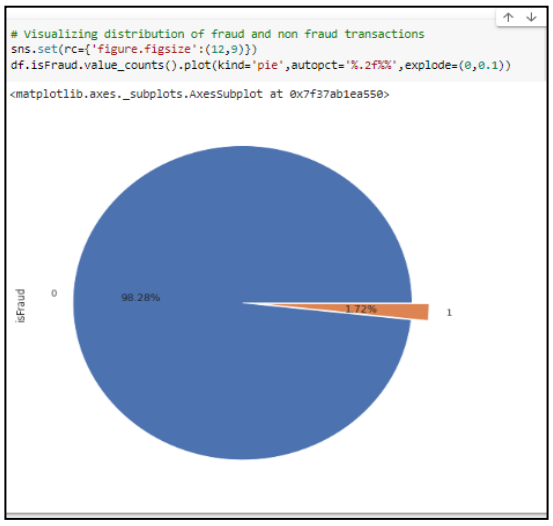
Berikut di bawah ini dilakukan juga pengecekan rasio dari kasus *fraud*, persentase data yang terdapat *fraud* dan non *fraud*, skew dari variabel transactionAmount dan juga kurtosis variabel transactionAmount :



```
# Print the ratio of fraud cases
ratio_cases = occ/len(df.index)
print(f'Ratio of fraudulent cases: {ratio_cases[1]}\nRatio of non-fraudulent cases: {ratio_cases[0]}'

Ratio of fraudulent cases: 0.017183984753394357
Ratio of non-fraudulent cases: 0.9828160152466057

percentage_fraud = ratio_cases[1] * 100
percentage_nonfraud = ratio_cases[0] * 100
print(f'Percentage of Fraud: {percentage_fraud:0.2f}% \nPercentage of non-Fraud: {percentage_nonfraud

Percentage of Fraud: 1.72%
Percentage of non-Fraud: 98.28%

#show skew of transaction amount
skew = scipy.stats.skew(df["transactionAmount"])
print("The Skew of the transactionAmount is:", skew)

The Skew of the transactionAmount is: 2.0967999524928373

#show kurtosis of transaction amount
kurtosis = scipy.stats.kurtosis(df["transactionAmount"])
print("The Kurtosis of the transactionAmount is:", kurtosis)

The Kurtosis of the transactionAmount is: 6.378700767945739
```

Gambar .11. Persentase *Fraud*

Berdasarkan gambar 11 diatas dihasilakan rasio dari data transaksi berlabel *fraud* adalah sejumlah 0,017 dan data transaksi yang berlabel non *fraud* sejumlah 0,96. Dihasilkan juga persentase data transaksi yang berlabel *fraud* adalah 1,72%, sedangkan persentase dari data transaksi yang berlabel non *fraud* sejumlah 98,28%. Skew dari variabel transactionAmount adalah 2,09, terakhir adalah kurtosis dari variabel transactionAmount sebanyak 6,37.



```
# Visualizing distribution of fraud and non fraud transactions
sns.set(rc={'figure.figsize':(12,9)})
df.isFraud.value_counts().plot(kind='pie',autopct='%.2f%%',explode=(0,0.1))

<matplotlib.axes._subplots.AxesSubplot at 0x7f37ab1ea550>
```


Gambar .12. Visualisasi Persentase *Fraud*

Dari gambar 12 diatas dapat disimpulkan bahwa data yang berlabel non *fraud* adalah 98,28%, sedangkan data yang berlabel *fraud* adalah sebanyak 1,72%.

Kemudian dilanjutkan dengan mencari nilai rata-rata dan juga mengecek variable transactionAmount yang hilang :

DOI: https://doi.org/10/25047/jtit.v9i2.297

```
#show mean transaction amount (non-fraud)
print("Mean Transaction Amount (Non-Fraud):", np.mean(df[df["isFraud"]==False]["transactionAmount"]))

Mean Transaction Amount (Non-Fraud): 133.48826134835

#show mean transaction amount (fraud)
print("Mean Transaction Amount (Fraud):", np.mean(df[df["isFraud"]==True]["transactionAmount"]))

Mean Transaction Amount (Fraud): 232.5557537642306
```

Gambar .13. Rata-rata Transaksi *Fraud* dan *Non Fraud*

Dari gambar 13 diatas dihasilkan rata-rata transaksi yang berlabel non *fraud* adalah sebanyak 133,48, sedangkan rata-rata transaksi yang berlabel *fraud* sebanyak 232,55.

```
# Check transactionAmount lost
transactionAmountlost=pd.DataFrame({'transactionAmount':df['transactionAmount'].describe(),'Non Fraud
transactionAmountlost
```

	transactionAmount	Non Fraud TransactionAmount	Fraud TransactionAmount
count	633846.000000	622954.000000	10892.000000
mean	135.190636	133.488261	232.555754
std	147.095318	145.652814	190.123501
min	0.000000	0.000000	0.000000
25%	32.320000	31.820000	91.260000
50%	85.820000	84.410000	185.225000
75%	189.050000	186.520000	324.517500
max	1825.250000	1825.250000	1743.510000

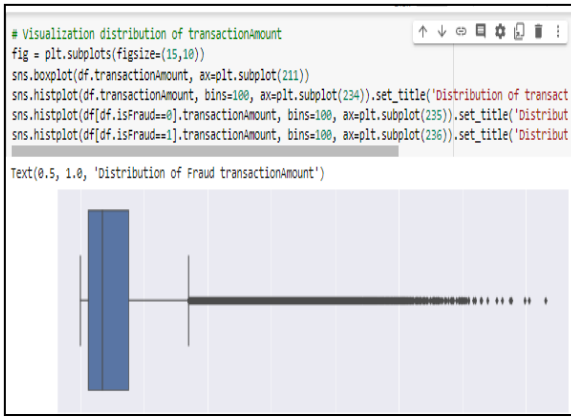
Gambar .14. transactionAmount yang Hilang

Dari gambar 13 dan 14 dapat diperoleh bahwa :

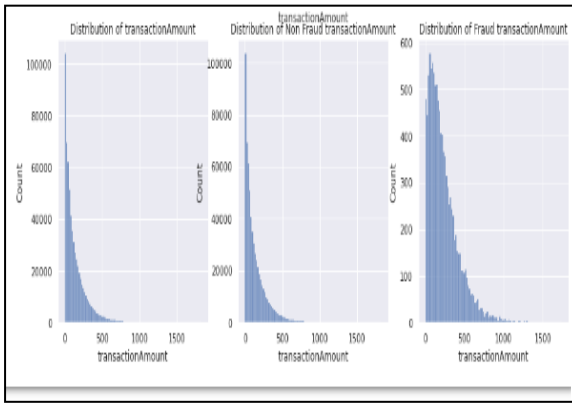
1. Rata-rata dan standar deviasi transactionAmount lebih kecil daripada jumlah *Fraud* TransactionAmount.
2. Rata-rata jumlah transaksi *Fraud* TransactionAmount sangat tinggi yang berarti jumlah kerugian dalam transaksi penipuan (*fraud*) sangat tinggi.

Berikut ini adalah visualisasi persebaran data pada variable transactionAmount :

```
# Visualization distribution of transactionAmount
fig = plt.subplots(figsize=(15,10))
sns.boxplot(df.transactionAmount, ax=plt.subplot(211))
sns.histplot(df.transactionAmount, bins=100, ax=plt.subplot(234)).set_title('Distribution of transact
sns.histplot(df[df.isFraud==0].transactionAmount, bins=100, ax=plt.subplot(235)).set_title('Distribut
sns.histplot(df[df.isFraud==1].transactionAmount, bins=100, ax=plt.subplot(236)).set_title('Distribut

Text(0.5, 1.0, 'Distribution of Fraud transactionAmount')
```



Gambar .15. Sintaks Visualisasi transactionAmount



Gambar .16. Boxplot transactionAmount

Dari gambar 15 dan gambar 16 diatas, pada boxplot transactionAmount tersebut memiliki banyak sekali *outlier*, ini menunjukkan bahwa variable transactionAmount memiliki pengaruh yang sangat besar dalam fitur menentukan *fraud*. Distribusi yang sangat miring dan memiliki kurtosis yang tinggi ini berarti bahwa jumlah transaksi tidak memiliki distribusi simetris dan Sebagian besar memiliki nilai terletak pada ujung yang jumlah lebih kecil.

Jadi, dari histogram yang terbentuk diketahui *Fraud* transactionAmount memiliki distribusi yang merata dan dimana transaksi *fraud* sering terjadi pada jumlah kecil berkisar 100 hingga 500. Sehingga dapat diketahui bahwa transaksi dengan jumlah kecil harus dipantau lebih jauh untuk mendeteksi dan mencegah terjadinya penipuan kartu kredit.

Selanjutnya dilakukan pendeteksian *fraud* berdasarkan dengan tipe transaksi yang banyak dilakukan:

```
# Fraud detection based on transaction type
fig = plt.subplots(figsize=(15,10))
sns.countplot(x='transactionType', hue='isFraud', data = df,ax=plt.subplot(221)).set_title('Distribus
plt.xticks(rotation=45)
sns.countplot(x=df[df.isFraud==1].transactionType, data = df,ax=plt.subplot(222)).set_title('Distribu
plt.xticks(rotation=45)
plt.show()
```

Gambar .17. Jenis Transaksi

Dari gambar 17 diatas, data bar plot tersebut menunjukkan bahwa :

1. Terdapat transaksi dengan tipe *purchase* yang mimiliki jumlah transaksi *fraud*/penipuan paling besar.
2. Terdapat transaksi dengan tipe *reversal* yang mengacu pada situasi kegagalan transaksi. Maka transaksi *reversal* harus diawasi.

DOI: https://doi.org/10.25047/jtit.v9i2.297

3. Terdapat transaksi dengan tipe address verification lebih kecil dibandingkan dengan tipe transaksi lainnya. Sehingga, transaksi tipe ini bisa menjadi solusi dari transaksi *fraud.*

4.3. *Split Data Training dan Data Testing.*

Setelah dilakukan *data preparation* dilakukan pengubahan nilai data terlebih dahulu menggunakan StandardScaler SKLearn dan juga menghapus kolom yang tidak akan digunakan dalam *project* ini.

```
scaler = StandardScaler()
df['Normalized_Amount'] = scaler.fit_transform(df.transactionAmount.values.reshape(-1,1)) # Normaliz
# Dropping columns
df.drop(['transactionAmount', 'transactionDateTime', 'customerId', 'merchantName', 'accountOpenDate'
df
```

	isFraud	Normalized_Amount
0	1	-0.162212
1	0	-0.750811
2	0	0.354936
3	1	0.626461
4	0	-0.919069
...	...	...
641909	0	-0.882562
641910	0	0.601715
641911	0	0.021954
641912	0	-0.808188
641913	0	-0.697920

633846 rows × 2 columns

Gambar .18. StandardScaler SKLearn

Selanjutnya data dibagi menjadi dua, yaitu data *training* dan juga data *testing.* Dimana algoritma dilatih menggunakan data *traning*, sedangkan data *testing* akan dipakai untuk algoritma yang sudah dilatih menggunakan data baru yang sebelumnyya ditemukan agar dapat melihat seberapa baik kinerjanya. 70% dari data dijadikan sebagai data *training* dan 30% untuk data *testing*.

```
Y = df.isFraud
X = df.drop(['isFraud'],axis = 1)
(train_x,test_x,train_y,test_y) = train_test_split(X,Y, test_size=0.3, random_state=42)

print("train_x size: ", train_x.shape)
print("test_x size: ", test_x.shape)

train_x size:  (443692, 1)
test_x size:  (190154, 1)
```

Gambar .19. *Split Data*

4.4. *Classification Model*

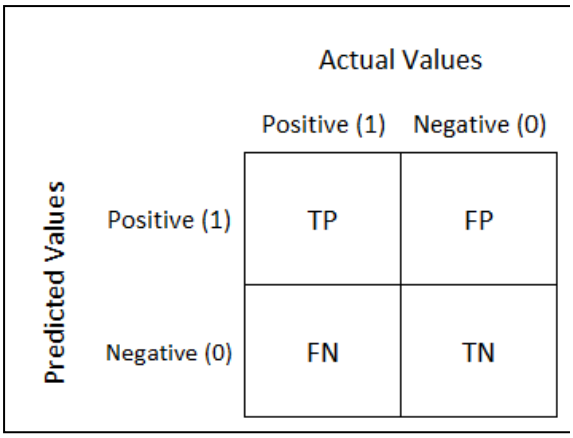
Pada tahap ini algoritma yang akan digunakan untuk mencari model terbaik adalah *Random Forest Classifiers* dan *Decision Tree Classifiers.*

```
# Preparing Classifiers
decision_tree = DecisionTreeClassifier()
random_forest = RandomForestClassifier(n_estimators=100)

# Decision Tree
decision_tree.fit(train_x,train_y)
predictions_dt = decision_tree.predict(test_x)
decision_tree_score = round(decision_tree.score(test_x,test_y) * 100, 2)

# Random Forest
random_forest.fit(train_x,train_y)
prediction_rf = random_forest.predict(test_x)
random_forest_score = round(random_forest.score(test_x,test_y) * 100,2)

print('Decision Tree Performance: ', decision_tree_score)
print('Random Forest Performance: ', random_forest_score)

Decision Tree Performance:  98.02
Random Forest Performance:  97.98
```

Gambar .20. Klasifikasi Model

Dari gambar 20 diatas diperoleh bahwa performa menggunakan *Decision Tree* sebesar 98.02, sedangkan performa menggunakan *Random Forest* sebesar 97.98.

Ketika berhadapan dengan model klasifikasi, ada beberapa metrik evaluasi yang dapat digunakan untuk melihat efesiensi model yang dibuat. Salah satunya adalah *confusion matrix* yang merupakan rangkuman dari hasil perkiraan yang dibandingkan dengan nilai sebenarnya dari kumpulan data. Bilai model dihasilkan menggunakan *table matrix* dengan menguji dalam *confusion matrix*. Kelas pertama yang terdapat dalam dua kelas dikenal sebagai kelas positif dan kelas kedua dikenal sebagai kelas negatif [16].



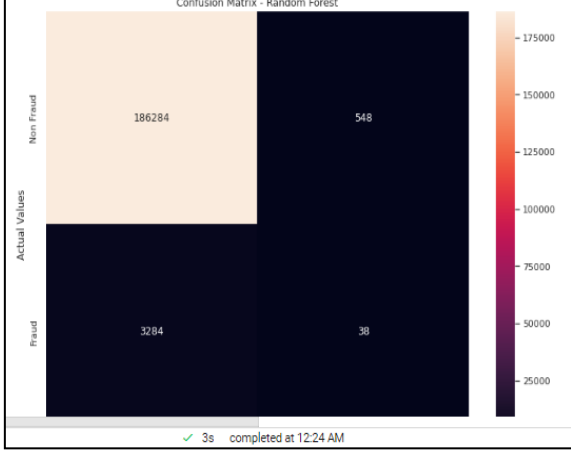
Gambar .21. *Confusion Matrix*

DOI: https://doi.org/10/25047/jtit.v9i2.297

Berikut ini adalah penerapan *confusion matrix* dengan *Decision Tree Classifier*.

```
# Confusion Matrix
confusion_matrix_random_forest = confusion_matrix(test_y, prediction_rf)
# Visualization
ax = plt.subplot()
sns.heatmap(confusion_matrix_random_forest, annot=True, fmt='g', ax = ax)
ax.set_xlabel('Predicted Values')
ax.set_ylabel('Actual Values')
ax.set_title('Confusion Matrix - Random Forest')
ax.xaxis.set_ticklabels(['Non Fraud','Fraud'])
ax.yaxis.set_ticklabels(['Non Fraud','Fraud'])
```

```
[Text(0, 0.5, 'Non Fraud'), Text(0, 1.5, 'Fraud')]
```




```
✓ 3s completed at 12:24 AM
```

Gambar .22. Visualisasi *Confusion Matrix*

Dari gambar 22 diatas dapat disimpulkan hasil sebagai berikut :

- 186,284 *True Positives*
- 548 *False Negatives*
- 3,284 *False Positives*
- 38 *True Positives*

```
# Printing performance metrics for Decision Tree classifier
metrics_decision_tree = [['Accuracy',(accuracy_score(test_y, predictions_dt))],
                         ['Precision',precision_score(test_y, predictions_dt)],
                         ['Recall', recall_score(test_y, predictions_dt)],
                         ['F1_score',f1_score(test_y, predictions_dt)]]
metric_values_decision_tree = pd.DataFrame(metrics_decision_tree, columns = ['Metrics', 'Result'])
metric_values_decision_tree
```

	Metrics	Result
0	Accuracy	0.980169
1	Precision	0.070746
2	Recall	0.011138
3	F1_score	0.019246

Gambar .23. *Confusion Matrix Decision Tree*

Dari gambar 23 diatas diperoleh bahwa tingkat akurasi dari *decision tree* menggunakan *confusion matrix* adalah 98%, *precissio* 7%, *recall* 1,1%, dan F1_*score* sebanyak 1,9%

4.5. *Synthetic Minority Oversampling Techniq*ue (SMOTE)

Pada tahap ini SMOTE akan menghasilkan poin data baru dari kelas minoritas menggunakan instans yang berbeda, sehingga menghasilkan sampel yang mirip dengan instans yang dimiliki tapi bukan salinan aslinya. *Random Oversampling* (ROS) merupakan SMOTE yang akan dipakai. Gambar dibawah ini merupakan sintaks yang digunakan untuk *oversampling* data menggunakan SMOTE.

```
# Importing SMOTE from imblearn lib
from imblearn.over_sampling import SMOTE
resampled_x, resampled_y = SMOTE().fit_resample(X,Y) # reshaping data

print('X New Shape: ', resampled_x.shape)
print('Y New Shape: ', resampled_y.shape)
```

```
X New Shape:  (1245908, 1)
Y New Shape:  (1245908,)
```

Gambar .24. *Oversampling* SMOTE

```
# Splitting our resampled data
(train_x,test_x,train_y,test_y) = train_test_split(resampled_x, resampled_y,
                                                   test_size = 0.3, random_state = 42)

# Applying Random Forest Classifier with new resampled data
resampled_random_forest = RandomForestClassifier(n_estimators = 100)
resampled_random_forest.fit(train_x,train_y)

predictions_resampled = resampled_random_forest.predict(test_x)
random_forest_new_score = round(resampled_random_forest.score(test_x,test_y) * 100, 4)
print('Performance: ', random_forest_new_score)
```

```
Performance:  90.6847
```

Gambar .25. *Split Resample Data*

Dari gambar 25 diatas, setelah data di *oversampling* menggunakan SMOTE, data dibagi kemudian digunakan pada *Random Forest Classifier* dengan performa mencapai 90,73%.

4.6. *Select the Best Accuracy Result*.

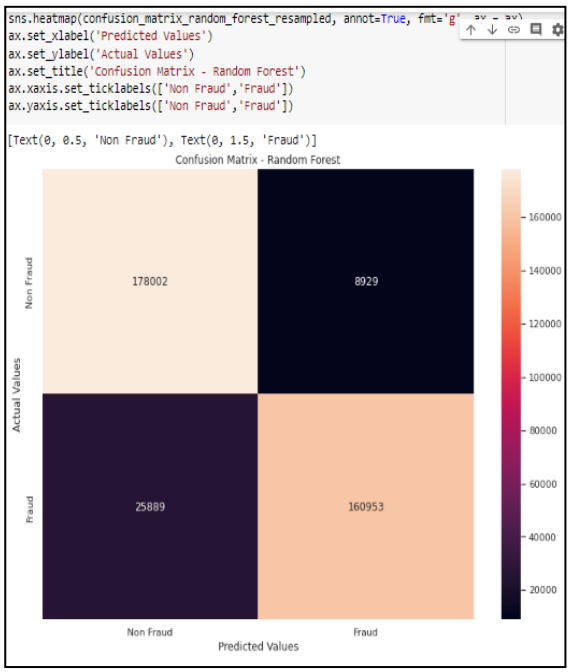
Dari percobaan yang dilakukan akan dianalisa dan dipilih model dengan akurasi terbaik. Berikut dibawah ini merupakan model dengan akurasi yang terbaik.

```
# Random Forest Evaluation metrics after oversampling
metrics_random_forest_resampled = [['Accuracy',(accuracy_score(test_y, predictions_resampled))],
                                   ['Precision',precision_score(test_y, predictions_resampled)],
                                   ['Recall', recall_score(test_y, predictions_resampled)],
                                   ['F1_score',f1_score(test_y, predictions_resampled)]]
metric_values_random_forest_resampled = pd.DataFrame(metrics_random_forest_resampled, columns = ['Met
metric_values_random_forest_resampled
```

	Metrics	Results
0	Accuracy	0.906847
1	Precision	0.947440
2	Recall	0.861439
3	F1_score	0.902395

Gambar .26. *Random Forest Evaluation*

DOI: https://doi.org/10.25047/jtit.v9i2.297



Gambar .27. *Confusion Matrix Random Forest*

Dari hasil menggunakan SMOTE untuk mengambil data secara *oversample* dapat dilihat bahwa model tersebut tidak lagi mendukung transaksi asli (*non fraud*) daripada transaksi *fraud* dan hasilnya adalah :

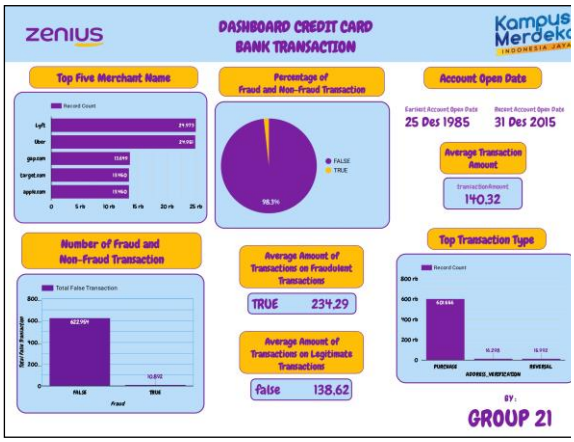
- 178.106 *True Positives*
- 8.825 *False Negatives*
- 25.754 *False Positives*
- 161.088 *True Negatives*

Selain itu, hasil lain yang dicapai :

- *Accuracy* 90,68%
- *Precision* 94,74%
- *Recall* 86,14%
- F1 *Score* 90,23%

Setelah mendapatkan semua data yang diperlukan yang dibuat menggunakan Google Colab dengan memakai bahasa pemrograman Python diatas. Data yang telah dipilih kemudian disimpan dalam bentuk CSV yang nantinya akan diupload ke dalam Google Data Studio untuk dirangkum berapa banyak data transaksi yang berlabel *fraud* dan juga data transaksi yang berlabel non *fraud*, lima *merchant name* yang paling banyak melakukan transaksi, persentase data transaksi yang berlabel *fraud* dan juga data transaksi yang berlabel non *fraud*, mencari rata-rata jumlah transaksi yang terdapat *fraud* dan juga mencari rata-rata jumlah transaksi yang dilakukan secara sah, mencari tipe transaksi yang paling banyak digunakan, mencari kapan tanggal pembuatan akun yang pertama kali dibuat, terakhir mencari tanggal kapan akun yang paling terbaru dibuat.

Berikut di bawah ini adalah hasil pengerjaan *project* yang dirangkum dan dibuat menjadi *dashboard* menggunakan Google Data Studio.



Gambar .28. *Dashboard*

Dari gambar 28 *dashboard* di atas dapat disimpulkan hasil sebagai berikut :

1. Banyak transaksi yang terdapat *fraud* adalah 10.892 dan banyak transaksi non *fraud* adalah 622.954.
2. Persentase banyaknya transaksi yang berlabel *fraud* adalah sebanyak 1,7%, sedangkan banyak transaksi yang berlabel non *fraud* adalah 98,3%.
3. Rata-rata jumlah transaksi yang terdapat *fraud* sejumlah 234,29.
4. Rata-rata transaksi sah adalah 138,62.
5. Tipe transaksi yang paling banyak digunakan adalah *purchase* sebanyak 601,556, *address_verification* 16,298, dan reversal 15,992.
6. Lima nama *merchant* yang paling banyak adalah Lyft sebanyak 24,973, Uber sebanyak 24,951, Gap.com sebanyak 13,649, Target.com sebanyak 13,450, terakhir apple.com sebanyak 13,450.
7. Rata-rata variabel transactionAmount adalah 140,32.
8. Tanggal akun yang baru dibuka adalah 31 Desember 2015.
9. Tanggal akun yang pertama kali dibuka adalah tanggal 15 Desember 1985.

KESIMPULAN

Berdasarkan pembahasan dalam penelitian menggunakan algoritma *random forest*, data yang digunakan memiliki 641.914 records dan 29 fields. Data ini bersifiat tidak seimbang, sehingga dilakukan penyeimbangkan menggunakan SMOTE.

Hasil dari penelitian menunjukkan bahwa algoritma *random forest* menghasilkan akurasi sebesar 90,68%, presisi sebesar 94,74%, recall sebesar 86,14%, dan F1 score sebesar 90,23% karena hasil klasifikasi secara keseluruhan berada diantara 90-100% maka nilai ini dikategorikan sangat baik. Berdasarkan dari hasil penelitian dapat diketahui bahwa hasil akurasi menggunakan algoritma *random forest* dan *oversampling* SMOTE lebih tinggi dibandingkan hasil akurasi

DOI: https://doi.org/10/25047/jtit.v9i2.297

penelitian sebelumnya yang menggunakan algoritma *random forest* dan optimasi *bagging* dan GA yaitu sebesar 88,30%. Dengan ini dapat disimpulkan bahwa model dan algoritma *random forest* yang dikembangkan dapat mendeteksi *fraud* yang terjadi pada transaksi kartu kredit sehingga mencegah terjadinya kerugian baik bagi konsumen maupun perbankan.

PENGHARGAAN

Ucapan terima kasih saya sampaikan kepada Ibu Eka Puji Agustini yang telah membimbing saya selama pembuatan proposal dan juga laporan.

REFERENSI

[1] D. J. Ardha, "Analisis Kasus Pemalsuan Kartu Kredit Sebagai Bentuk Tindak Pidana Perbankan," *J. Huk. Doctrin.*, vol. 5, no. 2, pp. 243–263, 2020.

[2] A. Wulandari, M. E. Putri, and Y. Marlina, "Pengaruh Audit Investigasi Terhadap Pengungkapan Fraud di Indonesia," *J. Akunt. Ummi*, vol. 1, no. 2, pp. 245–263, Feb. 2020, doi: 10.37150/jammi.v1i2.1147.

[3] N. Christian and J. Veronica, "DAMPAK KECURANGAN PADA BIDANG KEUANGAN DAN NON KEUANGAN TERHADAP JENIS FRAUD DI INDONESIA," *J. Ris. Akunt. Mercu Buana*, vol. 8, no. 1, pp. 91–102, May 2022, doi: 10.260486/jramb.v8i1.2401.

[4] A. Ariesta, "Kasus Fraud Naik Signifikan Selama Pandemi, Tertingi di Segmen E-commerce," *iNews.id*, p. 1, Sep. 24, 2021.

[5] E. I. Supriyadi and D. B. Asih, "IMPLEMENTASI ARTIFICIAL INTELLIGENCE (AI) DI BIDANG ADMINISTRASI PUBLIK PADA ERA REVOLUSI INDUSTRI 4.0," *J. RASI*, vol. 2, no. 2, pp. 12–22, Jan. 2021, doi: 10.52496/rasi.v2i2.62.

[6] A. Ahmad, "Mengenal Artificial Intelligence, Machine Learning, Neural Network, dan Deep Learning," *J. Teknol. Indones.*, pp. 1–5, Jun. 2017.

[7] Yoga Religia, Agung Nugroho, and Wahyu Hadikristanto, "Klasifikasi Analisis Perbandingan Algoritma Optimasi pada Random Forest untuk Klasifikasi Data Bank Marketing," *J. RESTI Rekayasa Sist. Dan Teknol. Inf.*, vol. 5, no. 1, pp. 187–192, Feb. 2021, doi: 10.29207/resti.v5i1.2813.

[8] G. A. Mursianto, I. M. Falih, M. Irfan, and D. S. Prasvita, "Perbandingan Metode Klasifikasi Random Forest dan XGBoost Serta Implementasi Teknik SMOTE pada Kasus Prediksi Hujan," *J. SENAMIKA*, vol. 2, no. 2, pp. 41–50, Sep. 2021.

[9] A. S. More and D. P. Rana, "Review of random forest classification techniques to resolve data imbalance," in *2017 1st International Conference on Intelligent Systems and Information Management (ICISIM)*, Aurangabad, Oct. 2017, pp. 72–78. doi: 10.1109/ICISIM.2017.8122151.

[10] A. Parmar, R. Katariya, and V. Patel, "A Review on Random Forest: An Ensemble Classifier," in *International Conference on Intelligent Data Communication Technologies and Internet of Things*, Aug. 2018, pp. 758–763. doi: 10.1007/978-3-030-03146-6-86.

[11] D. Irawan, E. B. Perkasa, Y. Yurindra, D. Wahyuningsih, and E. Helmud, "Perbandingan Klassifikasi SMS Berbasis Support Vector Machine, Naive Bayes Classifier, Random Forest dan Bagging Classifier," *J. Sisfokom Sist. Inf. Dan Komput.*, vol. 10, no. 3, pp. 432–437, Dec. 2021, doi: 10.32736/sisfokom.v10i3.1302.

[12] L. Breiman, "Random Forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.

[13] M. Y. Sahroni, N. A. Setifani, Universitas Islam Negeri Sunan Ampel, D. N. Fitriana, and Universitas Islam Negeri Sunan Ampel, "Analisis perbandingan algoritma Naïve Bayes, k-Nearest Neighbor dan Neural Network untuk permasalahan class-imbalanced data pada kasus credit card fraud dataset," *Teknologi*, vol. 11, no. 2, pp. 69–73, Jul. 2021, doi: 10.26594/teknologi.v11i2.2393.

[14] H. Sunata, "Komparasi Tujuh Algoritma Identifikasi Fraud ATM Pada PT. Bank Central Asia Tbk," *JATISI J. Tek. Inform. Dan Sist. Inf.*, vol. 7, no. 3, pp. 441–450, Dec. 2020, doi: 10.35957/jatisi.v7i3.471.

[15] Y. Mardi, "Data Mining : Klasifikasi Menggunakan Algoritma C4.5," *Edik Inform.*, vol. 2, no. 2, pp. 213–219, Feb. 2017, doi: 10.22202/ei.2016.v2i2.1465.

[16] I. K. Hasan, R. Resmawan, and J. Ibrahim, "Perbandingan K-Nearest Neighbor dan Random Forest dengan Seleksi Fitur Information Gain untuk Klasifikasi Lama Studi Mahasiswa," *Indones. J. Appl. Stat.*, vol. 5, no. 1, p. 58, May 2022, doi: 10.13057/ijas.v5i1.58056.