

LAPORAN TUGAS BESAR KECERDASAN BUATAN
KEPUTUSAN KREDIT THERA BANK DENGAN METODE
ARTIFICIAL NEURAL NETWORK

**Bane Rael Sharin¹⁾, Dimas Wahyu Saputro²⁾, Fadillah Muhesa Latri³⁾,
Lisnurani⁴⁾, Marhanny Zahra N⁵⁾.**

Program Studi Sains Data, Jurusan Sains, Institut Teknologi Sumatera
bane.120450101@student.itera.ac.id¹⁾, dimas.120450081@student.itera.ac.id²⁾,
fadillah.120450003@student.itera.ac.id³⁾, lisnurani.120450055@student.itera.ac.id⁴⁾,
marhanny.120450017@student.itera.ac.id⁵⁾

Abstrak

Dalam dunia keuangan, kemampuan bank untuk mengidentifikasi pelanggan potensial yang memiliki kemungkinan tinggi untuk membeli pinjaman sangat penting untuk meningkatkan rasio sukses kampanye pinjaman dengan mengurangi biaya yang dikeluarkan. Sebelumnya, kampanye yang dijalankan oleh bank untuk nasabah liabilitas menunjukkan tingkat konversi yang cukup baik, yaitu lebih dari 9% sukses. Namun, untuk meningkatkan tingkat konversi tersebut dan mengurangi biaya kampanye, bank ingin membangun model yang dapat membantu mengidentifikasi pelanggan potensial yang memiliki probabilitas lebih tinggi untuk membeli pinjaman.

Model yang akan dibangun adalah menggunakan teknik pembelajaran mesin, khususnya jaringan syaraf tiruan (ANN). ANN merupakan salah satu metode yang dapat digunakan untuk menyelesaikan masalah klasifikasi, seperti dalam kasus ini. Dengan menggunakan ANN, model dapat mempelajari pola dari data pelanggan yang ada dan memprediksi pelanggan potensial yang memiliki kemungkinan tinggi untuk membeli pinjaman. Dengan demikian, model yang dibangun menggunakan ANN diharapkan dapat membantu bank dalam meningkatkan rasio sukses kampanye pinjaman sambil mengurangi biaya yang dikeluarkan.

Kata Kunci : *Artificial Neural Network (ANN), Bunga Pinjaman, Kampanye Pinjaman, Target, Thera Bank.*

1. Pendahuluan

Kredit merupakan salah satu fasilitas dari bank untuk membantu seseorang atau badan usaha membeli produk dan mengembalikan dana yang dipakai dengan tambahan bunga[1]. Bank berhak membuat keputusan tentang pengajuan kredit apakah diterima atau ditolak dari histori rekening nasabah. kredit salah satu sumber utama pendapatan bagi sektor perbankan serta menjadi sumber terbesar resiko keuangan dari bank. Sebagian besar dari aset bank adalah hasil bunga dari kredit. Resiko yang dapat terjadi adalah nasabah yang mengajukan kredit tidak mampu membayar kembali pinjaman yang sudah diberi kasus tersebut disebut '*Credit Risk*'[2]. Untuk itu sangat penting bank melihat riwayat nasabah untuk menyesuaikan kredit sebelum mengesahkan pinjaman. Ada otoritas yang digunakan oleh bank dengan menggunakan lima(5) prinsip, yaitu : *Character* (karakter), *Capital* (modal), *Capacity* (kapasitas), *Collateral* (aset), dan *Conditions* (ketentuan) untuk mengevaluasi pinjaman [3].

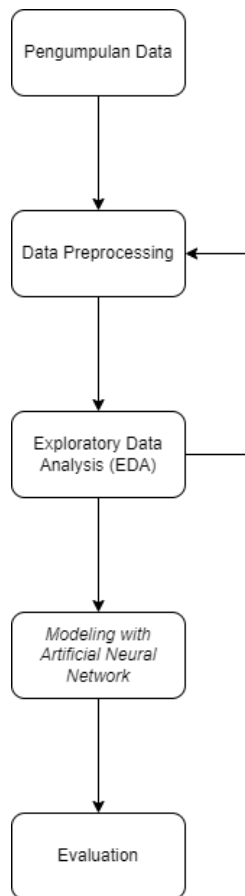
Evaluasi dilakukan berdasarkan pengalaman dan riwayat transaksi nasabah. Bank dapat menyetujui pinjaman dari nasabah namun tanpa kepastian apakah nasabah akan mengembalikan dana yang dipinjam atau tidak. Pada era lama bank mempekerjakan orang yang profesional hanya untuk mengevaluasi nasabah untuk memberi keputusan seorang nasabah dapat menerima sebuah pinjaman. kelayakan seorang nasabah untuk menerima pinjaman atau tidak disebut dengan *credit score*. Secara umum '*Credit Score*' dapat digunakan sebagai probabilitas nasabah mengembalikan dana pinjaman berdasarkan *credit score* atau riwayat penggunaan kartu [4].

Diperlukan algoritma statistik untuk menilai nasabah apakah layak atau tidak mendapat pinjaman dana. Ada metode baru yang digunakan bank dalam pengklasifikasian yaitu dengan menggunakan *machine learning* dan pembelajaran algoritma agar otomatis menilai kelayakan nasabah untuk mendapat izin peminjaman dana.

Oleh karena itu, untuk mengatasi nasabah yang tidak mengembalikan dana pinjaman maka diperlukan penelitian dengan menganalisis data bank dengan metode *Artificial Neural Network (ANN)* yang digunakan untuk menyelesaikan masalah memprediksi kelayakan nasabah untuk mendapat izin peminjaman dana berdasarkan *history* nasabah.

2. Metode

Metode yang dilakukan dapat dilihat pada gambar *flowchart* dibawah ini



Gambar 1. *Flowchart* Metode

2.1 Pengumpulan Data

Data yang digunakan berasal dari *source* terbuka (kaggle) dengan judul *Bank Loan Model using Decision Tree Classifier*. Terdapat 5000 baris dan 14 kolom diantaranya yaitu *ID*, *Age*, *Experience*, *Income*, *ZIP Code*, *Family*, *CCAvg*, *Education*, *Mortgage*, *Personal Loan*, *Securities Account*, *CD Account*, *Online*, *Credit Card*. Dapat dilihat pada Tabel 1 dibawah ini.

Tabel 1. Atribut pada Dataset beserta keterangannya

Atribut	Keterangan
ID	ID Pelanggan
Age	Umur Pelanggan
Experience	Jumlah pengalaman kerja dalam beberapa tahun
Income	Jumlah pendapatan tahunan (dalam ribuan)
ZIP Code	Kode pos tempat tinggal pelanggan

Family	Jumlah anggota keluarga
CCAvg	Rata-rata pengeluaran kartu kredit bulanan
Education	Tingkat pendidikan (1: Sarjana, 2: Master, 3: Gelar Lanjutan)
Mortgage	Hipotek rumah (dalam ribuan)
Personal Loan	Ini adalah variabel target (Masalah Klasifikasi Biner)
Securities Account	Apakah nasabah memiliki efek rekening
CD Account	Boolean apakah nasabah memiliki rekening Sertifikat Deposito
Online	Boolean apakah pelanggan menggunakan perbankan online
Credit Card	Apakah nasabah menggunakan kartu kredit yang dikeluarkan oleh bank?

	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	CreditCard
	0	1	25	1	49	91107	4	1.6	1	0	1	0	0	0
	1	2	45	19	34	90089	3	1.5	1	0	1	0	0	0
	2	3	39	15	11	94720	1	1.0	1	0	0	0	0	0
	3	4	35	9	100	94112	1	2.7	2	0	0	0	0	0
	4	5	35	8	45	91330	4	1.0	2	0	0	0	0	1

	4995	4996	29	3	40	92697	1	1.9	3	0	0	0	1	0
	4996	4997	30	4	15	92037	4	0.4	1	85	0	0	1	0
	4997	4998	63	39	24	93023	2	0.3	3	0	0	0	0	0
	4998	4999	65	40	49	90034	3	0.5	2	0	0	0	1	0
	4999	5000	28	4	83	92612	3	0.8	1	0	0	0	1	1

5000 rows × 14 columns

Gambar 2. Data awal sebelum diolah

2.2 Data Preprocessing

Data *preparation* merupakan tahapan sebelum proses pengklasifikasian data yang diperlukan untuk membersihkan, menghilangkan, dan atau mengubah bentuk data. Beberapa data pada tabel yang akan diuji harus ditransformasikan atau diinisiasi menjadi nilai tertentu. Data *preprocessing* bertujuan untuk meningkatkan kualitas pada data sebelum dilakukan pemodelan agar didapatkan model yang memiliki performa yang baik. Beberapa langkah data *preprocessing* yang dilakukan pada laporan ini yaitu membersihkan dan memperbaiki data yang rusak, menghapus data yang tidak penting, serta menggabungkan beberapa data yang sejenis menjadi satu atribut.

2.3 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) atau diartikan dalam Bahasa Indonesia yaitu analisis data eksplorasi adalah suatu langkah penting dalam setiap analisis suatu penelitian. Tujuan utama dari langkah ini adalah guna memeriksa data untuk distribusi, *outlier*, dan lainnya untuk mengarahkan pengujian spesifik dari hipotesis yang dimiliki. Langkah ini juga berguna untuk menjadi alat pembuatan hipotesis melalui visualisasi data sehingga membantu memahami data melalui representasi grafis yang ditampilkan. *Exploratory Data Analysis* (EDA) juga berguna untuk pengenalan pola alami dari analisis. Biasanya analisis data eksplorasi dilakukan dengan 3 cara, yaitu analisis univariat guna analisis deskriptif yang memiliki satu variabel, analisis bivariat untuk analisis dengan dua variabel atau target variabel, dan analisis multivariat guna analisis dengan lebih dari satu variabel. Dari ketiga cara tersebut, tiap cara dapat dilakukan dengan cara grafis maupun secara non grafis [5]. Pada penelitian ini dilakukannya analisis data eksplorasi, sebagai berikut :

```
bank_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype  
---  -
 0   ID                    5000 non-null   int64  
 1   Age                   5000 non-null   int64  
 2   Experience             5000 non-null   int64  
 3   Income                 5000 non-null   int64  
 4   ZIP Code              5000 non-null   int64  
 5   Family                5000 non-null   int64  
 6   CCAvg                 5000 non-null   float64 
 7   Education             5000 non-null   int64  
 8   Mortgage              5000 non-null   int64  
 9   Personal Loan         5000 non-null   int64  
10  Securities Account    5000 non-null   int64  
11  CD Account            5000 non-null   int64  
12  Online                5000 non-null   int64  
13  CreditCard            5000 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 547.0 KB
```

Gambar 3. Informasi detail mengenai dataframe

```
bank_df.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
ID	5000.0	2500.500000	1443.520003	1.0	1250.75	2500.5	3750.25	5000.0
Age	5000.0	45.338400	11.463166	23.0	35.00	45.0	55.00	67.0
Experience	5000.0	20.104600	11.467954	-3.0	10.00	20.0	30.00	43.0
Income	5000.0	73.774200	46.033729	8.0	39.00	64.0	98.00	224.0
ZIP Code	5000.0	93152.503000	2121.852197	9307.0	91911.00	93437.0	94608.00	96651.0
Family	5000.0	2.396400	1.147663	1.0	1.00	2.0	3.00	4.0
CCAvg	5000.0	1.937938	1.747659	0.0	0.70	1.5	2.50	10.0
Education	5000.0	1.881000	0.839869	1.0	1.00	2.0	3.00	3.0
Mortgage	5000.0	56.498800	101.713802	0.0	0.00	0.0	101.00	635.0
Personal Loan	5000.0	0.096000	0.294621	0.0	0.00	0.0	0.00	1.0
Securities Account	5000.0	0.104400	0.305809	0.0	0.00	0.0	0.00	1.0
CD Account	5000.0	0.060400	0.238250	0.0	0.00	0.0	0.00	1.0
Online	5000.0	0.596800	0.490589	0.0	0.00	1.0	1.00	1.0
CreditCard	5000.0	0.294000	0.455637	0.0	0.00	0.0	1.00	1.0

Gambar 4. Ringkasan deskriptif mengenai dataframe yang telah di transpose

```
bank_df.isnull().sum()

ID          0
Age         0
Experience  0
Income      0
ZIP Code    0
Family      0
CCAvg       0
Education   0
Mortgage    0
Personal Loan 0
Securities Account 0
CD Account  0
Online      0
CreditCard  0
dtype: int64
```

Gambar 5. Mendeteksi nilai yang tidak diketahui dalam dataframe

```
avg_age = bank_df["Age"].mean()
print ("The average age of this dataset is {:.1f}.".format(avg_age))

The average age of this dataset is 45.3.
```

Gambar 6. Rata-rata umur *customer bank*

```
percent_cc = sum(bank_df["CreditCard"] == 1)/len(bank_df)
print ("The percentage of customers that own the bank's credit card is {:.2%}.".format(percent_cc))

The percentage of customers that own the bank's credit card is 29.40%.
```

Gambar 7. Persentase *customer* yang memiliki kartu kredit

```
percent_loan = sum(bank_df["Personal Loan"] == 1)/len(bank_df)
print ("The percentage of customers that took out a personal loan is {:.2%}.".format(percent_loan))
```

The percentage of customers that took out a personal loan is 9.60%.

Gambar 8. Persentase *customer* yang mengambil pinjaman pribadi

2.4 Modeling with Artificial Neural Network

Pemodelan klasifikasi untuk keputusan kredit bank yang dilakukan pada penelitian ini menggunakan algoritma *Artificial Neural Network* (ANN). ANN merupakan salah satu pemodelan kompleks yang dapat memprediksi bagaimana ekosistem merespon perubahan variabel lingkungan dengan terinspirasi oleh cara kerja sistem saraf biologis, khususnya pada sel otak manusia dalam memproses informasi. Dikarenakan model ANN terinspirasi oleh sistem saraf biologis manusia, arsitekturnya pun dibuat seperti struktur otak manusia dimana terdiri dari neuron yang saling terhubung satu sama lain dan bentuk yang kompleks dan non-linear. Jumlah data yang digunakan sebanyak 5000 baris dan 14 kolom.

Adapun pemodelan jaringan pada ANN yang digunakan yaitu *Multi-Layer Neural Network*. *Multi-Layer Neural Network* adalah neural network yang memiliki karakteristik multi layer dimana setiap node pada suatu layer terhubung dengan setiap node pada layer di depannya.

2.5 Matriks Evaluasi

Pada penelitian ini menggunakan *confusion matrix* yaitu pengukuran masalah klasifikasi *machine learning* dengan dua (2) keluaran atau lebih, pada *confusion matrix* memiliki empat (4) tabel, setiap tabel berisi *True Positive*, *True Negative*, *False Positif*, dan *False Negatif*.

Confusion Matrix		
	Prediksi	
Aktual	TRUE	FALSE
TRUE	TP	FP
FALSE	FN	TN

Gambar 9. Tabel *confusion matrix*

Selain *confusion matrix* digunakan juga *accuracy matrix* yaitu menghitung rasio prediksi benar dibagi dengan prediksi salah.

Accuracy	Predictions/ Classifications	Correct
		Correct + Incorrect

Gambar 10. *matrix accuracy*

3. Hasil dan Pembahasan

Artificial neural networks (ANNs) adalah sebuah model komputasi yang menirukan struktur dan fungsi jaringan saraf pada otak manusia. Dalam proyek ini, kami akan membangun dan melatih sebuah model jaringan saraf mendalam untuk memprediksi kemungkinan seorang pelanggan liabilitas membeli pinjaman pribadi berdasarkan fitur pelanggan. Kami akan menggunakan teknik pembelajaran mesin dan jaringan saraf untuk membangun model tersebut, kemudian melatihnya dengan data yang kami miliki. Setelah melakukan pelatihan, kami akan mengevaluasi kemampuan model tersebut dalam memprediksi kemungkinan seorang pelanggan membeli pinjaman pribadi.

Building a multi-layer neural network model

```
1 # sequential model
2 ann_model = keras.Sequential()
3
4 # adding dense layer
5 ann_model.add(Dense(250, input_dim=13, kernel_initializer='normal', activation='relu'))
6 ann_model.add(Dropout(0.3))
7 ann_model.add(Dense(500, activation='relu'))
8 ann_model.add(Dropout(0.3))
9 ann_model.add(Dense(500, activation='relu'))
10 ann_model.add(Dropout(0.3))
11 ann_model.add(Dense(500, activation='relu'))
12 ann_model.add(Dropout(0.4))
13 ann_model.add(Dense(250, activation='linear'))
14 ann_model.add(Dropout(0.4))
15
16 # adding dense layer with softmax activation/output layer
17 ann_model.add(Dense(2, activation='softmax'))
18 ann_model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 250)	3500
dropout (Dropout)	(None, 250)	0
dense_1 (Dense)	(None, 500)	125500
dropout_1 (Dropout)	(None, 500)	0

Gambar 11. Model ANN

Pada Gambar 11 di atas, terdapat implementasi model ANN menggunakan library tensorflow. Model ANN tersebut dibangun menggunakan sequential model, yaitu sebuah model ANN yang terdiri dari lapisan-lapisan yang terhubung secara berurutan. Selanjutnya, kode tersebut menambahkan beberapa lapisan ke dalam model ANN. Lapisan pertama terdiri dari lapisan dense yang menerima input dengan dimensi 13 dan terdapat sebanyak 250 neuron. Lapisan ini menggunakan inisialisasi kernel normal dan fungsi aktivasi relu. Setelah lapisan dense tersebut, terdapat lapisan dropout dengan tingkat dropout sebesar 0.3 yang bertujuan untuk menghindari *overfitting* pada model ANN.

Setelah lapisan pertama, terdapat tiga lapisan dense lain yang masing-masing memiliki jumlah neuron sebanyak 500 dan menggunakan fungsi aktivasi relu. Setiap lapisan dense tersebut diikuti oleh lapisan dropout dengan tingkat dropout yang berbeda-beda.

Kemudian, terdapat lapisan dense lain yang memiliki jumlah neuron sebanyak 250 dan menggunakan fungsi aktivasi linear. Lapisan ini juga diikuti oleh lapisan dropout dengan tingkat *dropout* sebesar 0.4.

Terakhir, terdapat lapisan output yang terdiri dari 2 neuron dan menggunakan fungsi aktivasi *softmax*. Fungsi *softmax* akan mengubah nilai *output* dari neuron menjadi nilai probabilitas, yang mana jumlah dari semua probabilitas tersebut akan bernilai 1. Selain itu, kode tersebut juga menampilkan ringkasan dari model ANN yang telah dibuat menggunakan fungsi *summary()*. Ringkasan tersebut akan menampilkan jumlah parameter yang digunakan oleh masing-masing lapisan, sehingga dapat memberikan informasi tentang ukuran model ANN tersebut.

```
[ ] 1 ann_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=[f1_m]) # metrics=['accuracy']
```

Gambar 12. Compile Model

Pada kode Gambar 12, terdapat fungsi *compile* yang digunakan untuk mengkompilasi model ANN yang telah dibuat sebelumnya. Fungsi *compile* akan menentukan *loss function*, *optimizer*, dan *metric* yang akan digunakan oleh model ANN tersebut saat melakukan pelatihan.

Pada kode di atas, *loss function* yang digunakan adalah *categorical_crossentropy*, yang mana merupakan *loss function* yang sesuai untuk klasifikasi biner atau multi kelas. Optimizer yang digunakan adalah *adam*, yang mana merupakan salah satu optimizer populer yang digunakan dalam ANNs. Metric yang digunakan adalah *f1_m*, yang mana merupakan salah satu metric yang digunakan untuk mengukur akurasi model ANN.

Setelah melakukan kompilasi, model ANN tersebut siap dilatih dengan data yang tersedia. Selama proses pelatihan, model ANN akan menggunakan *loss function*, *optimizer*, dan *metric* yang telah ditentukan untuk mengoptimalkan performa model ANN tersebut. Hasil dari pelatihan tersebut akan dapat digunakan untuk memprediksi kemungkinan seorang pelanggan membeli pinjaman pribadi.

Selanjutnya, akan dilakukan *fit data*, seperti pada Gambar 13. Terdapat fungsi *fit* yang digunakan untuk melakukan pelatihan model ANN yang telah dibuat dan dikompilasi sebelumnya. Fungsi *fit* akan menerima input data latih (*X_train*), target data latih (*y_train*), jumlah epoch (jumlah iterasi pelatihan), persentase data validasi, dan *verbose level* sebagai parameter.

```
1 history = ann_model.fit(X_train, y_train, epochs=20, validation_split=0.2, verbose=1)
```

Epoch 1/20

Gambar 13. Fit Model

Pada contoh output di atas, proses pelatihan model ANN tersebut dilakukan sebanyak 20 epoch. Setiap epoch, model ANN akan melakukan *forward propagation* dan *backpropagation* dengan menggunakan data latih yang telah diberikan. Selama proses pelatihan, model ANN akan mengevaluasi hasil pelatihannya dengan menggunakan data

validasi yang telah ditentukan. Setelah setiap epoch, hasil dari pelatihan dan evaluasi akan ditampilkan sesuai dengan *verbose level* yang telah ditentukan.

```
Epoch 1/20
113/113 [=====] - 4s 17ms/step - loss: 0.1811 - f1_m: 0.9336 - val_loss: 0.0867 - val_f1_m: 0.9655
Epoch 2/20
113/113 [=====] - 2s 15ms/step - loss: 0.1102 - f1_m: 0.9560 - val_loss: 0.0830 - val_f1_m: 0.9644
Epoch 3/20
113/113 [=====] - 2s 15ms/step - loss: 0.0850 - f1_m: 0.9699 - val_loss: 0.0501 - val_f1_m: 0.9784
Epoch 4/20
113/113 [=====] - 2s 14ms/step - loss: 0.0762 - f1_m: 0.9751 - val_loss: 0.0604 - val_f1_m: 0.9806
Epoch 5/20
113/113 [=====] - 2s 15ms/step - loss: 0.0705 - f1_m: 0.9773 - val_loss: 0.0509 - val_f1_m: 0.9849
Epoch 6/20
113/113 [=====] - 2s 15ms/step - loss: 0.0672 - f1_m: 0.9770 - val_loss: 0.0526 - val_f1_m: 0.9838
Epoch 7/20
113/113 [=====] - 2s 15ms/step - loss: 0.0620 - f1_m: 0.9790 - val_loss: 0.0489 - val_f1_m: 0.9817
Epoch 8/20
113/113 [=====] - 2s 15ms/step - loss: 0.0541 - f1_m: 0.9820 - val_loss: 0.0502 - val_f1_m: 0.9806
Epoch 9/20
113/113 [=====] - 2s 17ms/step - loss: 0.0565 - f1_m: 0.9829 - val_loss: 0.0740 - val_f1_m: 0.9838
Epoch 10/20
113/113 [=====] - 2s 15ms/step - loss: 0.0500 - f1_m: 0.9817 - val_loss: 0.0560 - val_f1_m: 0.9795
Epoch 11/20
113/113 [=====] - 2s 15ms/step - loss: 0.0414 - f1_m: 0.9884 - val_loss: 0.0525 - val_f1_m: 0.9828
Epoch 12/20
113/113 [=====] - 2s 17ms/step - loss: 0.0505 - f1_m: 0.9837 - val_loss: 0.0393 - val_f1_m: 0.9871
Epoch 13/20
113/113 [=====] - 2s 16ms/step - loss: 0.0499 - f1_m: 0.9829 - val_loss: 0.0440 - val_f1_m: 0.9828
Epoch 14/20
113/113 [=====] - 2s 19ms/step - loss: 0.0484 - f1_m: 0.9848 - val_loss: 0.0477 - val_f1_m: 0.9849
Epoch 15/20
113/113 [=====] - 2s 18ms/step - loss: 0.0428 - f1_m: 0.9859 - val_loss: 0.0563 - val_f1_m: 0.9774
Epoch 16/20
113/113 [=====] - 2s 18ms/step - loss: 0.0407 - f1_m: 0.9859 - val_loss: 0.0816 - val_f1_m: 0.9828
Epoch 17/20
113/113 [=====] - 2s 19ms/step - loss: 0.0466 - f1_m: 0.9848 - val_loss: 0.0505 - val_f1_m: 0.9817
Epoch 18/20
113/113 [=====] - 2s 18ms/step - loss: 0.0380 - f1_m: 0.9862 - val_loss: 0.0645 - val_f1_m: 0.9795
Epoch 19/20
113/113 [=====] - 2s 18ms/step - loss: 0.0433 - f1_m: 0.9845 - val_loss: 0.0418 - val_f1_m: 0.9838
Epoch 20/20
113/113 [=====] - 2s 18ms/step - loss: 0.0358 - f1_m: 0.9873 - val_loss: 0.0513 - val_f1_m: 0.9817
```

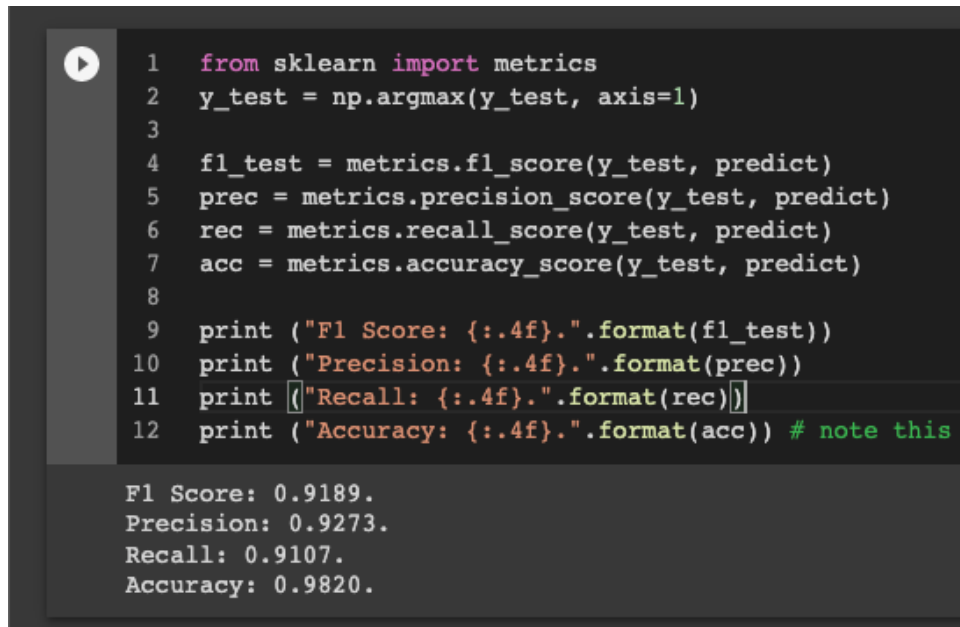
Gambar 14. Output Fit Model

Pada contoh output di atas, terlihat bahwa nilai loss dan f1_m berkurang seiring dengan bertambahnya epoch. Hal ini menunjukkan bahwa model ANN tersebut mampu belajar dari data latih dan meningkatkan performanya selama proses pelatihan. Selain itu, terlihat juga bahwa nilai loss dan f1_m pada data validasi cenderung lebih tinggi dibandingkan nilai loss dan f1_m pada data latih. Hal ini dapat terjadi karena data validasi digunakan untuk mengevaluasi performa model ANN tanpa menggunakan data yang digunakan untuk pelatihan. Nilai loss dan f1_m yang lebih tinggi pada data validasi menunjukkan bahwa model ANN tersebut mungkin mengalami *overfitting* pada data latih. Untuk memudahkan, akan dilakukan visualisasi, seperti pada Gambar 15.



Gambar 15. Visualisasi Model Loss

Selanjutnya, dilakukan evaluasi model, seperti pada Gambar 16. Terdapat fungsi *predict* yang digunakan untuk mengevaluasi performa model ANN terhadap data uji (*X_test*). Fungsi *predict* akan menghasilkan sebuah array yang berisi nilai prediksi untuk setiap data uji yang diberikan. Selanjutnya, nilai prediksi tersebut akan dikonversi menjadi sebuah array dengan mengambil nilai argumen maksimum dari setiap prediksi.



```
1 from sklearn import metrics
2 y_test = np.argmax(y_test, axis=1)
3
4 fl_test = metrics.f1_score(y_test, predict)
5 prec = metrics.precision_score(y_test, predict)
6 rec = metrics.recall_score(y_test, predict)
7 acc = metrics.accuracy_score(y_test, predict)
8
9 print ("F1 Score: {:.4f}.".format(fl_test))
10 print ("Precision: {:.4f}.".format(prec))
11 print ("Recall: {:.4f}.".format(rec))
12 print ("Accuracy: {:.4f}.".format(acc)) # note this
```

F1 Score: 0.9189.
Precision: 0.9273.
Recall: 0.9107.
Accuracy: 0.9820.

Gambar 16. Evaluasi Model

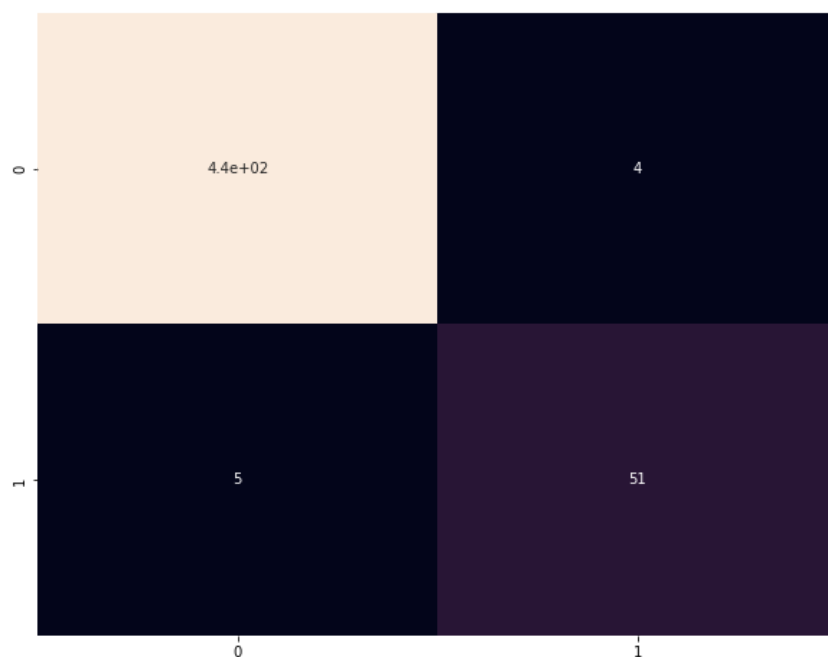
Setelah mendapatkan nilai prediksi, kode diatas akan menggunakan library sklearn untuk menghitung metric yang digunakan untuk mengevaluasi performa model ANN, yaitu *f1_score*, *precision_score*, *recall_score*, dan *accuracy_score*. Pada contoh *output* di atas, terlihat bahwa nilai dari metric tersebut cukup tinggi, dengan nilai *f1_score* sebesar 0.9189, *precision_score* sebesar 0.9273, *recall_score* sebesar 0.9107, dan *accuracy_score* sebesar 0.9820.

Nilai metric yang tinggi menunjukkan bahwa model ANN yang dibangun dan dilatih dapat memprediksi kemungkinan seorang pelanggan membeli pinjaman pribadi dengan tingkat akurasi yang cukup tinggi. Namun, perlu diingat bahwa tidak semua metric yang tinggi menunjukkan bahwa model tersebut performanya bagus. Dalam proyek ini, karena dataset yang digunakan tidak seimbang, maka metric *accuracy_score* tidak dapat dijadikan sebagai ukuran performa yang baik untuk model ANN tersebut.

Selanjutnya, digunakan *Confusion Matrix*, seperti pada Gambar 17. Terdapat fungsi *confusion_matrix* yang digunakan untuk menghitung nilai confusion matrix dari hasil prediksi model ANN terhadap data uji (*y_test*). Nilai *confusion matrix* terdiri dari TP (*true positive*), FP (*false positive*), FN (*false negative*), dan TN (*true negative*), yang mana masing-masing menunjukkan jumlah data yang benar dan salah prediksi oleh model ANN.

Pada contoh output di atas, terlihat bahwa nilai TP, FP, FN, dan TN cukup besar. Nilai TP menunjukkan bahwa jumlah data yang benar diprediksi sebagai positif sebanyak 4.4×10^2 , sedangkan nilai FP menunjukkan bahwa jumlah data yang salah diprediksi sebagai positif sebanyak 4. Nilai FN menunjukkan bahwa jumlah data yang salah diprediksi sebagai negatif sebanyak 5, sedangkan nilai TN menunjukkan bahwa jumlah data yang benar diprediksi sebagai negatif sebanyak 51.

Hal yang bisa diambil dari nilai *confusion matrix* tersebut adalah bahwa model ANN tersebut memiliki tingkat akurasi yang cukup tinggi dalam memprediksi kemungkinan seorang pelanggan membeli pinjaman pribadi. Namun, jumlah data yang salah diprediksi cukup besar, terutama data yang salah diprediksi sebagai positif. Hal ini menunjukkan bahwa model ANN tersebut mungkin memiliki tingkat false positive yang cukup tinggi. Oleh karena itu, perlu dilakukan evaluasi lebih lanjut terhadap model ANN tersebut untuk meningkatkan performanya.



Gambar 17. *Confusion Matrix*

Selanjutnya, akan dibuat *classification report*. Tujuannya adalah laporan kinerja model yang menunjukkan nilai *precision*, *recall*, dan *f1-score* dari model untuk setiap kelas yang diuji. *Precision* adalah rasio dari jumlah prediksi yang benar untuk setiap kelas terhadap total prediksi untuk kelas tersebut. *Recall* adalah rasio dari jumlah prediksi yang benar untuk setiap kelas terhadap total kelas yang sebenarnya ada dalam data uji. *F1-score* adalah rata-rata harmonik dari *precision* dan *recall*.

Dari *output* kode di atas, dapat dilihat bahwa nilai *precision*, *recall*, dan *f1-score* untuk kelas 0 adalah 0.99, 0.99, dan 0.99. Ini berarti bahwa model mampu memprediksi kelas 0 dengan tingkat akurasi yang tinggi, yaitu 99%. Nilai *precision*, *recall*, dan *f1-score* untuk

kelas 1 adalah 0.93, 0.91, dan 0.92. Ini berarti bahwa model mampu memprediksi kelas 1 dengan tingkat akurasi yang cukup tinggi, yaitu 91-93%.

Laporan kinerja model ini juga menunjukkan nilai akurasi (*accuracy*) dari model, yaitu rasio dari jumlah prediksi yang benar secara keseluruhan terhadap total prediksi yang dilakukan oleh model. Dari *output* kode di atas, dapat dilihat bahwa nilai akurasi model adalah 0.98. Ini berarti bahwa model mampu memprediksi seluruh data uji dengan tingkat akurasi yang tinggi, yaitu 98%.

Selain itu, laporan kinerja model ini juga menunjukkan nilai rata-rata (*average*) dari *precision*, *recall*, dan *f1-score* untuk semua kelas. Dari *output* kode di atas, dapat dilihat bahwa nilai rata-rata *precision*, *recall*, dan *f1-score* adalah 0.96, 0.95, dan 0.95. Ini berarti bahwa model mampu memprediksi seluruh kelas dalam data uji dengan tingkat akurasi yang cukup tinggi, yaitu 95-96%.

Dengan demikian, dapat disimpulkan bahwa model yang digunakan dalam laporan ini memiliki kinerja yang baik dalam memprediksi data uji, seperti pada Gambar 18. Nilai *precision*, *recall*, *f1-score*, dan akurasi yang tinggi menunjukkan bahwa model mampu memprediksi data uji dengan tingkat akurasi yang tinggi, baik untuk setiap kelas maupun secara keseluruhan.

```
[ ] 1 print(metrics.classification_report(y_test, predict))
```

	precision	recall	f1-score	support
0	0.99	0.99	0.99	444
1	0.93	0.91	0.92	56
accuracy			0.98	500
macro avg	0.96	0.95	0.95	500
weighted avg	0.98	0.98	0.98	500

Gambar 18. *Classification Report*

4. Kesimpulan

Rancangan dalam *Artificial Neural Network* (ANN) telah berhasil diterapkan untuk keputusan kredit Thera Bank. Pengaplikasian dalam rancangan ANN menggunakan multilayer. Berdasarkan hasil analisis menggunakan ANN dapat disimpulkan bahwa akurasi dari hasil prediksi yang didapatkan mencapai 98%. Maka dapat disimpulkan bahwasanya model yang digunakan dalam laporan ini memiliki kinerja yang baik dalam memprediksi data uji dan rancangan ANN berhasil diterapkan untuk keputusan kredit pada Thera Bank.

Referensi.

- [1] OJK (2021. Mei 24) *APA ITU KREDIT DAN PEMBIAYAAN* [online] Available <http://sikapiuangmu.ojk.go.id/FrontEnd/CMS/Article/316>
- [2] Aslam U, Aziz H I T, Sohail A and Batcha N K 2019 An empirical study on loan default prediction models *Journal of Computational and Theoretical Nanoscience* 16 pp 3483–8
- [3] Li Y 2019 Credit risk prediction based on machine learning methods *The 14th Int. Conf. on Computer Science & Education (ICCSE)* pp 1011–3
- [4] Zhu L, Qiu D, Ergu D, Ying C and Liu K 2019 A study on predicting loan default based on the random forest algorithm *The 7th Int. Conf. on Information Technol. and Quantitative Management (ITQM)* 162 pp 503–13
- [5] Komorowski, M., C., D., Marshall, Saliccioli, J. D., & Crutain, Y. (2016, October 13). *Exploratory Data Analysis*. 10.1007/978-3-319-43742-2_15