

## **TUGAS AKHIR SAINS DATA**

**Prediksi Gagal Bayar Pinjaman (*Default*) dengan Menggunakan  
Perbandingan Metode Klasifikasi *Decision Tree* dan *AdaBoost***



**DOSEN PENGAMPU:**  
**Dra. BEVINA DESJWIANDRA HANDARI, M.Sc., Ph.D.**

**DISUSUN OLEH:**

<b>DEDE AMANDA JULIETA</b>	<b>2206810111</b>
<b>HILMY RAHMADANI</b>	<b>2206810490</b>
<b>ANANDA RESTU AULIA</b>	<b>2206815756</b>
<b>AHDI NAZZALUL AHSAN</b>	<b>2206812956</b>

**PROGRAM STUDI MATEMATIKA**  
**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**  
**UNIVERSITAS INDONESIA**  
**2024**

## **ABSTRAK**

Risiko gagal bayar pinjaman merupakan masalah krusial dalam industri perbankan yang dapat berdampak signifikan pada stabilitas keuangan dan profitabilitas bank. Untuk memitigasi risiko ini, bank perlu mengembangkan sistem prediksi yang efektif guna mengidentifikasi nasabah yang berpotensi mengalami gagal bayar. Tujuan penulisan makalah ini adalah untuk membandingkan penggunaan 2 metode klasifikasi yang berbeda, yaitu *Decision Tree* dan *AdaBoost*, dalam memprediksi kemungkinan gagal bayar pinjaman. Hasilnya nanti akan menunjukkan bahwa metode *AdaBoost* mencapai akurasi yang lebih tinggi dibandingkan dengan *Decision Tree*. Implementasi perbandingan metode *Decision Tree* dan *AdaBoost* ini diharapkan dapat diterapkan secara lebih luas dalam sistem manajemen risiko kredit di bank atau platform tertentu, guna membantu dalam mengambil keputusan kredit yang lebih tepat dan mengurangi kerugian akibat gagal bayar.

**Kata Kunci:** *Data, Klasifikasi, Decision Tree, AdaBoost*

## **BAB I**

### **PENDAHULUAN**

#### **A. Latar Belakang**

Teknologi informasi terus berkembang pesat dan mempengaruhi segala aspek kehidupan manusia, termasuk pada sektor keuangan. Inovasi jasa layanan keuangan yang memanfaatkan penggunaan teknologi dan informasi di sektor keuangan disebut *Financial Technology (Fintech)*. Salah satu jenis *fintech* adalah *peer-to-peer lending* atau dikenal dengan pinjaman online, yaitu sebuah layanan jasa keuangan untuk mempertemukan pemberi pinjaman dengan penerima pinjaman dalam rangka melakukan perjanjian pinjam meminjam uang secara langsung melalui sistem elektronik.

Dalam beberapa tahun terakhir, jasa pinjaman online tengah marak terjadi di dunia. Hal ini karena pinjaman online menawarkan solusi yang cepat, mudah, dan fleksibel bagi peminjam. Meskipun demikian, risiko gagal bayar pinjaman juga dapat menjadi salah satu tantangan yang dihadapi dari kemajuan ini. Gagal bayar terjadi ketika peminjam tidak mampu memenuhi kewajiban pembayaran cicilan sesuai yang telah disepakati. Lembaga keuangan perlu meminimalisir terjadinya gagal bayar pinjaman dengan memprediksi kemungkinan terjadinya gagal bayar dari peminjam, sehingga dapat membantu dalam pengambilan keputusan kredit.

Penulis mencoba memprediksi gagal bayar pinjaman dengan memanfaatkan data historis peminjam menggunakan metode klasifikasi *decision tree*.

#### **B. Tujuan Penulisan**

1. Menganalisis faktor-faktor yang mempengaruhi kemungkinan gagal bayar.
2. Mengetahui kasus gagal bayar yang relevan dalam skala nasional dan internasional.
3. Mengimplementasi dan membuat model untuk prediksi gagal bayar dengan metode klasifikasi *decision tree* dan *AdaBoost*.
4. Memahami konsep *decision tree* dan *AdaBoost*.
5. Memenuhi tugas akhir mata kuliah Sains Data.

## **BAB II**

### **PEMBAHASAN**

#### **A. Data**

Dalam kasus ini, data diambil dari Kaggle (sumber: [Loan Default Prediction Dataset \(kaggle.com\)](#)) atau lebih tepatnya *Coursera's Loan Default Prediction Challenge*, yang mencakup informasi tentang pinjaman yang diberikan kepada nasabah. Data ini berisi berbagai variabel yang dapat mempengaruhi kemungkinan gagal bayar pinjaman. *Default* nantinya akan dipilih sebagai variabel target dan komponen variabel lainnya sebagai variabel prediktor. Pada dataset ini terdiri dari 255346 observasi, 17 kolom. Dari kolom-kolom tersebut, 2 kolom bertipe float, 8 kolom bertipe integer, dan 7 kolom bertipe object. Berikut adalah deskripsi detail mengenai variabel-variabel yang ada dalam dataset:

No	Nama Variabel	Deskripsi
0	Age	Usia peminjam
1	Income	Pendapatan tahunan peminjam
2	LoanAmount	Jumlah uang yang dipinjam
3	CreditScore	Skor kredit peminjam, menunjukkan kelayakan kredit mereka
4	MonthsEmployed	Jumlah bulan peminjam telah bekerja
5	NumCreditLines	Jumlah jalur kredit yang dimiliki
6	InterestRate	Suku bunga untuk pinjaman
7	LoanTerm	Jangka waktu pinjaman dalam bulan
8	DTIRatio	Rasio Utang terhadap Pendapatan, menunjukkan jumlah utang peminjam dibandingkan dengan pendapatan mereka
9	Education	Tingkat pendidikan tertinggi yang telah dicapai oleh peminjam

		(PhD, Master, Sarjana, SMA)
10	EmploymentType	Jenis status pekerjaan peminjam (Pekerja penuh waktu, Wiraswasta, Pengangguran)
11	MaritalStatus	Status pernikahan peminjam (lajang, menikah, bercerai).
12	HasMortgage	apakah peminjam memiliki hipotek (ya atau tidak)
13	HasDependents	Apakah peminjam memiliki tanggungan (Ya atau Tidak)
14	LoanPurpose	Tujuan dari pinjaman (Rumah, Mobil, Pendidikan, Bisnis, Lainnya)
15	HasCOSigner	apakah pinjaman memiliki penandatangan bersama (ya atau tidak)
16	Default	Default, variabel target biner yang menunjukkan apakah pinjaman gagal bayar (1) atau tidak (0)

## B. Studi Kasus

Masyarakat masih memanfaatkan akses pinjaman saat mengalami masalah keuangan. Banyak orang yang mengandalkan pinjaman online sebagai salah satu pilihan saat kondisi mendesak. Padahal konsekuensinya dapat berujung pada masalah hukum jika ada kegagalan dalam pembayaran. Otoritas Jasa Keuangan (OJK) menyebutkan bahwa data pinjaman online di Indonesia yang tercatat sekitar 155, sisanya ada lebih dari 5 ribu termasuk pinjaman online ilegal.

Faktanya adalah fenomena ini tidak hanya terjadi di Indonesia saja. Contohnya adalah Amerika Serikat (AS), terdapat platform pinjaman online yang menawarkan pinjaman pribadi tanpa jaminan hingga memiliki lebih dari 6 juta nasabah. Di Inggris juga memiliki bisnis pinjaman online yang diatur khusus dalam Policy Statement nomor PS19/14 oleh *Financial Conduct Authority*.

### C. Data Preparation

Langkah-langkah yang dilakukan sebelum membuat modelling dari metode yang digunakan sebagai berikut:

#### a. Data Collecting

Dataset dikumpulkan berdasarkan data yang telah ada/dibuat yang bersumber dari internet, yaitu dari web Kaggle.

#### b. Data Cleaning

*Data cleaning* dilakukan untuk memastikan bahwa dataset tidak mengandung nilai yang hilang (*missing values*) atau *outliers* yang tidak relevan. Jika terdapat missing values pada dataset, atasi dengan memanfaatkan metode imputasi mean atau median. Jika terdapat outliers, atasi dengan menghapus atau mengoreksinya.

#### c. Data Pre-Processing

- *Encoding data* dengan teknik *label encoder*, yaitu proses mengubah data kategorik menjadi data numerik agar lebih mudah dipahami oleh sistem. Pelabelan dilakukan dengan menggunakan angka dari 0 sampai N-1, dimana N adalah jumlah banyaknya sampel.
- Standarisasi adalah salah satu metode *feature scaling* yaitu mengubah data setiap fitur numerik memiliki mean mendekati 0 dan memiliki satuan standar deviasi, dengan rumus:

$$Z = \frac{X - \mu}{\sigma}$$

Z = Nilai standarisasi

X = Nilai fitur

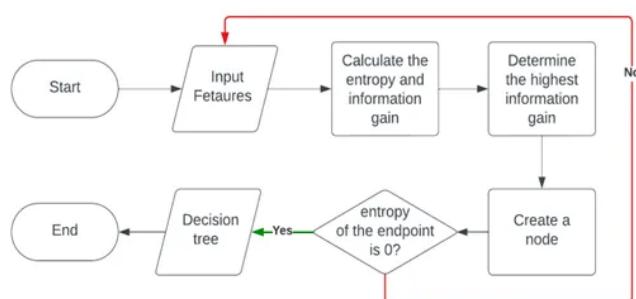
$\mu$  = Rata-rata fitur

$\sigma$  = Standar deviasi fitur

### D. Metode Decision Tree

#### a. Algoritma Decision Tree

*Decision tree* merupakan suatu metode yang dapat mengubah data kompleks menjadi model berbentuk pohon keputusan. Jenis algoritma yang umum digunakan, yaitu *Entropy*.



Gambar 1. Algoritma *Entropy Decision Tree*

Algoritma entropy memisahkan data pada *node root* menjadi dua subset (*node leaf*) yang dilakukan dengan cara sebagai berikut:

- 1) Input menerima masukan berupa angka-angka
- 2) Selanjutnya akan dihitung nilai entropy dari keseluruhan dataset.

$$Ent(D) = -\sum_{k=1}^{|y|} p_k \log_2 p_k$$

$k = 1, 2, \dots, |y|$

$p_k$  = proporsi kelas ke-k dalam dataset D

- 3) Untuk setiap fitur:

- Hitung nilai entropy pada nilai di masing-masing fitur.
- Hitung information gain, yaitu selisih antara entropy keseluruhan dataset dengan entropy setelah dataset dipartisi berdasarkan suatu fitur.

$$IG(D, a) = Ent(D) - \sum_{v=1}^{|D^v|} \frac{|D^v|}{|D|} Ent(D^v)$$

$D^v$  = subset dari D untuk setiap nilai v dari fitur a

$a = \{a_1, a_2, \dots, a_v\}$ .

- Pilih fitur dengan information gain terbesar.
- Lakukan kembali proses hingga semua dataset yang diamati dapat diklasifikasikan.

#### b. Hyperparameter Tuning

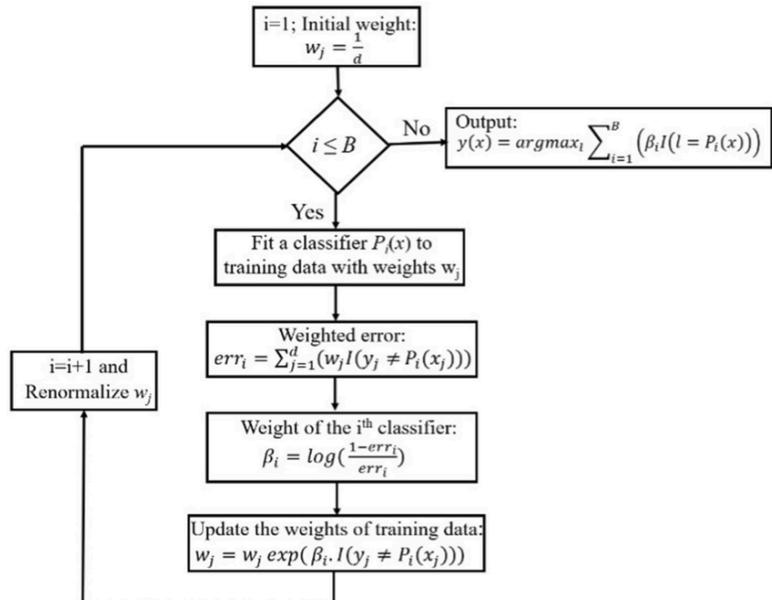
*Hyperparameter tuning* digunakan untuk mengoptimalkan kinerja model *decision tree* dengan menggunakan teknik *grid search*. Data ini menggunakan tiga parameter:

- 1) **Max Depth**, parameter ini mengontrol kedalaman maksimal pohon (jumlah total lapisan dari akar hingga daun terdalam). *Max depth* dapat membantu menyeimbangkan antara *overfitting* dan *underfitting* jika terjadi pada data.
- 2) **Min Samples Split**, parameter ini mengontrol jumlah minimum sampel yang diperlukan untuk memisahkan node. *Min samples split* dapat membantu mencegah pohon membuat cabang dengan sampel yang sangat sedikit.
- 3) **Min Samples Leaf**, parameter ini mengontrol jumlah minimum sampel yang diperlukan pada node daun. Simpul daun adalah simpul yang tidak memiliki cabang. *Min samples leaf* dapat membantu mencegah pohon membuat cabang dengan sampel yang sangat sedikit.

### E. Metode AdaBoost

#### a. Algoritma AdaBoost

*AdaBoost* adalah metode untuk meningkatkan kemampuan prediksi dengan menggabungkan beberapa *weak learners* (model lemah) berbentuk *decision stump* menjadi satu *strong learner* (model kuat). Konsepnya membangun sebuah model dengan menyesuaikan bobot sampel pada dataset *training*, kemudian membangun model kedua dan seterusnya untuk memperbaiki kesalahan yang ada pada model sebelumnya.



Gambar 2. Algoritma AdaBoost

Langkah-langkahnya sebagai berikut:

- 1) Inisialisasikan bobot awal pada setiap sampel dari dataset sebanyak d sampel.

$$w_j = \frac{1}{d}; j = 1, 2, \dots, d$$

- 2) Lakukan training weak classifier pada setiap iterasi untuk mengklasifikasikan sampel sebagai positif atau negatif. Dapat dengan menghitung entropy dan membandingkan information gain antarfitur seperti yang dilakukan pada decision tree.
- 3) Evaluasi performa dengan menghitung error.

$$\epsilon_i = \sum_{j=1}^d w_j I$$

I adalah fungsi indikator dari banyaknya data yang mengalami ketidaksamaan antara prediksi model *stump* dengan yang ada pada sampel.

- 4) Hitung bobot dari weak classifier sebagai prediksi final untuk model stump ke-i.

$$\beta_i = \log\left(\frac{1 - \epsilon_i}{\epsilon_i}\right)$$

- 5) Perbarui bobot dengan meningkatkan bobot pada sampel yang salah diklasifikasikan.

$$w_j = w_j e^{\beta_i}$$

- 6) Normalisasi agar total bobot yang baru sama dengan 1.

$$w_{j+1} = \frac{w_j e^{\beta_i}}{\sum_{j=1}^d w_j e^{\beta_i}}$$

- 7) Bobot yang telah dinormalisasi akan menjadi bobot baru untuk acuan pada model stump yang ke i = 2, .., B

- 8) Lakukan kembali proses hingga semua dataset yang diamati dapat diklasifikasikan. Jumlah bobot yang terbanyak dari prediksi positif atau negatif akan menjadi hasil final.

#### b. *Hyperparameter Tuning*

*Hyperparameter tuning* digunakan untuk mengoptimalkan kinerja model *AdaBoost* dengan menggunakan teknik *grid search*. Data ini menggunakan dua parameter:

- 1) ***N Estimators***, digunakan untuk menentukan jumlah *weak learners* yang akan di-training secara iteratif.
- 2) ***Learning Rate***, yaitu bobot dari setiap *weak learners* yang akan mengontrol kontribusi setiap pengklasifikasian.

Terdapat *trade-off* antara *N Estimator* dan *Learning Rate*, dimana model dengan *Learning Rate* lebih kecil akan memerlukan lebih banyak *N Estimator* untuk mencapai kinerja yang paling baik.

### F. Model Evaluasi

#### a. Akurasi

Akurasi akan mengukur seberapa sering pengklasifikasi memprediksi dengan benar. Semakin tinggi nilai akurasi, maka model tersebut memiliki performa yang sangat baik.

		Actual Values		$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$
		Positive (1)	Negative (0)	
Predicted Values	Positive (1)	TP	FP	
	Negative (0)	FN	TN	

#### b. ROC-AUC

- *ROC (Receiver Operating Characteristic) Curve* adalah grafik yang memplot antara True Positive Rate dan False Positive Rate.
  - 1) Sensitivitas/Recall untuk mengetahui proporsi kelas positif yang diklasifikasikan dengan benar.
  - 2) False Positive Rate (FPR) untuk mengetahui proporsi kelas negatif yang salah diklasifikasikan.

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

- AUC (Area Under the Curve) adalah luas area dibawah kurva ROC. Pada nilai  $0,5 < \text{AUC} < 1$  artinya pengklasifikasian akan menghasilkan model seimbang. Pada nilai  $\text{AUC} = 0,5$  artinya menghasilkan model dengan nilai yang acak. Pada nilai  $0 < \text{AUC} < 0,5$  artinya menghasilkan model yang tidak seimbang.

$$FPR = \frac{FP}{TN + FP} :$$

### c. F1 Score

F1 score digunakan untuk mengatasi ketidakseimbangan antara kelas positif dan negatif dalam dataset.

$$F1 = 2 \cdot \frac{Precision \times Recall}{Precision + Recall}$$

$$\text{dimana, } Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

## BAB III

### IMPLEMENTASI DAN ANALISIS DATA

#### A. Pre-Processing

##### a. Data Collecting

Data Description																	
	Age	Income	LoanAmount	CreditScore	MonthsEmployed	NumCreditLines	InterestRate	LoanTerm	DTIRatio	Education	EmploymentType	MaritalStatus	HasMortgage	HasDependents	LoanPurpose	HasCoSigner	Default
0	56	85994	50587	520	80	4	15.23	36	0.44	Bachelor's	Full-time	Divorced	Yes	Yes	Other	Yes	0
1	69	50432	124440	458	15	1	4.81	60	0.68	Master's	Full-time	Married	No	No	Other	Yes	0
2	46	84208	129188	451	26	3	21.17	24	0.31	Master's	Unemployed	Divorced	Yes	Yes	Auto	No	1
3	32	31713	44799	743	0	3	7.07	24	0.23	High School	Full-time	Married	No	No	Business	No	0
4	60	20437	9139	633	8	4	6.51	48	0.73	Bachelor's	Unemployed	Divorced	No	Yes	Auto	No	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
255342	19	37979	210682	541	109	4	14.11	12	0.85	Bachelor's	Full-time	Married	No	No	Other	No	0
255343	32	51953	189899	511	14	2	11.55	24	0.21	High School	Part-time	Divorced	No	No	Home	No	1
255344	56	84820	208294	597	70	3	5.29	60	0.50	High School	Self-employed	Married	Yes	Yes	Auto	Yes	0
255345	42	85109	60575	809	40	1	20.90	48	0.44	High School	Part-time	Single	Yes	Yes	Other	No	0
255346	62	22418	18481	636	113	2	6.73	12	0.48	Bachelor's	Unemployed	Divorced	Yes	No	Education	Yes	0

Gambar 3. Tampilan Dataset Utama

Upload dataset terlebih dahulu yang akan diolah dan definisikan dataset dengan nama variabel “df”. Dilakukan penghapusan kolom “LoanID” karena variabel tersebut tidak berpengaruh terhadap variabel target “Default”

	Age	Income	LoanAmount	CreditScore	MonthsEmployed	NumCreditLines	InterestRate	LoanTerm	DTIRatio	Default
count	255347.000000	255347.000000	255347.000000	255347.000000	255347.000000	255347.000000	255347.000000	255347.000000	255347.000000	255347.000000
mean	43.498306	82499.304597	127578.865512	574.264346	59.541976	2.501036	13.492773	36.025894	0.500212	0.116128
std	14.990258	38963.013729	70840.706142	158.903867	34.643376	1.117018	6.636443	16.969330	0.230917	0.320379
min	18.000000	15000.000000	5000.000000	300.000000	0.000000	1.000000	2.000000	12.000000	0.100000	0.000000
25%	31.000000	48825.500000	66156.000000	437.000000	30.000000	2.000000	7.770000	24.000000	0.300000	0.000000
50%	43.000000	82466.000000	127556.000000	574.000000	60.000000	2.000000	13.460000	36.000000	0.500000	0.000000
75%	56.000000	116219.000000	188985.000000	712.000000	90.000000	3.000000	19.250000	48.000000	0.700000	0.000000
max	69.000000	149999.000000	249999.000000	849.000000	119.000000	4.000000	25.000000	60.000000	0.900000	1.000000

Gambar 4. Deskripsi Statistik Numerik

Gambar di atas merupakan deskripsi statistik variabel-variabel numerik dari dataset

	Education	EmploymentType	MaritalStatus	HasMortgage	HasDependents	LoanPurpose	HasCoSigner
count	255347	255347	255347	255347	255347	255347	255347
unique	4	4	3	2	2	5	2
top	Bachelor's	Part-time	Married	Yes	Yes	Business	Yes
freq	64366	64161	85302	127677	127742	51298	127701

Gambar 5. Deskripsi Statistik Kategorik

Gambar di atas merupakan deskripsi statistik variabel-variabel kategorik dari dataset

```

RangeIndex: 255347 entries, 0 to 255346
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Age          255347 non-null   int64  
 1   Income        255347 non-null   int64  
 2   LoanAmount    255347 non-null   int64  
 3   CreditScore   255347 non-null   int64  
 4   MonthsEmployed 255347 non-null   int64  
 5   NumCreditLines 255347 non-null   int64  
 6   InterestRate  255347 non-null   float64 
 7   LoanTerm       255347 non-null   int64  
 8   DTIRatio       255347 non-null   float64 
 9   Education      255347 non-null   object  
 10  EmploymentType 255347 non-null   object  
 11  MaritalStatus  255347 non-null   object  
 12  HasMortgage    255347 non-null   object  
 13  HasDependents  255347 non-null   object  
 14  LoanPurpose    255347 non-null   object  
 15  HasCoSigner   255347 non-null   object  
 16  Default         255347 non-null   int64  
dtypes: float64(2), int64(8), object(7)

```

**Gambar 6. Informasi Dataset**

Ukuran dari *dataset* yang digunakan yaitu 255347 baris dan 17 kolom. Deskripsi *dataset* yang digunakan, terlihat bahwa tidak ada *missing value* pada *dataset* karena semua fitur memiliki semua data.

	missing_sum	missing_rate	dtype
<b>Age</b>	0	0.0	int64
<b>Income</b>	0	0.0	int64
<b>LoanAmount</b>	0	0.0	int64
<b>CreditScore</b>	0	0.0	int64
<b>MonthsEmployed</b>	0	0.0	int64
<b>NumCreditLines</b>	0	0.0	int64
<b>InterestRate</b>	0	0.0	float64
<b>LoanTerm</b>	0	0.0	int64
<b>DTIRatio</b>	0	0.0	float64
<b>Education</b>	0	0.0	object
<b>EmploymentType</b>	0	0.0	object
<b>MaritalStatus</b>	0	0.0	object
<b>HasMortgage</b>	0	0.0	object
<b>HasDependents</b>	0	0.0	object
<b>LoanPurpose</b>	0	0.0	object
<b>HasCoSigner</b>	0	0.0	object
<b>Default</b>	0	0.0	int64

**Gambar 7. Jumlah Data Hilang**

*Tidak ada data yang hilang atau bernilai “NaN”.*

## b. Data Preparation

Label encoding untuk data kategorik

```
Education
{"Bachelor's": 0, "High School": 1, "Master's": 2, "PhD": 3}
EmploymentType
{'Full-time': 0, 'Part-time': 1, 'Self-employed': 2, 'Unemployed': 3}
MaritalStatus
{'Divorced': 0, 'Married': 1, 'Single': 2}
HasMortgage
{'No': 0, 'Yes': 1}
HasDependents
{'No': 0, 'Yes': 1}
LoanPurpose
{'Auto': 0, 'Business': 1, 'Education': 2, 'Home': 3, 'Other': 4}
HasCoSigner
{'No': 0, 'Yes': 1}
```

Gambar 8. Label Encoding variabel kategorik

Untuk data yang bersifat kategorik, akan dipresentasikan data kategorik sebagai 0, 1, 2, 3, atau 4.

```
# copy of datasets
df3 = df2_onehot.copy()

# apply standardization on numerical features
stand_cols = set(float_col) - set(['Default']) #Karena kita tidak perlu men standardisasi variabel target

for i in stand_cols:
    scale = StandardScaler() # fit on training data column
    df3[i] = scale.fit_transform(df3[[i]]) # transform the data column

df3
```

Gambar 9. Code Standarisasi Data

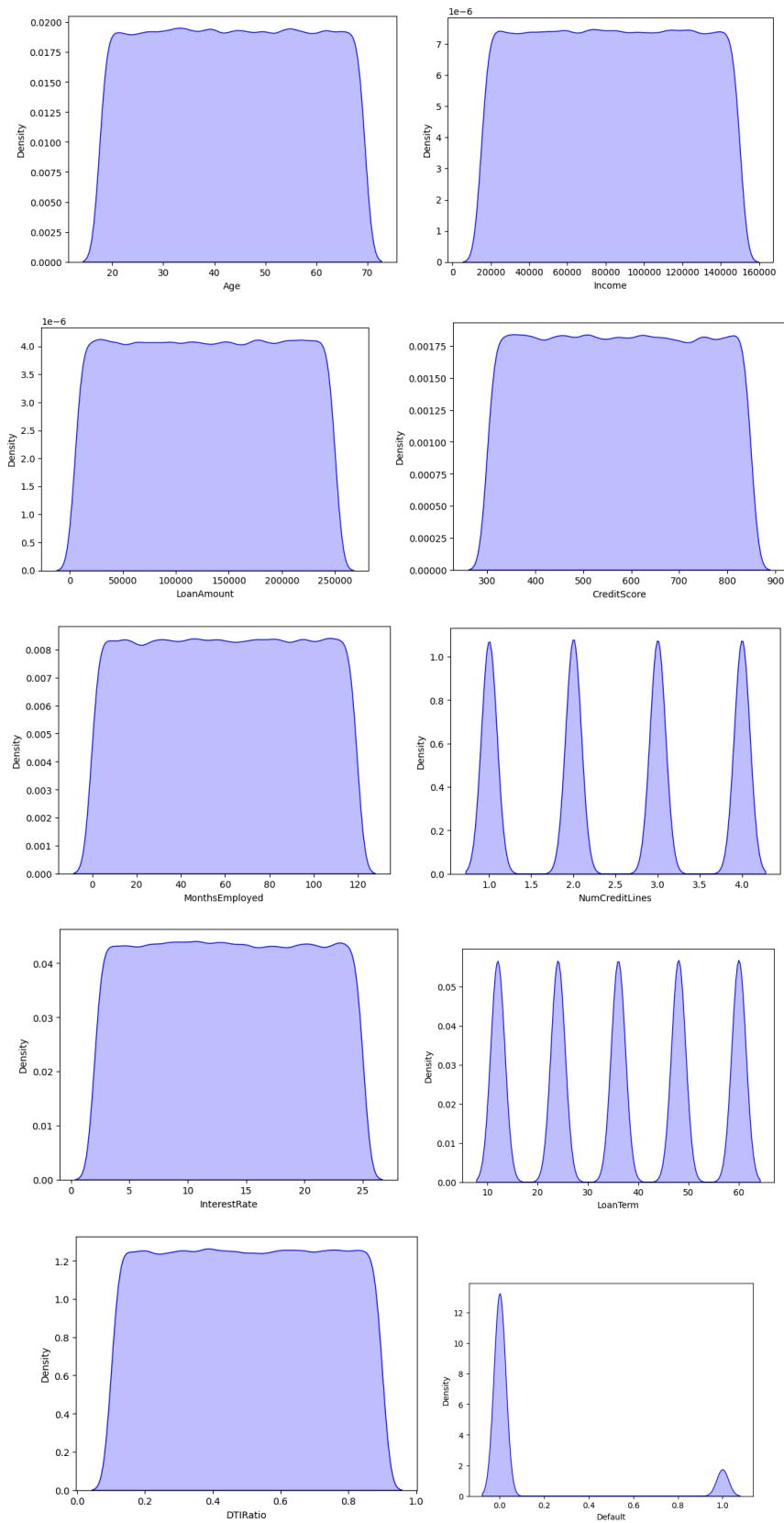
Proses standarisasi data numerik untuk memodifikasi nilai dalam variabel sehingga dapat diukur dalam skala umum. Skala yang dipakai 0-1. Berikut *dataset* yang sudah dilakukan *label encoding* untuk data kategori dan sudah dinormalisasi untuk data numerik.

	Age	Income	LoanAmount	CreditScore	MonthsEmployed	NumCreditLines	InterestRate	LoanTerm	DTRatio	Education	EmploymentType	MaritalStatus	HasMortgage	HasDependents	LoanPurpose	HasCoSigner	Default
0	0.833990	0.089693	-1.066833	-0.341492	0.590533	1.341937	0.261771	-0.001526	-0.260753	0	0	0	0	1	1	4	1 Not Default
1	1.701221	-0.820201	-0.044309	-0.731666	-1.285731	-1.343791	-1.308350	1.412793	0.778585	2	0	1	0	0	4	1 Not Default	
2	0.166688	0.043854	0.022715	-0.775718	-0.968209	0.446694	1.156831	-0.708685	-0.823728	2	3	0	1	1	0	0 Default	
3	-0.767053	-1.303452	-1.168638	1.061875	-1.718715	0.446694	-0.967805	-0.708685	-1.170174	1	0	1	0	0	1	0 Not Default	
4	1.100830	-1.592855	-1.671921	0.369631	-1.487790	1.341937	-1.052188	0.705634	0.995114	0	3	0	0	1	0	0 Not Default	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
255342	-1.634285	-1.142632	1.173101	-0.209337	1.427638	1.341937	0.093006	-1.415845	1.514783	0	0	1	0	0	4	0 Not Default	
255343	-0.767053	-0.783984	0.679724	-0.398130	-1.314597	-0.446694	-0.297744	-0.708685	-1.256785	1	1	0	0	0	3	0 Default	
255344	0.833990	0.059562	1.139391	0.143078	0.301877	0.446694	-1.236022	1.412793	-0.000918	1	2	1	1	1	0	1 Not Default	
255345	-0.099952	0.066979	-0.945840	1.477221	-0.564091	-1.343791	1.116146	0.705634	-0.260753	1	1	2	1	1	4	0 Not Default	
255346	1.234250	-1.542012	-1.540048	0.388510	1.543098	-0.446694	-1.019038	-1.415845	-0.087530	0	3	0	1	0	2	1 Not Default	

Gambar 10. Tampilan data setelah preprocessing

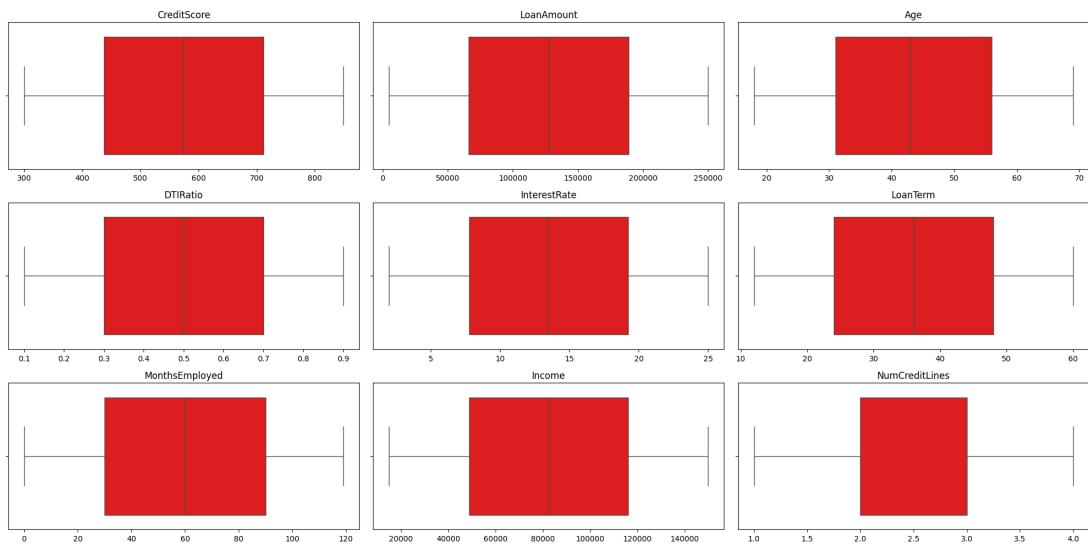
## B. Exploratory Data Analysis (EDA)

Berikut adalah visualisasi distribusi setiap variabel numerik dari *dataset*:



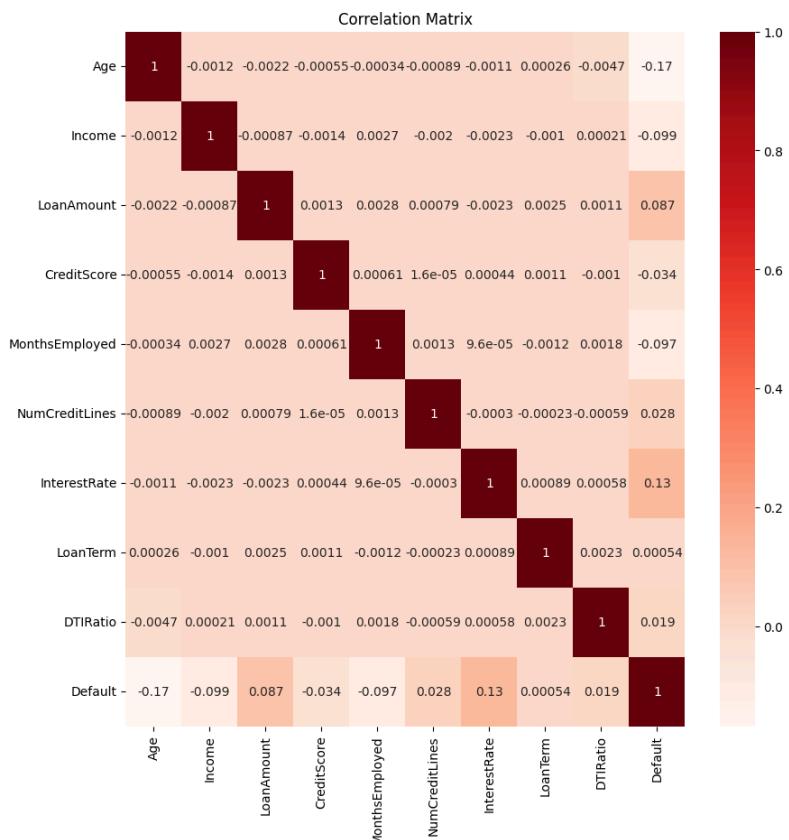
**Gambar 11. Distribusi Data Numerik**

Kemudian, selanjutnya akan di cek distribusi *box plot* untuk mengidentifikasi *outlier* dalam data gagal bayar pinjaman.



**Ganbar 12. Distribusi Box Plot**

Berdasarkan distribusi *box plot*, dapat dilihat bahwa tidak terdapat *outlier* atau nilai yang jauh berbeda dari nilai-nilai lain dalam data. Serta, tidak terdapat pula *skew* atau kemiringan pada distribusi data, karena median berada di tengah kotak. Setelah itu, akan ditampilkan korelasi antara variabel dalam bentuk matriks berwarna, biasanya disebut dengan distribusi *heatmap*.



**Gambar 13. Distribusi Heatmap**

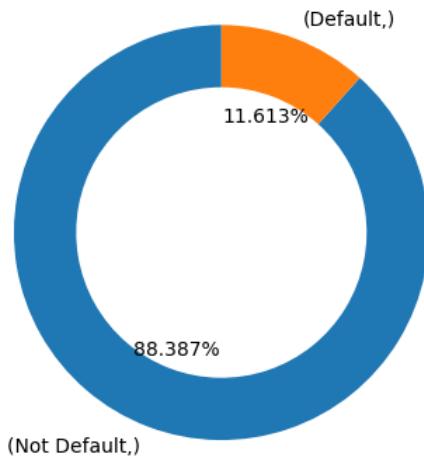
Pada distribusi *heatmap* terlihat korelasi negatif atau positif antar variabel pada dataset. Dengan contoh, fitur *default* dan *interest rate* berkorelasi positif (0.13), *default* dan *age* berkorelasi negatif (-0.17), dan *loan term* berkorelasi paling lemah dengan *default* karena nilainya semakin mendekati 0. Selanjutnya, akan ditunjukkan persebaran nilai unik pada fitur-fitur kategorikal dengan menggunakan distribusi *pie plot*.



**Gambar 14. Distribusi Pie Plot**

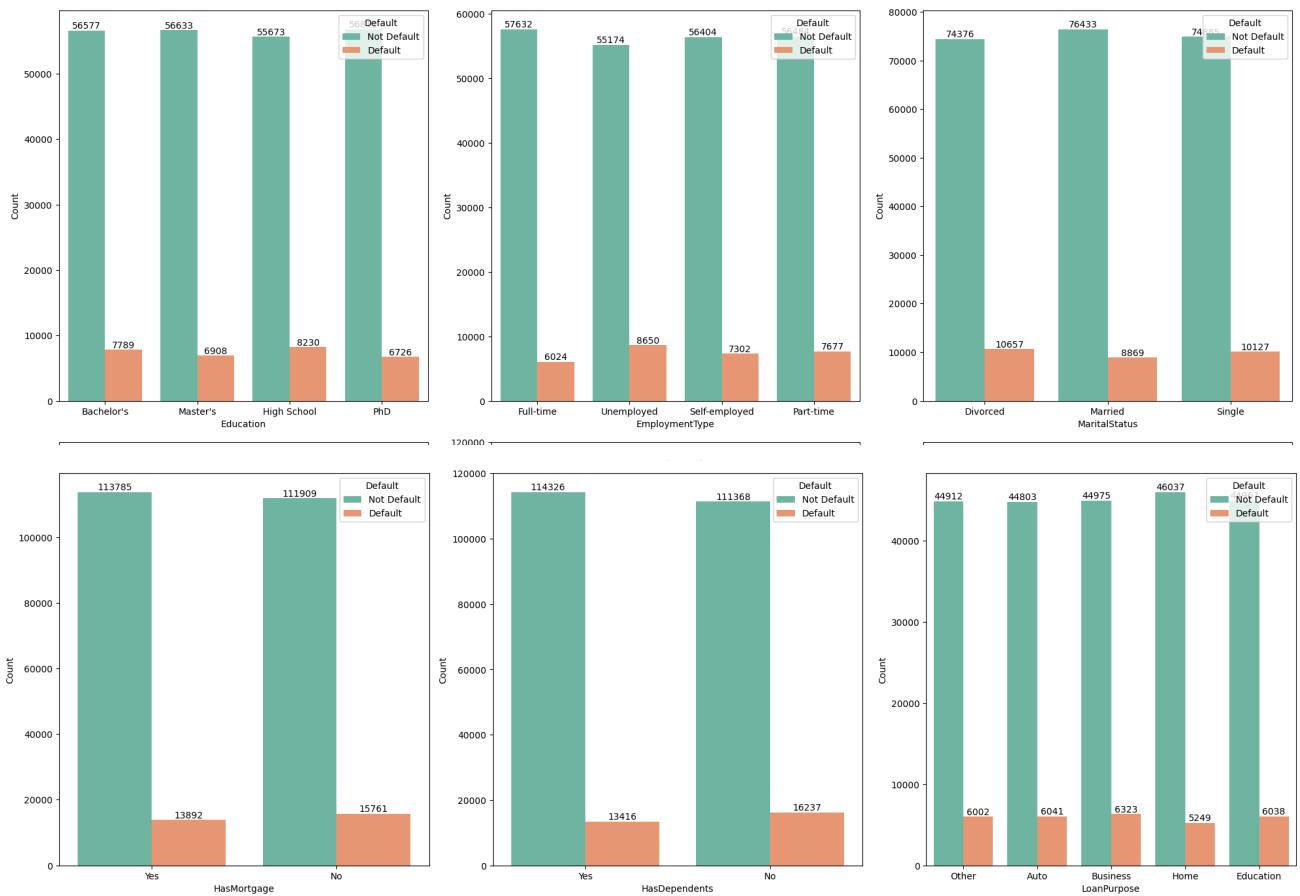
Terlihat proporsi atau distribusi fitur-fitur kategorik pada gambar diatas menunjukan seimbang. Berbeda halnya dengan distribusi *pie plot* data target (*default*) yang menunjukan proporsi datanya tidak seimbang, antara data *default* dan data *not default*.

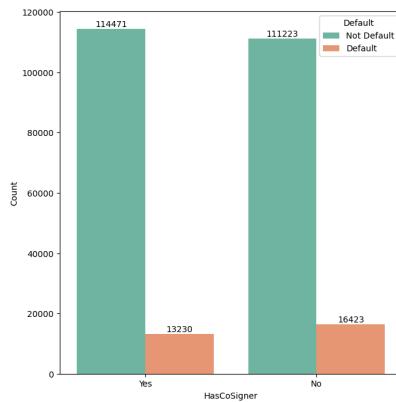
**Distribution of Default**



**Gambar 14. Distribusi Pie Plot Data Default**

Dengan menggunakan berbagai teknik statistik dan visualisasi, selanjutnya dataset dapat dianalisis bagaimana kategori dalam suatu variabel independen (prediktor) berhubungan dengan variabel target (dependent).





Gambar 15. Perbandingan Variabel Kategorik dengan Target

### C. Modelling Decision Tree

Prediksi masalah yang akan digunakan yaitu dengan klasifikasi biner dengan target yang ingin diprediksi adalah apakah terkategorikan Gagal bayar (*Default*) atau Tidak gagal bayar (*Not Default*). Berikut merupakan *coding* yang digunakan untuk menyelesaikan masalah:

```
▶ X = df3.drop(['Default'], axis = 1)
y = df3.Default
```

Gambar 16. Pembagian data independen dengan dependen (target)

Definisikan data-data yang ada pada kolom fitur selain kolom *Default* sebagai X dan definisikan kolom *Default* sebagai Y.

```
▶ le2 = LabelEncoder()
y_le = le2.fit_transform(y)
y_le = pd.DataFrame(y_le)
y_le
```

	0
0	1
1	1
2	0
3	1
4	1
...	...
255342	1
255343	0
255344	1
255345	1
255346	1

255347 rows × 1 columns

Gambar 17. Label encoding variabel target

Dilakukan *label encoding* pada variabel target, memanfaatkan *library scikit-learn*.

```
▶ SEED = 42
X_train, X_val, y_train, y_val = train_test_split(X, y_le, random_state = SEED, stratify=y, test_size=0.3)
```

Gambar 18. Pembagian Data *training* dan *test*

Dilakukan pembagian data latihan (*training*) dan tes (*test*)

```
▶ dectree = tree.DecisionTreeClassifier(criterion = 'entropy', max_depth = 1, random_state = SEED)
dectree.fit(X_train, y_train)

→ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=1, random_state=42)
```

Gambar 19. Model *Decision Tree Classifier*

Arsitektur model pohon keputusan (*Decision Tree*) dengan parameter *criterion* = *entropy*, *max\_depth* = 1, dan *random\_state* = 42. kemudian dilakukan *fitting model* terhadap data pelatihan

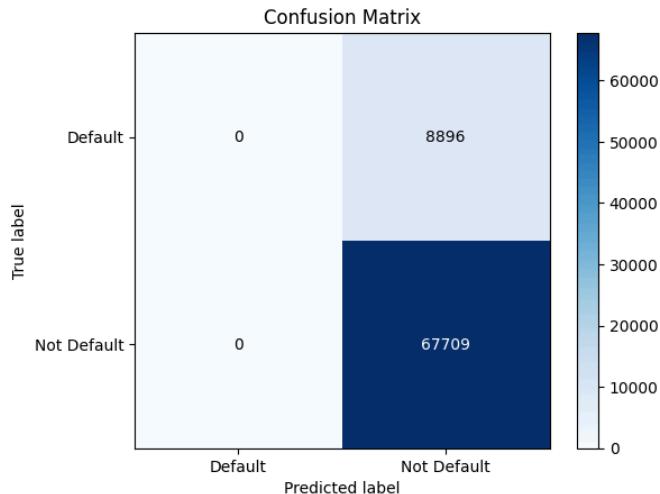
```
▶ acc = accuracy_score(y_val, y_pred_dectree)
f1 = f1_score(y_val, y_pred_dectree)
roc_auc = roc_auc_score(y_val, dectree.predict_proba(X_val)[:, 1])

print("accuracy score: ", acc)
print("f1 score: ", f1)
print("ROC-AUC score: ", roc_auc)

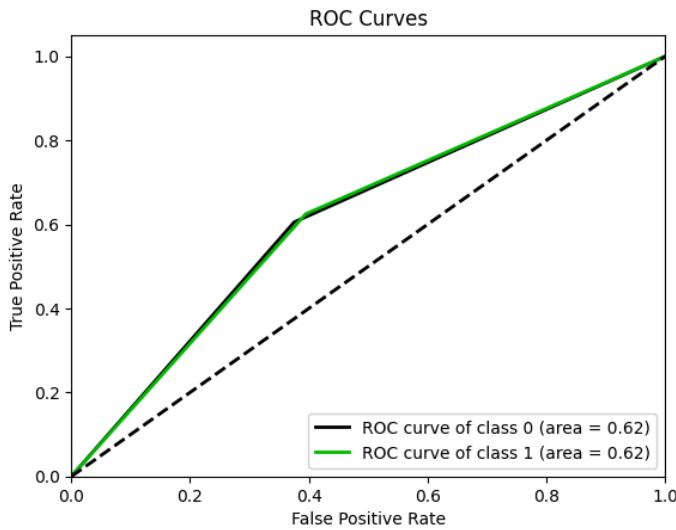
→ accuracy score:  0.8838718099340774
f1 score:  0.9383566389955237
ROC-AUC score:  0.6150155047836962
```

Gambar 20. Code dan output skor metrik *decision tree*

Didapat skor akurasi, skor f1, dan skor ROC-AUC dari model pohon keputusan pertama

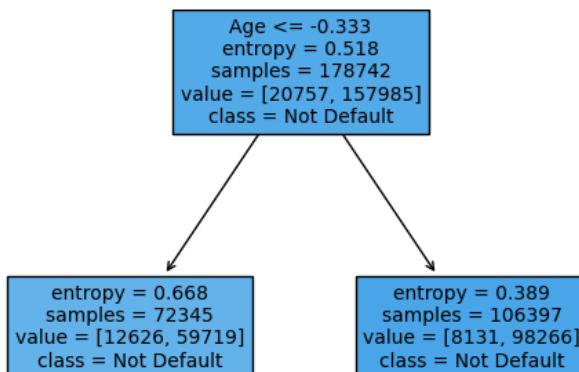


Gambar 21. Confusion matrix model *decision tree*



Gambar 22. kurva ROC model *decision tree*

Didapat juga hasil *confusion matrix* (Gambar 21) dari model pohon keputusan pertama dan plot kurva ROC (Gambar 22).



Gambar 23. Pohon Keputusan *Decision Tree*

#### D. Decision Tree dengan Hyperparameter Tuning

```

[118]: dectree_stand_param_grid = {
        'max_depth': range(1, 10),
        "min_samples_split": range(2, 10),
        'min_samples_leaf': range(2,10)
    }

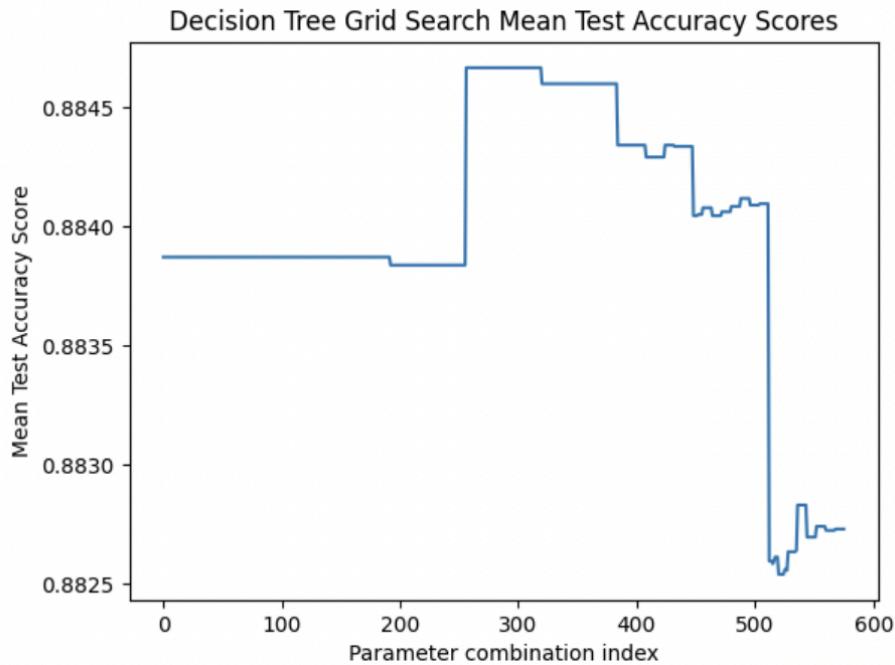
    dectree_grid_search = GridSearchCV(dectree, dectree_stand_param_grid, cv = 3,
                                         n_jobs = -1, verbose = 1, scoring='accuracy')
    dectree_grid_search.fit(X_train, y_train)

    ➜ Fitting 3 folds for each of 576 candidates, totalling 1728 fits
    ▶ GridSearchCV
    ▶ estimator: DecisionTreeClassifier
        ▶ DecisionTreeClassifier
  
```

Gambar 24. Algoritma dan output *GridSearchCV*

Bagian ini mendefinisikan parameter grid yang akan dicari oleh *GridSearchCV*. Tiga parameter utama dari model Decision Tree yang dipertimbangkan adalah:

- *max\_depth*: Kedalaman maksimal dari pohon keputusan. Nilai yang akan dicoba adalah dari 1 hingga 9.
- *min\_samples\_split*: Jumlah minimum sampel yang diperlukan untuk membagi node internal. Nilai yang akan dicoba adalah dari 2 hingga 9.
- *min\_samples\_leaf*: Jumlah minimum sampel yang harus dimiliki oleh node daun. Nilai yang akan dicoba adalah dari 2 hingga 9.



Gambar 25. Hasil skor akurasi *GridSearchCV Decision Tree*

Bagian ini membuat plot garis dari *mean\_test\_score*, yaitu rata-rata skor akurasi dari setiap kombinasi hyperparameter yang diuji selama *Grid Search*. Hasil rata-rata skor akurasi seluruh eksperimen

```
print(dectree_grid_search.best_params_)

{'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 2}
```

Gambar 26. Parameter terbaik dari *GridSearchCV*

Kode dan hasil yang diberikan menunjukkan cara untuk menampilkan hyperparameter terbaik yang ditemukan oleh *GridSearchCV* setelah proses pencarian selesai.

```
dectree_hypetune = tree.DecisionTreeClassifier(criterion = 'entropy', **dectree_grid_search.best_params_, random_state = SEED)
dectree_hypetune.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=5, min_samples_leaf=2,
                     random_state=42)
```

Gambar 27. Model *Decision Tree Classifier* dengan *Hyperparameter Tuning*

Menggunakan parameter terbaik yang ditemukan oleh *GridSearchCV* untuk melatih model *Decision Tree*

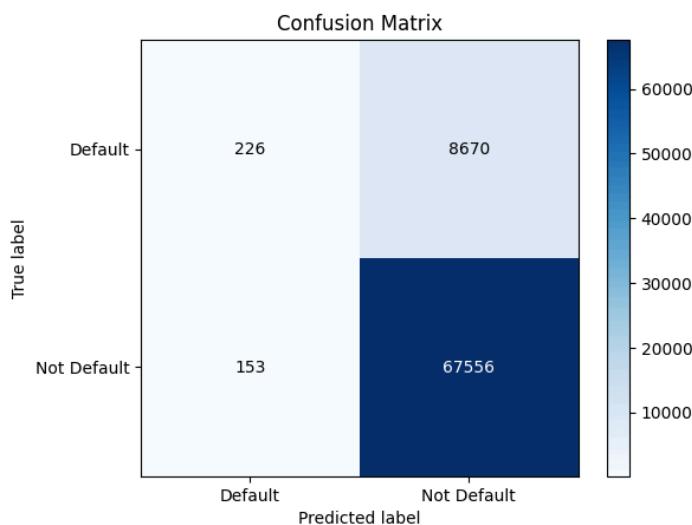
```
[85] y_pred_dectree_hypetune = dectree_hypetune.predict(X_val)
    acc = accuracy_score(y_val, y_pred_dectree_hypetune)
    f1 = f1_score(y_val, y_pred_dectree_hypetune)
    roc_auc = roc_auc_score(y_val, dectree_hypetune.predict_proba(X_val)[:, 1])

    print("accuracy score: ", acc)
    print("f1 score: ", f1)
    print("ROC-AUC score: ", roc_auc)

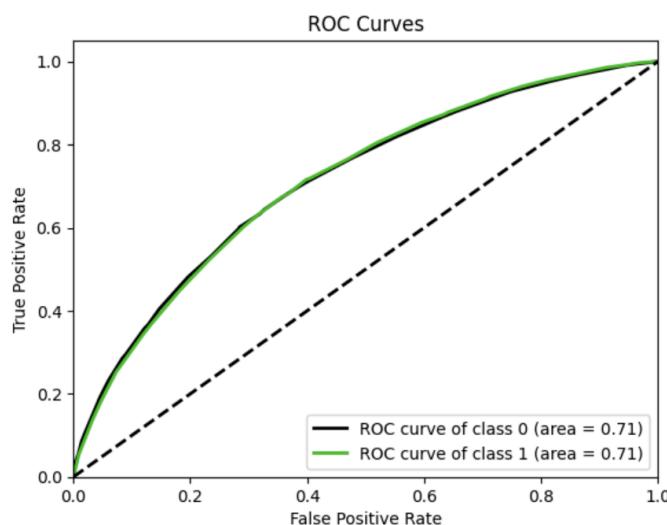
→ accuracy score: 0.8848247503426669
f1 score: 0.938701497203599
ROC-AUC score: 0.7134008376714422
```

**Gambar 28. Code dan output skor metrik *decision tree Hyperparameter Tuning***

Kode di atas digunakan untuk mengevaluasi performa model Decision Tree yang telah dilatih dengan hyperparameter terbaik pada data validasi.

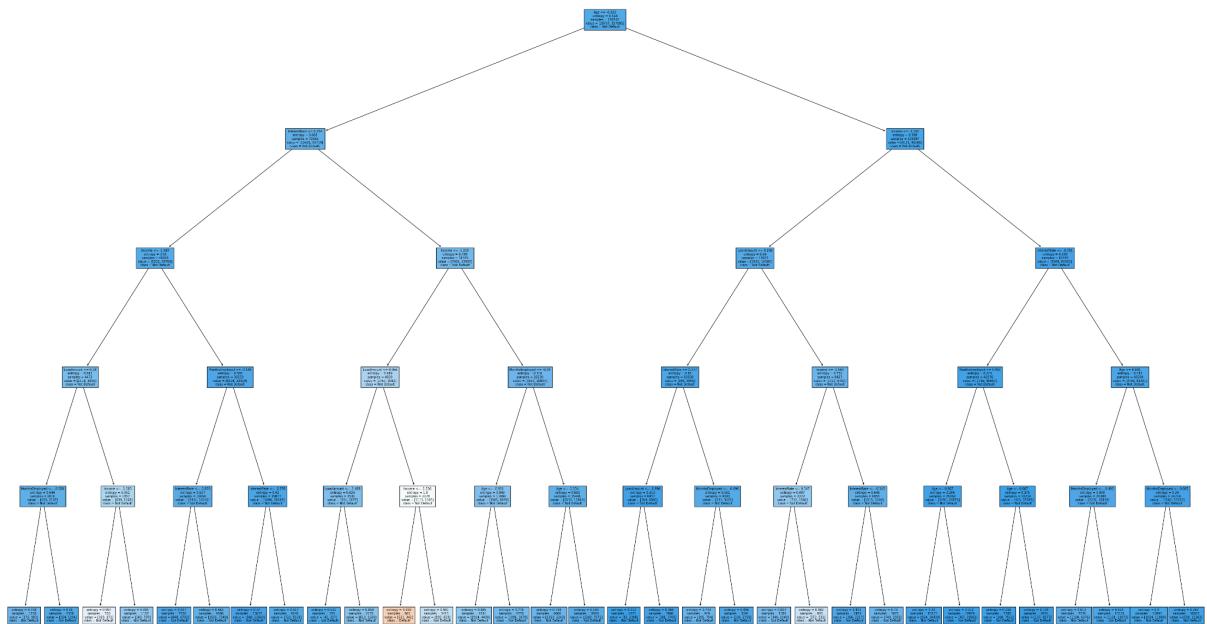


**Gambar 29. Confusion Matrix decision tree setelah Hyperparameter Tuning**



**Gambar 30. Kurva ROC decision tree setelah Hyperparameter Tuning**

Didapat *confusion matrix* (Gambar 29) dan plot kurva ROC (Gambar 30) dari model pohon keputusan dengan parameter terbaik hasil *Hyperparameter tuning*.



Gambar 31. Visualisasi model *Decision Tree*

## E. Modelling AdaBoost

algoritma ensemble AdaBoost untuk meningkatkan kinerja model Decision Tree.

```
▶ from sklearn.ensemble import AdaBoostClassifier
  adaboost = AdaBoostClassifier(decTree, n_estimators=50, random_state=SEED)
  adaboost.fit(X_train, y_train)

  ▶ AdaBoostClassifier
  ▶ estimator: DecisionTreeClassifier
    ▶ DecisionTreeClassifier
```

Gambar 32. Algoritma *AdaBoost*

Pertama dilakukan importasi model *AdaBoostClassifier* dari module *scikit-learn*. Model AdaBoost dibuat dengan menggunakan *Decision Tree* tinggi 1 sebagai *base estimator*. Model dilatih menggunakan data pelatihan (*X\_train*, *y\_train*)

```

▶ y_adaboost_1 = adaboost.predict(X_val)
acc = accuracy_score(y_val, y_adaboost_1)
f1 = f1_score(y_val, y_adaboost_1)
roc_auc = roc_auc_score(y_val, adaboost.predict_proba(X_val)[:, 1])

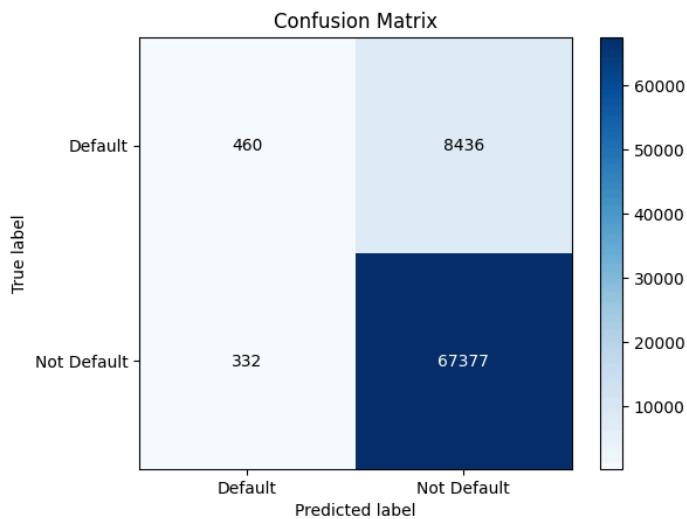
print("accuracy score: ", acc)
print("f1 score: ", f1)
print("ROC-AUC score: ", roc_auc)

⇒ accuracy score:  0.885542719143659
f1 score:  0.938908320675576
ROC-AUC score:  0.7460650863364604

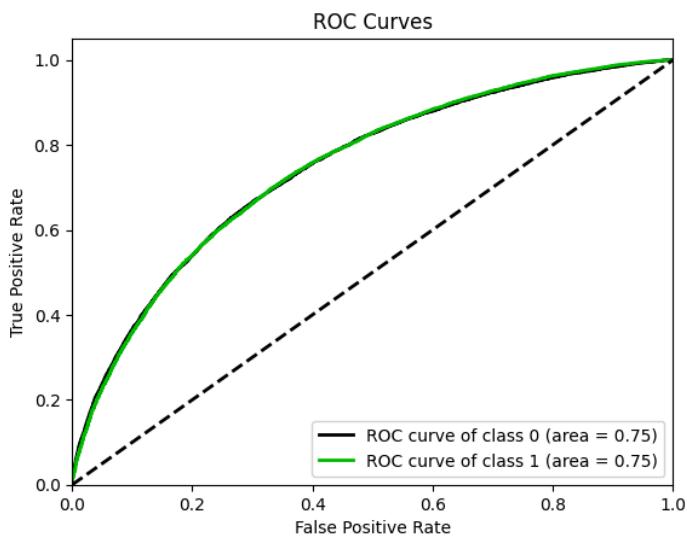
```

Gambar 33. Code dan output skor metrik *AdaBoostClassifier*

Didapat skor akurasi, skor f1, dan skor ROC-AUC dari model *AdaBoost Classifier* pertama



Gambar 34. Confusion Matrices *AdaBoostClassifier* Pertama



gambar35. Kurva ROC *AdaBoostClassifier* pertama

Didapat juga hasil *confusion matrix* (Gambar 34) dari model *AdaBoostClassifier* pertama dan plot kurva ROC (Gambar 35).

## F. AdaBoost dengan Hyperparameter Tuning

Diimplementasikan Grid Search menggunakan *GridSearchCV* untuk mencari kombinasi hyperparameter terbaik dari model *AdaBoostClassifier*.

```
47m [96] adaboost3 = AdaBoostClassifier(dectree, random_state=SEED)
parameters = {
    'n_estimators' : [100, 200, 300, 400, 500],
    'learning_rate' : [0.01, 0.1, 0.5, 1]
}

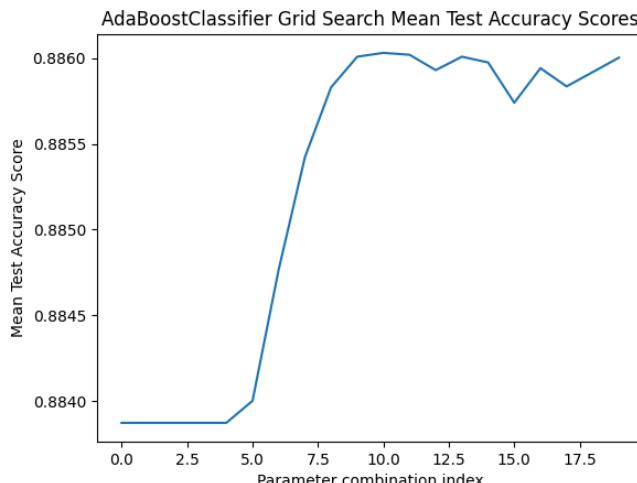
grid_search = GridSearchCV(adaboost3, parameters, n_jobs = -1, cv = 3, verbose = 1, scoring='accuracy')
grid_search.fit(X_train, y_train.values.ravel())

Fitting 3 folds for each of 20 candidates, totalling 60 fits
> GridSearchCV
> estimator: AdaBoostClassifier
> estimator: DecisionTreeClassifier
> DecisionTreeClassifier
```

Gambar 36. Algoritma AdaBoost

Bagian ini mendefinisikan parameter grid yang akan dicari oleh *GridSearchCV*. Dua parameter utama dari model *AdaBoostClassifier* yang dipertimbangkan adalah:

- *n\_estimators*: Jumlah estimator yang digunakan dalam boosting. Nilai yang diuji adalah 100, 200, 300, 400, dan 500.
- *learning\_rate*: Faktor yang mengatur kontribusi masing-masing estimator. Nilai yang diuji adalah 0.01, 0.1, 0.5, dan 1.



Gambar 37. Hasil skor akurasi *GridSearchCV AdaBoost*

Bagian ini membuat plot garis dari *mean\_test\_score*, yaitu rata-rata skor akurasi dari setiap kombinasi hyperparameter yang diuji selama *Grid Search*. Hasil rata-rata skor akurasi seluruh eksperimen

```
▶ print(grid_search.best_params_)

→ {'learning_rate': 0.5, 'n_estimators': 100}
```

Gambar 38. Parameter Terbaik AdaBoost

Kode dan hasil yang diberikan menunjukkan cara untuk menampilkan hyperparameter terbaik yang ditemukan oleh *GridSearchCV* setelah proses pencarian selesai.

```

[100] adaboost_3_hypetune = AdaBoostClassifier(
        dectree, **grid_search.best_params_, random_state=SEED
    )

    adaboost_3_hypetune.fit(X_train, y_train)

→      ▶ AdaBoostClassifier
      ▶ estimator: DecisionTreeClassifier
          ▶ DecisionTreeClassifier

```

Gambar 39. Model *AdaBoostClassifier* dengan *Hyperparameter Tuning*

Menggunakan hyperparameter terbaik yang ditemukan oleh *GridSearchCV* untuk melatih model *AdaBoostClassifier*

```

[101] y_adaboost_3_hypetune = adaboost_3_hypetune.predict(X_val)
    acc = accuracy_score(y_val, y_adaboost_3_hypetune)
    f1 = f1_score(y_val, y_adaboost_3_hypetune)
    roc_auc = roc_auc_score(y_val, adaboost_3_hypetune.predict_proba(X_val)[:, 1])

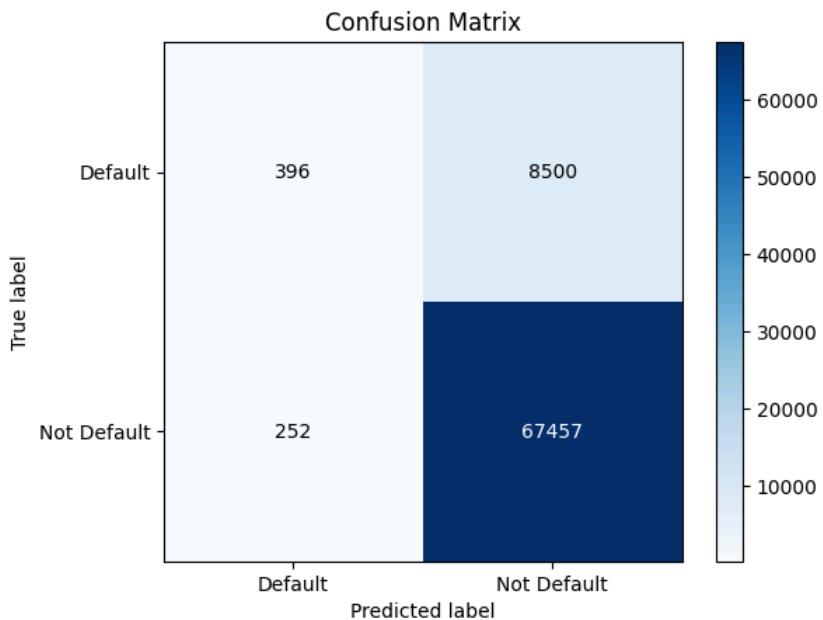
    print("accuracy score: ", acc)
    print("f1 score: ", f1)
    print("ROC-AUC score: ", roc_auc)

→  accuracy score:  0.8857515827948568
    f1 score:  0.9390809238093912
    ROC-AUC score:  0.749050761698311

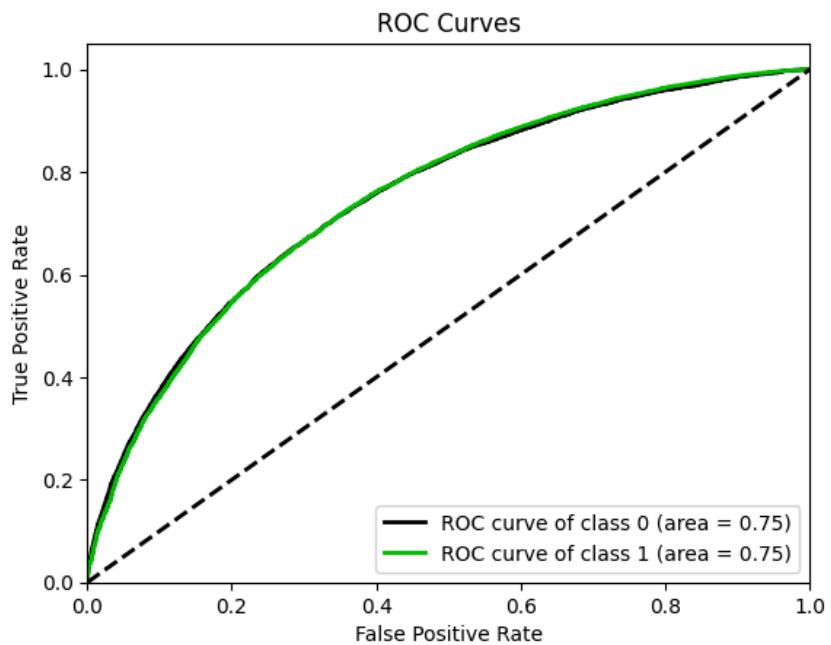
```

Gambar 40. Code dan output skor metrik *AdaBoostClassifier Hyperparameter Tuning*

Didapat skor akurasi, skor f1, dan skor ROC-AUC dari model *AdaBoost Classifier* dengan parameter terbaik dari *Hyperparameter Tuning*



Gambar 41. Confusion Matrix *AdaBoostClassifier Hyperparameter Tuning*



Gambar 42. Kurva ROC *AdaBoostClassifier Hyperparameter Tuning*

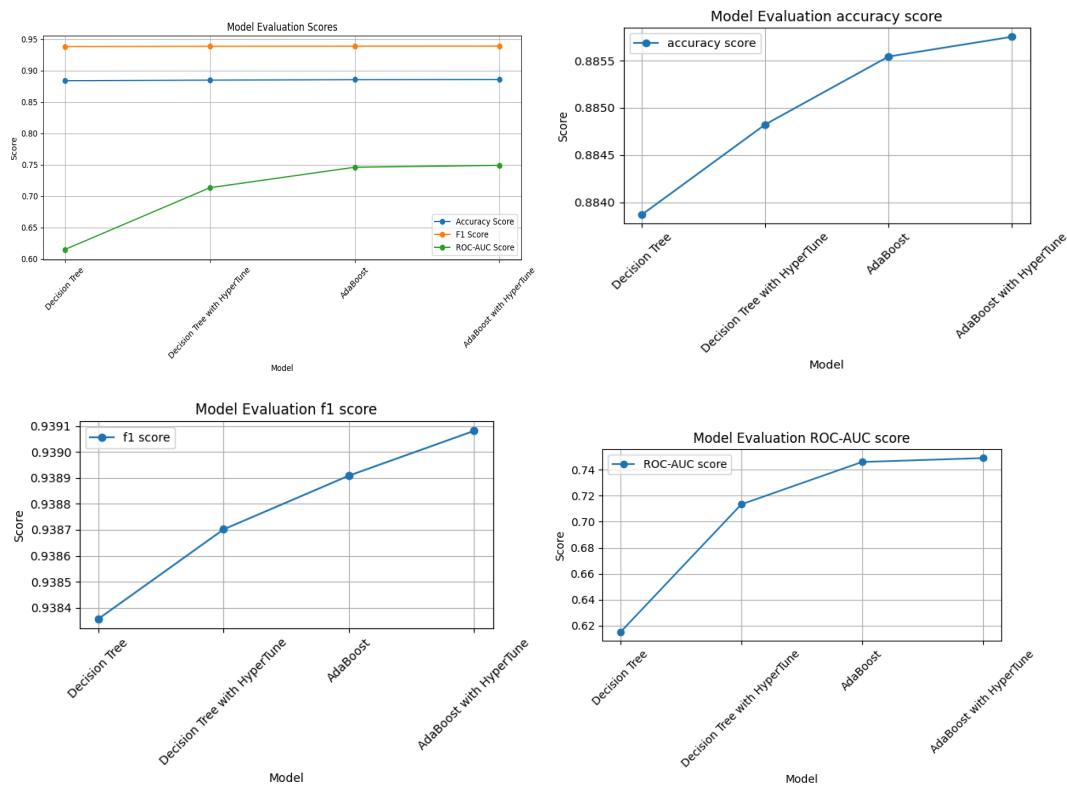
Didapat juga hasil *confusion matrix* (Gambar 41) dari model *AdaBoostClassifier* dengan parameter terbaik hasil *Hyperparameter tuning* dan plot kurva ROC (Gambar 42).

## BAB IV

### KESIMPULAN

#### A. Kesimpulan

Dengan program yang telah kami buat, kami mendapatkan bahwa model *AdaBoost* dengan *hyperparameter tuning* menunjukkan performa terbaik dalam analisis prediksi gagal bayar pinjaman (*default*) ini. Hal ini dibuktikan melalui skor tertinggi pada matrik evaluasi yang digunakan, yaitu *accuracy*, *F1-score*, dan ROC-AUC. Skor *accuracy* yang tinggi menunjukkan bahwa model ini mampu memprediksi dengan benar sebagian besar data yang diuji. *F1-score* yang tinggi mengindikasikan keseimbangan yang baik antara presisi dan *recall*, memastikan bahwa model tidak hanya akurat tetapi juga efektif dalam menangani ketidakseimbangan kelas. Selain itu, skor ROC-AUC yang tinggi menunjukkan kemampuan model dalam membedakan antara kelas positif dan negatif secara konsisten. Dengan kata lain, model *AdaBoost* yang telah dioptimalkan dengan *hyperparameter tuning* ini memberikan kinerja yang unggul dibandingkan dengan model *decision tree*.



## **DAFTAR PUSTAKA**

Otoritas Jasa Keuangan. (2024). *Financial technology - P2P Landing*. Jakarta. Diakses dari <https://ojk.go.id/id/kanal/iknb/financial-technology/default.aspx>

GeeksforGeeks. (2024). ID3 Algorithm - Sklearn Iterative Dichotomiser 3. Diakses dari <https://www.geeksforgeeks.org/sklearn-iterative-dichotomiser-3-id3-algorithms>

Anggraeni, Rina. (2024). Benarkah di Luar Negeri Juga Ada Pinjol seperti Indonesia, Ini Faktanya. Diakses dari <https://economy.okezone.com/read/2024/05/30/622/3014991/benarkah-di-luar-negeri-juga-ad-a-pinjol-seperti-indonesia-ini-faktanya?page=2>

Pramod, O. (2023). Decision Trees. Diakses dari <https://medium.com/@ompramod9921/decision-trees-8e2391f93fa7>

Agrawal, S. K. (2024). Metrics to Evaluate Your Classification Model to Take the Right Decisions. Diakses dari <https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/>

Bhandari, Aniruddha. (2024). Understanding AUC - ROC Curve. Diakses dari <https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/>