Understanding the Factors Influencing Customer Bank Churn: A Machine Learning Approach use Decision Tree and SVM Algorithm

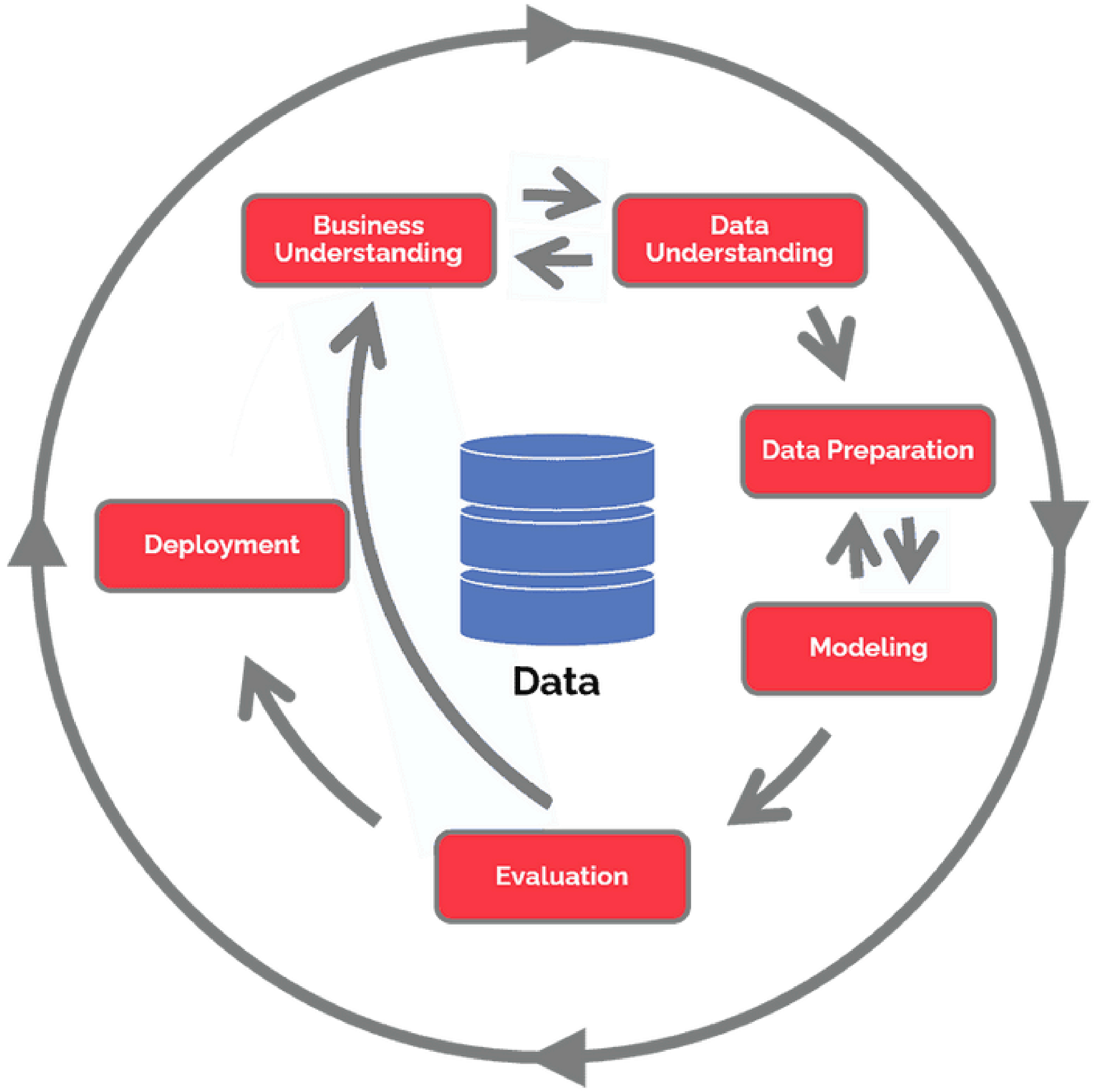
Anggota Kelompok

Dwi Yulianto (00000074859)

Nicholas Soesilo (0000006649)

Marcello Roy (00000072593)

CRISP DM (Cross Industry Standard Process for Data Mining)



Business Understanding

Ini merupakan fase untuk memahami secara mendalam tujuan bisnis yang ingin dicapai dan masalah yang ingin dipecahkan.

Data Understanding

di fase ini terfokus pada pemahaman awal terhadap data yang ada. Ini mencakup eksplorasi data dan identifikasi karakteristik penting.

Data Preparation

Di proses ini dilakukan pembersihan, transformasi, dan pengorganisasian data agar dapat digunakan untuk analisis selanjutnya saat modeling.

Modeling

Di fase ini penerapan model machine learning untuk menganalisis data supaya bisa mencapai tujuan bisnis.

Evaluation

Evaluasi kinerja model yang telah dilakukan menggunakan metrik tertentu.

Deployment

Tahap di mana model yang telah dibuat diimplementasikan dalam lingkungan bisnis.

Business Understanding

| Customer Churn Dalam Industri Bank

Customer churn atau customer attrition adalah istilah yang lebih sering digunakan untuk menggambarkan kehilangan pelanggan atau nasabah dari suatu layanan, termasuk di sektor perbankan.

Customer churn dalam industri perbankan merujuk pada pelanggan yang memutuskan untuk berhenti menggunakan produk atau layanan perbankan dari suatu bank. Faktor pelanggan meninggalkan bank bisa bermacam-macam.

Analisis "customer churn" pada bank sebagai suatu metode atau analisis yang digunakan oleh bank untuk mengidentifikasi dan mengurangi jumlah pelanggan yang meninggalkan layanan bank, maka ini bisa mencakup strategi retensi pelanggan, analisis data untuk mengidentifikasi pola churn.

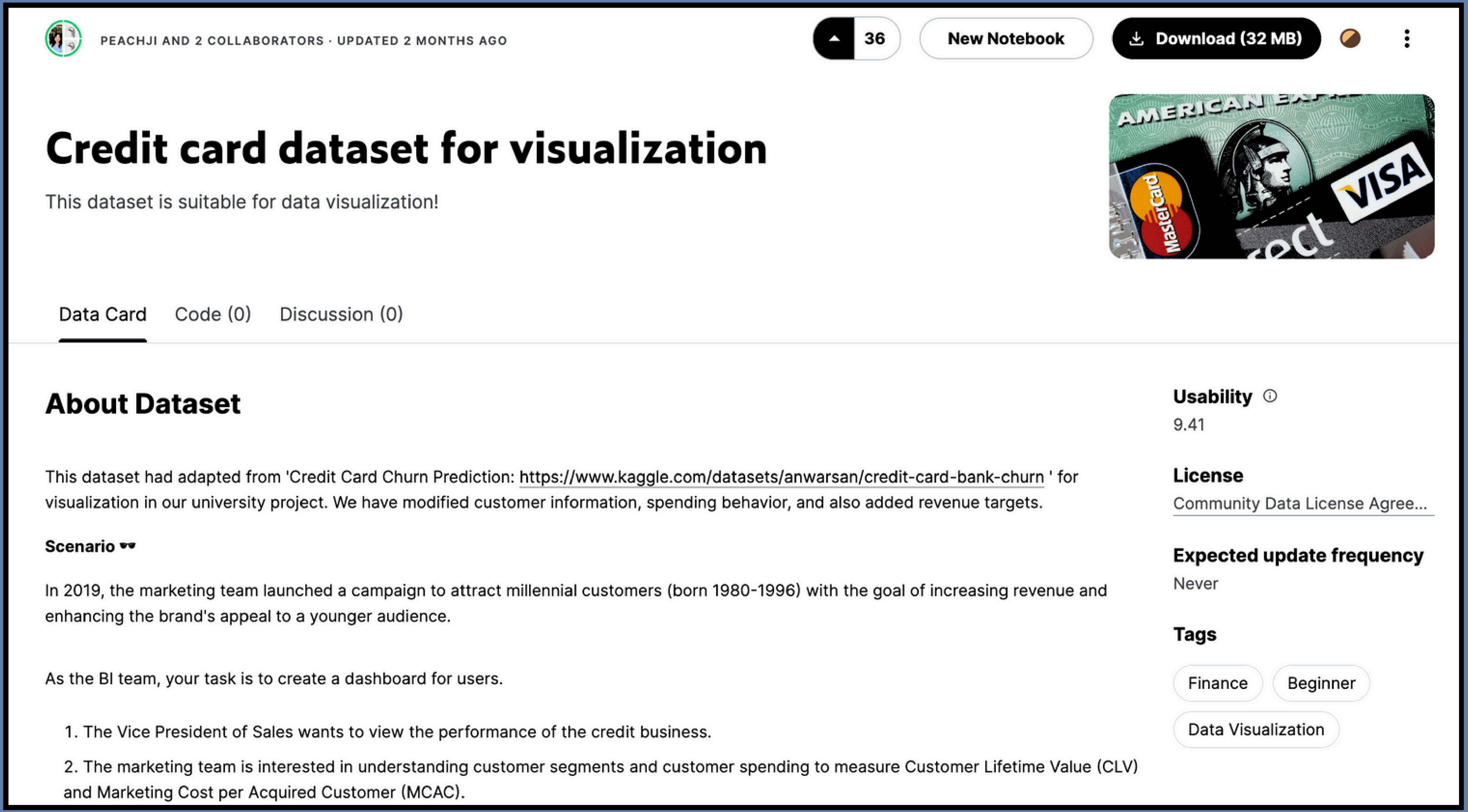
Case Study Objectives

Dalam project ini sebuah bank ingin memahami apa faktor-faktor pelanggan meninggalkan layanan bank, dengan memahami faktor-faktor yang dicari harapannya perusahaan bank bisa menyusun strategi untuk mengurangi tingkat churn dengan berfokus pada faktor-faktor yang mempengaruhi churn ini sehingga pendapatan bank bisa terus meningkat.

Data Understanding

| Pemahaman Dataset

Dalam studi kasus ini, dalam melakukan analisa kami menggunakan suatu dataset yang kami dapatkan dari platform Kaggle. Dataset tersebut berjudul : “Credit Card for Data Visualization”, dataset ini mengandung beberapa informasi mengenai pelanggan dan juga record transaksi di suatu bank.



| Pemahaman Dataset

Data Frame 1

```python
credit_info = pd.read_excel('credit_2018_2019.xlsx', sheet_name='info_all')
credit_finance = pd.read_excel('credit_2018_2019.xlsx', sheet_name='Finance_all')
```

```python
credit_info
```

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_b
0	712672083	Existing Customer	65	F	0	High School	Married	Less than $40K	Blue	
1	713049933	Existing Customer	47	F	1	Graduate	Married	$40K-60K$	Silver	
2	713049933	Existing Customer	48	F	1	Graduate	Married	$40K-60K$	Silver	
3	713135883	Existing Customer	65	F	0	College	Married	Less than $40K	Blue	
4	713135883	Existing Customer	64	F	0	College	Married	Less than $40K	Blue	
...	...	...	...	...	...	...	...	...	...	
20066	900203139	Existing Customer	24	F	0	College	Single	Less than $40K	Silver	
20067	900203141	Existing Customer	24	F	1	College	Divorced	Less than $40K	Silver	
20068	900203144	Existing Customer	24	M	2	College	Married	$60K-80K$	Blue	
20069	900203145	Existing Customer	24	M	0	College	Single	$40K-60K$	Blue	
20070	900203146	Existing Customer	24	M	1	College	Single	Less than $40K	Blue	

20071 rows × 21 columns

Informasi mengenai nasabah Bank.

Data Frame 2

```python
credit_finance
```

	CLIENTNUM	Attrition_Flag	Year	Quarter	Date	Type	Trans_Amount	Revenue
0	708082083	Existing Customer	2018	Q1	Q1,2018	Shop	120.1200	2.40240
1	708083283	Attrited Customer	2018	Q1	Q1,2018	Shop	135.5550	2.71110
2	708084558	Attrited Customer	2018	Q1	Q1,2018	Shop	396.2700	7.92540
3	708085458	Existing Customer	2018	Q1	Q1,2018	Shop	122.9550	2.45910
4	708086958	Existing Customer	2018	Q1	Q1,2018	Shop	134.6400	2.69280
...	...	...	...	...	...	...	...	...
306871	721164483	Existing Customer	2019	Q4	Q4,2019	Cash	56.5440	2.82720
306872	708095133	Existing Customer	2019	Q4	Q4,2019	Cash	238.9592	11.94796
306873	900202780	Existing Customer	2019	Q4	Q4,2019	Cash	13.6496	0.68248
306874	779770683	Existing Customer	2019	Q4	Q4,2019	Cash	60.1008	3.00504
306875	709632483	Existing Customer	2019	Q4	Q4,2019	Cash	2.0672	0.10336

306876 rows × 8 columns

Informasi mengenai transaksi nasabah terkait di Bank.

Daftar Kolom pada Dataset

1	CLIENTNUM	Nomor identifikasi unik untuk setiap pelanggan
2	Attrition_Flag	Menunjukkan informasi apakah pelanggan sudah keluar (churn) atau masih menjadi pelanggan
3	Customer_Age	Menunjukkan usia pelanggan
4	Gender	Menunjukkan jenis kelamin pelanggan
5	Dependent_count	Menunjukkan Jumlah anggota tanggungan pelanggan
6	Education_Level	Menunjukkan Tingkat pendidikan pelanggan
7	Marital_Status	Menunjukkan Status pernikahan pelanggan
8	Income_Category	Menunjukkan Kategori pendapatan pelanggan
9	Card_Category	Menunjukkan Kategori kartu kredit yang dimiliki pelanggan
10	Months_on_book	Menunjukkan Jumlah bulan pelanggan telah menjadi nasabah bank

Daftar Kolom pada Dataset

11	Total_Relationship_C oun	Menunjukkan Jumlah produk atau layanan perbankan yang dimiliki oleh pelanggan
12	Credit_Limit	Menunjukkan Batas kredit pelanggan
13	Quarter, Year, Date_Leave	Menunjukkan Informasi temporal yang mungkin mencerminkan waktu ketika pelanggan keluar.
14	Months_Inactive	Menunjukkan jumlah bulan ketika pelanggan tidak melakukan transaksi atau tidak aktif
15	Contacts_Count	Menunjukkan Jumlah kontak atau interaksi yang dilakukan oleh bank dengan pelanggan selama periode tertentu
16	Total_Revolving_Bal	Menunjukkan Jumlah saldo yang masih ada pada akun pelanggan setelah pembayaran minimum dilakukan pada kartu kredit.
17	Total_Trans_Ct	Menunjukkan Jumlah total transaksi yang dilakukan oleh pelanggan
18	Avg_Open_To_Buy	Menunjukkan Rata-rata jumlah kredit yang masih tersedia untuk digunakan oleh pelanggan
19	Avg_Utilization_Ratio	Menunjukkan Rata-rata rasio penggunaan kredit oleh pelanggan
20	Trans_Amount	Menunjukkan Jumlah total uang yang dikeluarkan atau dipindahkan oleh pelanggan
21	Revenue	Menunjukkan pendapatan yang dihasilkan oleh pelanggan untuk bank selama periode tertentu

Data Understanding for credit_info

Deskripsi DataFrame pertama

```
credit_info.describe()
```

	CLIENTNUM	Customer_Age	Dependent_count	Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon	Contacts_Count_12_mon	Credit_
count	2.007100e+04	20071.000000	20071.000000	20071.000000	20071.000000	20071.000000	20071.000000	20071.00
mean	7.510799e+08	46.331872	1.954810	41.039510	3.816701	2.665388	2.449106	8637.12
std	5.472039e+07	8.385922	1.201685	10.392045	1.553507	1.612679	1.104183	9084.35
min	7.080821e+08	24.000000	0.000000	13.000000	1.000000	0.000000	0.000000	1400.00
25%	7.134855e+08	41.000000	1.000000	35.000000	3.000000	1.000000	2.000000	2548.50
50%	7.188436e+08	47.000000	2.000000	41.000000	4.000000	3.000000	2.000000	4532.00
75%	7.810140e+08	52.000000	3.000000	48.000000	5.000000	3.000000	3.000000	11062.00
max	9.002031e+08	65.000000	5.000000	68.000000	6.000000	6.000000	6.000000	35000.00

Daftar Kolom dan tipe data

```
credit_info.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20071 entries, 0 to 20070
Data columns (total 21 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   CLIENTNUM                 20071 non-null  int64  
 1   Attrition_Flag            20071 non-null  object 
 2   Customer_Age              20071 non-null  int64  
 3   Gender                    20071 non-null  object 
 4   Dependent_count           20071 non-null  int64  
 5   Education_Level           20071 non-null  object 
 6   Marital_Status            20071 non-null  object 
 7   Income_Category           20071 non-null  object 
 8   Card_Category             20071 non-null  object 
 9   Months_on_book            20071 non-null  int64  
 10  Total_Relationship_Count  20071 non-null  int64  
 11  Months_Inactive_12_mon    20071 non-null  int64  
 12  Contacts_Count_12_mon     20071 non-null  int64  
 13  Credit_Limit              20071 non-null  float64
 14  Total_Revolving_Bal       20071 non-null  int64  
 15  Avg_Open_To_Buy           20071 non-null  float64
 16  Total_Trans_Ct            20071 non-null  int64  
 17  Avg_Utilization_Ratio     20071 non-null  float64
 18  Quarter                   20071 non-null  object 
 19  Year                      20071 non-null  int64  
 20  Date_Leave                20071 non-null  object 
dtypes: float64(3), int64(10), object(8)
memory usage: 3.2+ MB
```

Cek Missing Data

```
missing_values = credit_info.isnull().sum()
percentage_missing = (missing_values / len(credit_info)) * 100
missing_data = pd.DataFrame({'Missing Values': missing_values, 'Percentage': percentage_missing})
print(missing_data)
```

```
                          Missing Values  Percentage
CLIENTNUM                              0         0.0
Attrition_Flag                         0         0.0
Customer_Age                           0         0.0
Gender                                 0         0.0
Dependent_count                        0         0.0
Education_Level                        0         0.0
Marital_Status                         0         0.0
Income_Category                        0         0.0
Card_Category                          0         0.0
Months_on_book                         0         0.0
Total_Relationship_Count               0         0.0
Months_Inactive_12_mon                 0         0.0
Contacts_Count_12_mon                  0         0.0
Credit_Limit                           0         0.0
Total_Revolving_Bal                    0         0.0
Avg_Open_To_Buy                        0         0.0
Total_Trans_Ct                         0         0.0
Avg_Utilization_Ratio                  0         0.0
Quarter                                0         0.0
Year                                   0         0.0
Date_Leave                             0         0.0
```

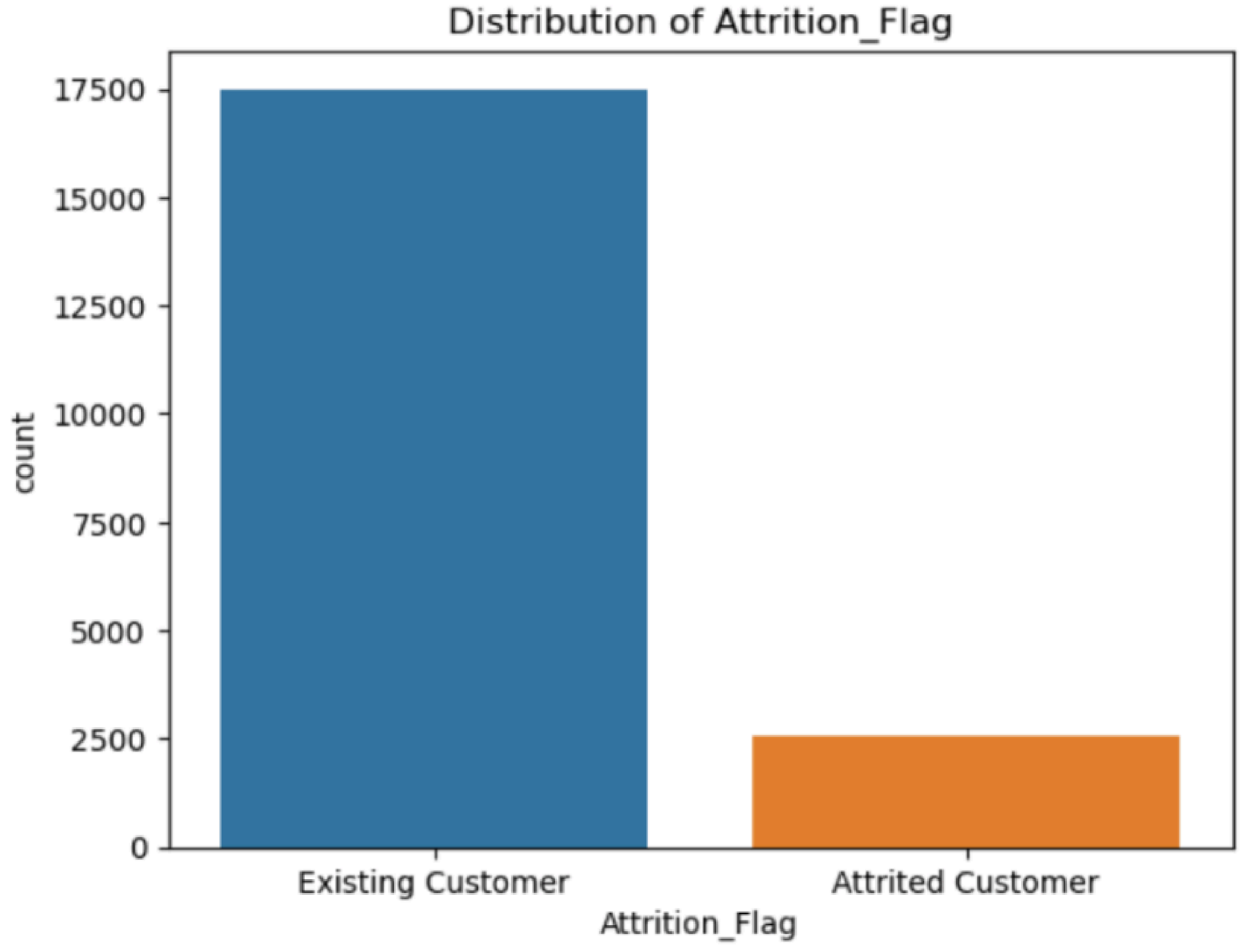
Cek Nilai Unik Setiap Kolom

```
unique_counts = credit_info.nunique()
unique_counts
```

```
CLIENTNUM                   11571
Attrition_Flag                  2
Customer_Age                   42
Gender                          2
Dependent_count                 6
Education_Level                 6
Marital_Status                  3
Income_Category                 5
Card_Category                   4
Months_on_book                 56
Total_Relationship_Count        6
Months_Inactive_12_mon          7
Contacts_Count_12_mon           7
Credit_Limit                 6219
Total_Revolving_Bal          2232
Avg_Open_To_Buy              6814
Total_Trans_Ct                126
Avg_Utilization_Ratio         965
Quarter                         5
Year                            2
Date_Leave                     10
dtype: int64
```

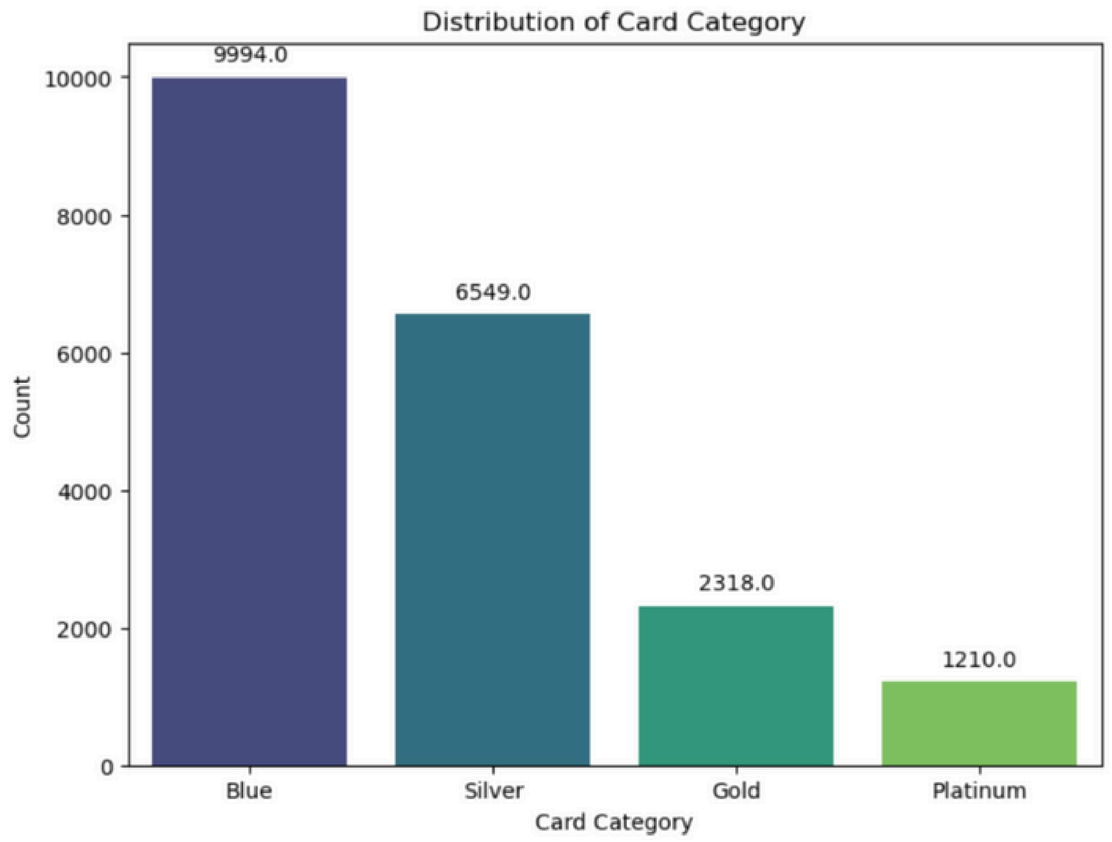
Distribusi Kolom Pelanggan yang churn

```
sns.countplot(x='Attrition_Flag', data=credit_info)
plt.title('Distribution of Attrition_Flag')
plt.show()
```



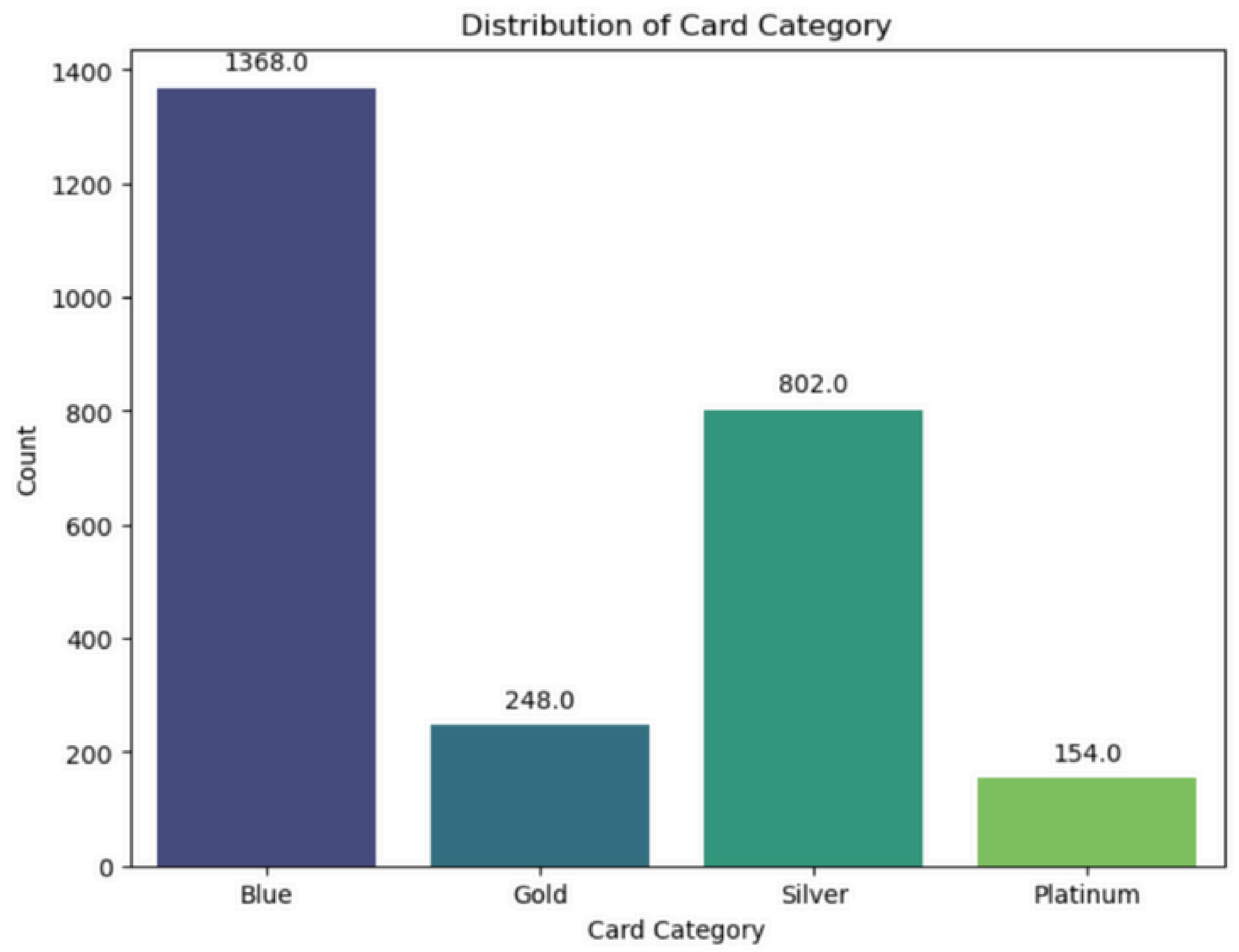
Distribusi Card Category

```
# Membuat bar chart untuk distribusi Card_Category
plt.figure(figsize=(8, 6))
ax = sns.countplot(x='Card_Category', data=credit_info, palette='viridis')

# Menambahkan keterangan jumlah per kategori kartu
for p in ax.patches:
    ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center', xytext=(0, 10), textcoords='offset points')

plt.title('Distribution of Card Category')
plt.xlabel('Card Category')
plt.ylabel('Count')
plt.show()
```



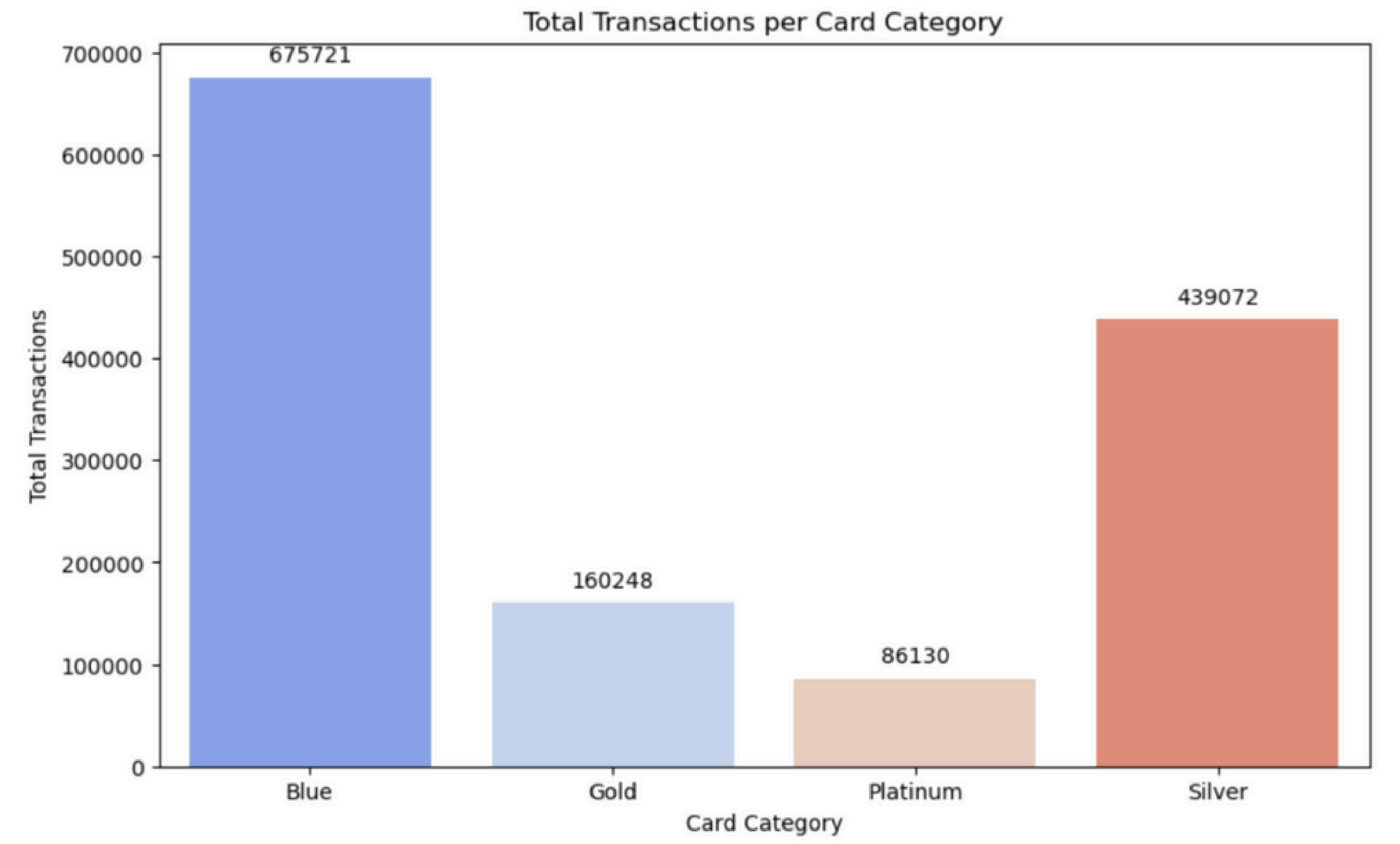
Distribusi Card pada Customer yang churn

```
churn_data = credit_info[credit_info['Attrition_Flag'] == 'Attrited Customer']
# Membuat bar chart untuk distribusi Card_Category
plt.figure(figsize=(8, 6))
ax = sns.countplot(x='Card_Category', data=churn_data, palette='viridis')

# Menambahkan keterangan jumlah per kategori kartu
for p in ax.patches:
    ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center', xytext=(0, 10), textcoords='offset points')

plt.title('Distribution of Card Category')
plt.xlabel('Card Category')
plt.ylabel('Count')
plt.show()
```



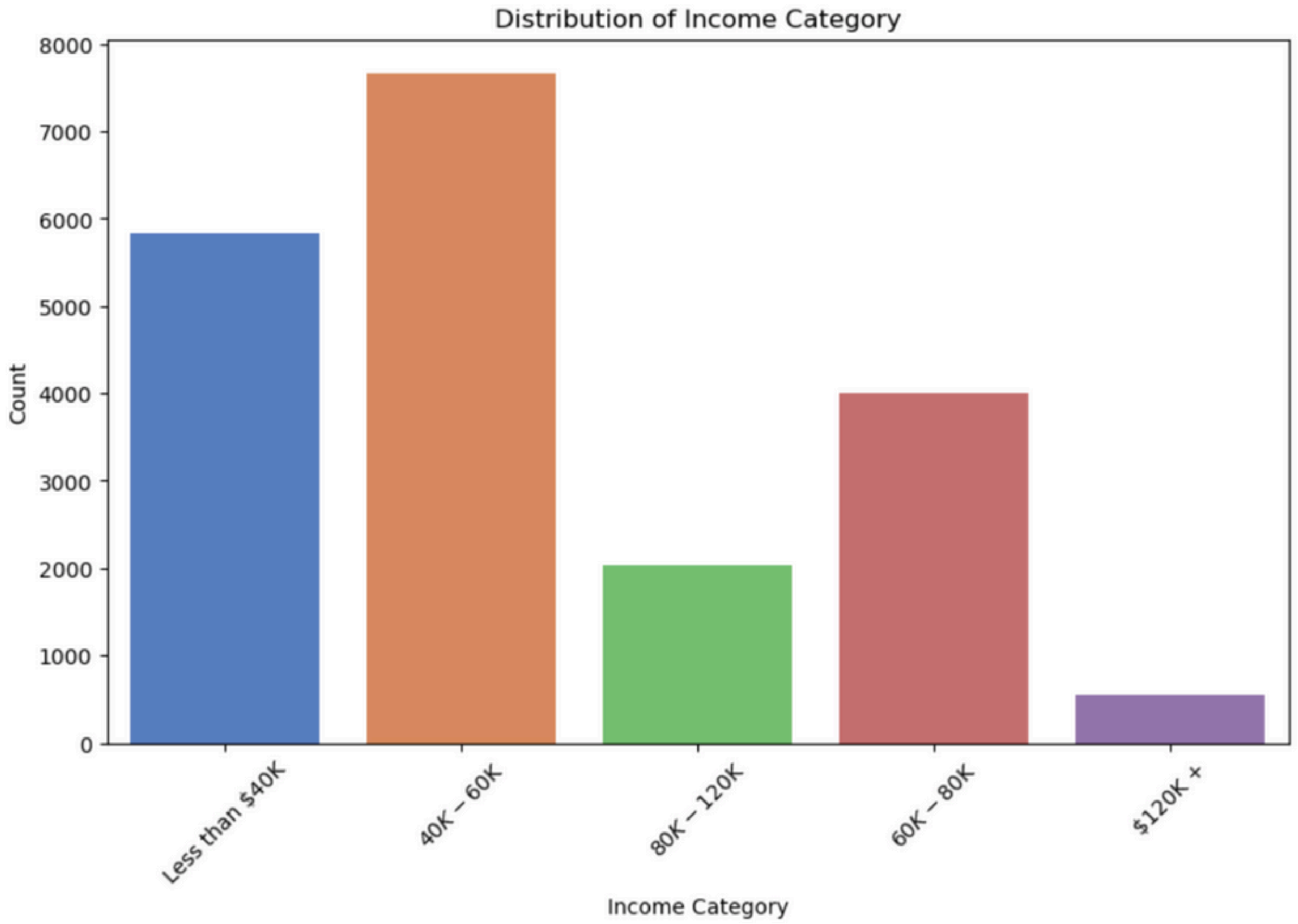
Total Transaksi berdasarkan per Jenis Kartu

```
# Menghitung total transaksi per jenis Card_Category
total_transactions = credit_info.groupby('Card_Category')['Total_Trans_Ct'].sum().reset_index()

# Membuat bar chart untuk total transaksi per jenis Card_Category
plt.figure(figsize=(10, 6))
ax = sns.barplot(x='Card_Category', y='Total_Trans_Ct', data=total_transactions, palette='coolwarm')

# Menambahkan label keterangan jumlah di atas setiap batang
for p in ax.patches:
    ax.annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center', xytext=(0, 10), textcoords='offset points')

plt.title('Total Transactions per Card Category')
plt.xlabel('Card Category')
plt.ylabel('Total Transactions')

plt.show()
```



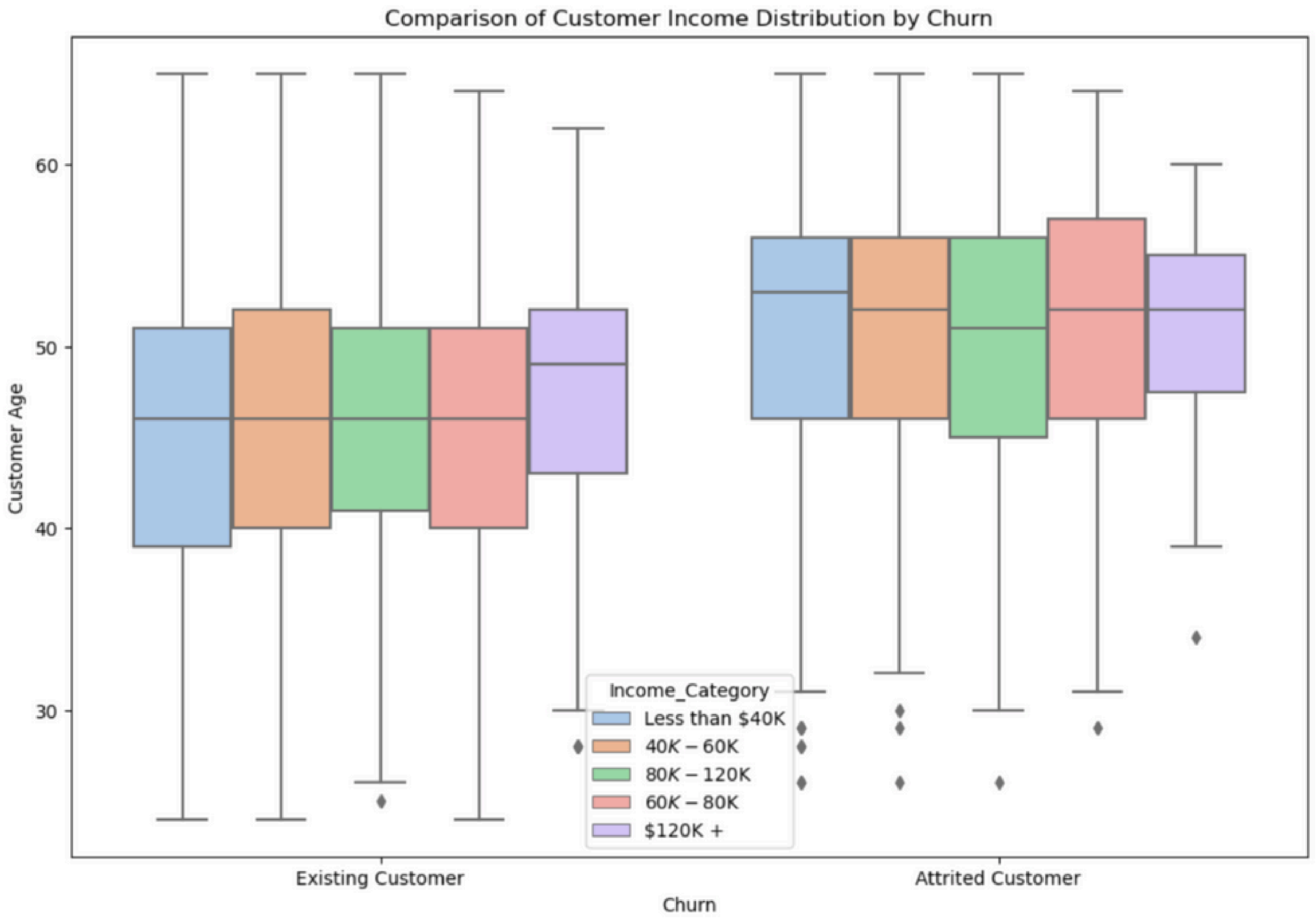
Distribusi Income Category

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Income_Category', data=credit_info, palette='muted')
plt.title('Distribution of Income Category')
plt.xlabel('Income Category')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



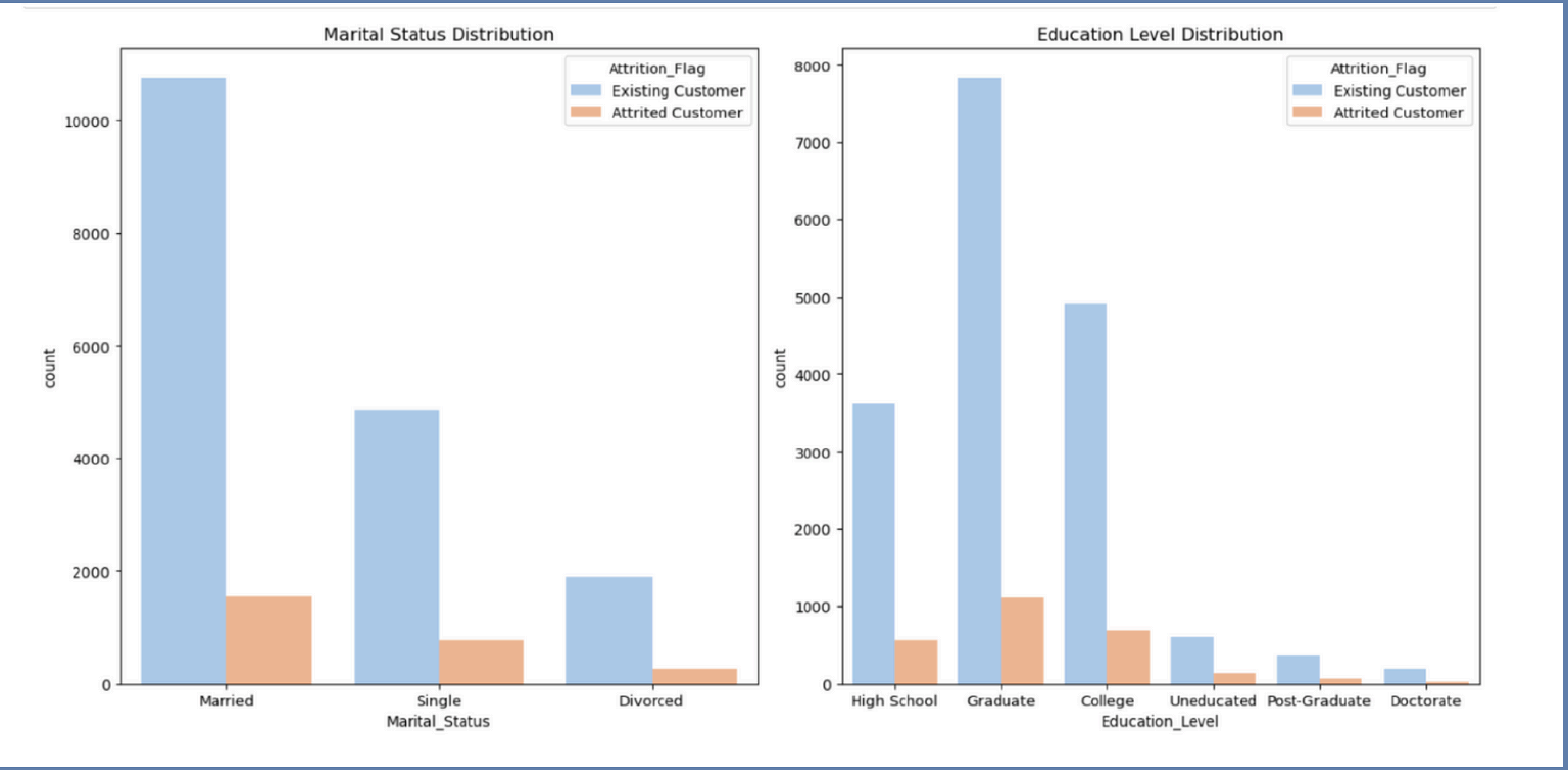
Box plot untuk membandingkan distribusi pendapatan pada setiap kategori churn

```
plt.figure(figsize=(12, 8))
sns.boxplot(x='Attrition_Flag', y='Customer_Age', data=credit_info, hue='Income_Category', palette='pastel')
plt.title('Comparison of Customer Income Distribution by Churn')
plt.xlabel('Churn')
plt.ylabel('Customer Age')
plt.show()
```



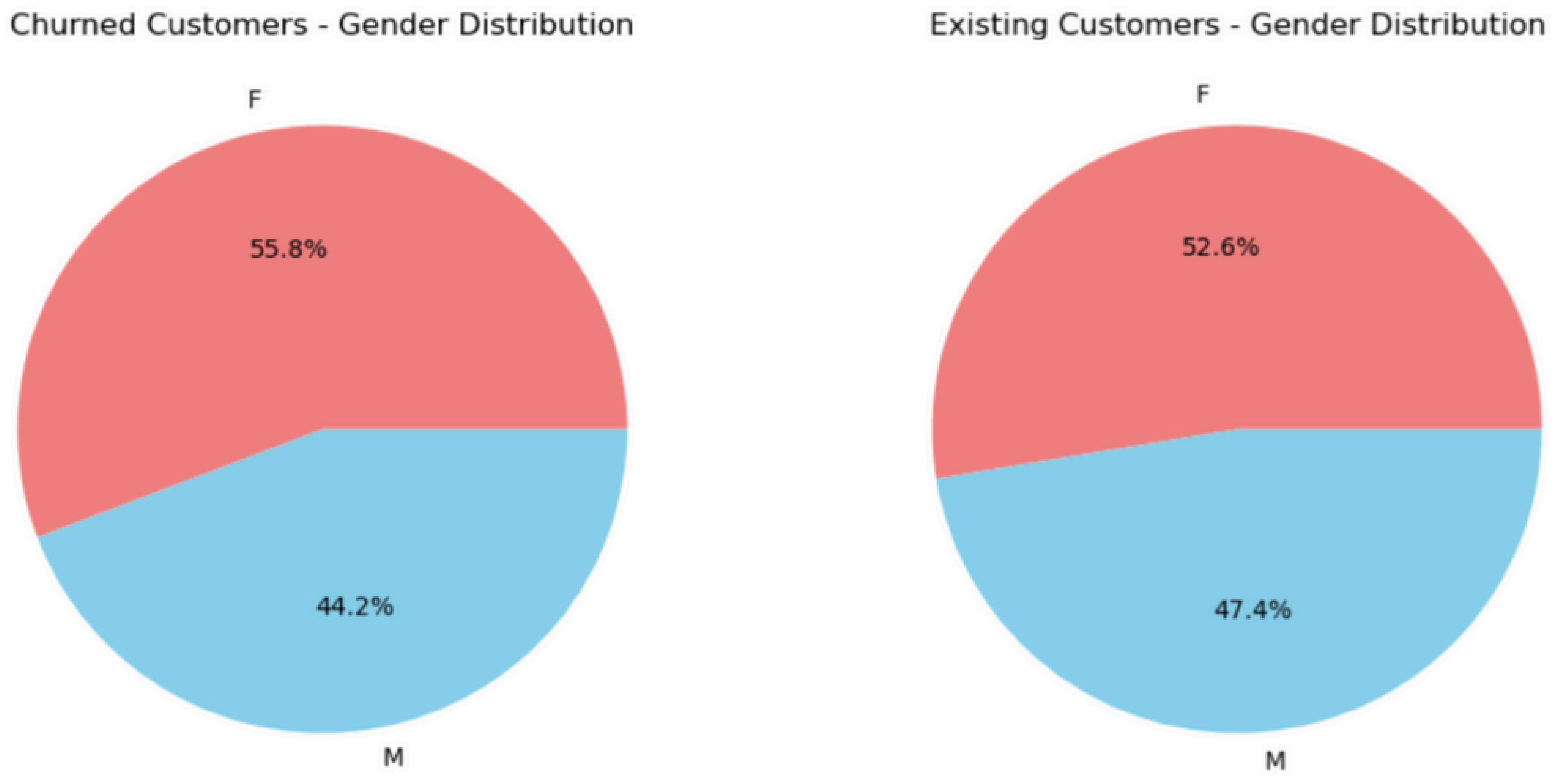
Distribusi berdasarkan Marial Status dan Education Level

```
# Filter data untuk pelanggan yang churn
churn_data = credit_info[credit_info['Attrition_Flag'] == 'Attrited Customer']

# Bar chart untuk Marital Status
plt.figure(figsize=(14, 7))
plt.subplot(1, 2, 1)
sns.countplot(x='Marital_Status', data=credit_info, hue='Attrition_Flag', palette='pastel')
plt.title('Marital Status Distribution')

# Bar chart untuk Education Level
plt.subplot(1, 2, 2)
sns.countplot(x='Education_Level', data=credit_info, hue='Attrition_Flag', palette='pastel')
plt.title('Education Level Distribution')

plt.tight_layout()
plt.show()
```



Visual Komposisi Customer berdasarkan Gender

```
churn_data = credit_info[credit_info['Attrition_Flag'] == 'Attrited Customer']
existing_data = credit_info[credit_info['Attrition_Flag'] == 'Existing Customer']

# Pie chart untuk Gender Distribution pada pelanggan yang Churn
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
churn_gender_distribution = churn_data['Gender'].value_counts()
plt.pie(churn_gender_distribution, labels=churn_gender_distribution.index,
        autopct='%1.1f%%', colors=['lightcoral', 'skyblue'])
plt.title('Churned Customers - Gender Distribution')

# Pie chart untuk Gender Distribution pada pelanggan yang Existing
plt.subplot(1, 2, 2)
existing_gender_distribution = existing_data['Gender'].value_counts()
plt.pie(existing_gender_distribution, labels=existing_gender_distribution.index,
        autopct='%1.1f%%', colors=['lightcoral', 'skyblue'])
plt.title('Existing Customers - Gender Distribution')

plt.show()
```



Data Understanding for credit_finance

Isi dataFrame

```
credit_finance
```

	CLIENTNUM	Attrition_Flag	Year	Quarter	Date	Type	Trans_Amount	Revenue
0	708082083	Existing Customer	2018	Q1	Q1,2018	Shop	120.1200	2.40240
1	708083283	Attrited Customer	2018	Q1	Q1,2018	Shop	135.5550	2.71110
2	708084558	Attrited Customer	2018	Q1	Q1,2018	Shop	396.2700	7.92540
3	708085458	Existing Customer	2018	Q1	Q1,2018	Shop	122.9550	2.45910
4	708086958	Existing Customer	2018	Q1	Q1,2018	Shop	134.6400	2.69280
...	...	...	...	...	...	...	...	...
306871	721164483	Existing Customer	2019	Q4	Q4,2019	Cash	56.5440	2.82720
306872	708095133	Existing Customer	2019	Q4	Q4,2019	Cash	238.9592	11.94796
306873	900202780	Existing Customer	2019	Q4	Q4,2019	Cash	13.6496	0.68248
306874	779770683	Existing Customer	2019	Q4	Q4,2019	Cash	60.1008	3.00504
306875	709632483	Existing Customer	2019	Q4	Q4,2019	Cash	2.0672	0.10336

306876 rows × 8 columns

Deskripsi data

```
credit_finance.describe()
```

	CLIENTNUM	Year	Trans_Amount	Revenue
count	3.068760e+05	306876.000000	306876.000000	306876.000000
mean	7.516599e+08	2018.499381	239.960680	6.815908
std	5.533922e+07	0.500000	336.611644	8.825891
min	7.080821e+08	2018.000000	0.000000	0.000000
25%	7.135329e+08	2018.000000	50.490000	1.688865
50%	7.189092e+08	2018.000000	125.467500	3.663747
75%	7.833819e+08	2019.000000	282.398525	8.652000
max	9.002031e+08	2019.000000	5373.900000	150.202500

Informasi nama kolom dan tipe data

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 306876 entries, 0 to 306875
Data columns (total 8 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   CLIENTNUM       306876 non-null  int64
 1   Attrition_Flag  306876 non-null  object
 2   Year            306876 non-null  int64
 3   Quarter         306876 non-null  object
 4   Date            306876 non-null  object
 5   Type            306876 non-null  object
 6   Trans_Amount    306876 non-null  float64
 7   Revenue         306876 non-null  float64
dtypes: float64(2), int64(2), object(4)
memory usage: 18.7+ MB
```

Cek missing value

```
missing_values = credit_finance.isnull().sum()
percentage_missing = (missing_values / len(credit_finance)) * 100
missing_data = pd.DataFrame({'Missing Values': missing_values, 'Percentage': percentage_missing})
print(missing_data)
```

```
                Missing Values  Percentage
CLIENTNUM                    0         0.0
Attrition_Flag               0         0.0
Year                         0         0.0
Quarter                      0         0.0
Date                         0         0.0
Type                         0         0.0
Trans_Amount                 0         0.0
Revenue                      0         0.0
```

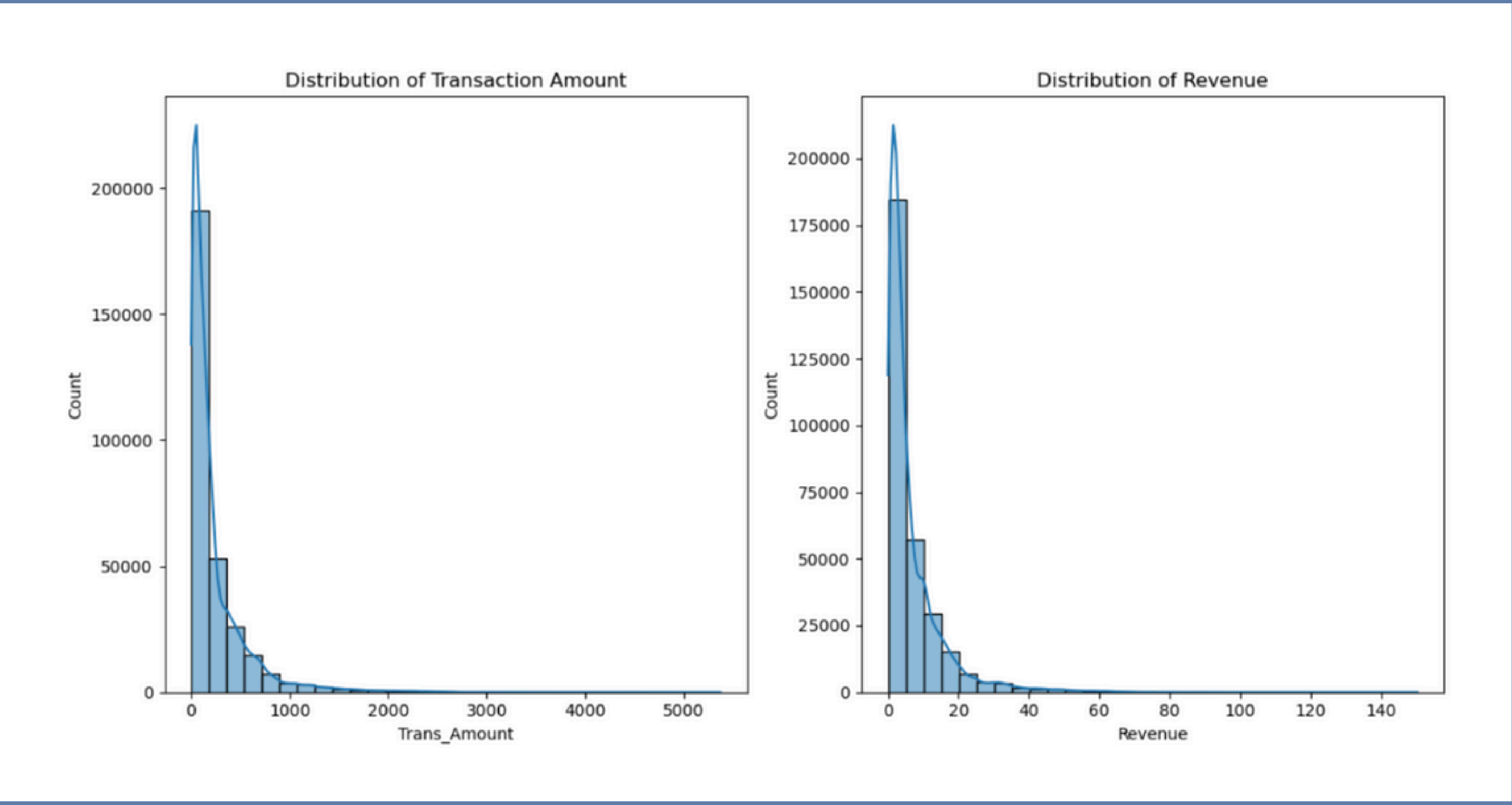
Cek Nilai unik pada setiap kolom

```
#cek nilai unik di dataframe
unique_counts = credit_finance.nunique()
print(unique_counts)
```

```
CLIENTNUM         11571
Attrition_Flag        2
Year                  2
Quarter               4
Date                  8
Type                  4
Trans_Amount     125931
Revenue          133015
dtype: int64
```

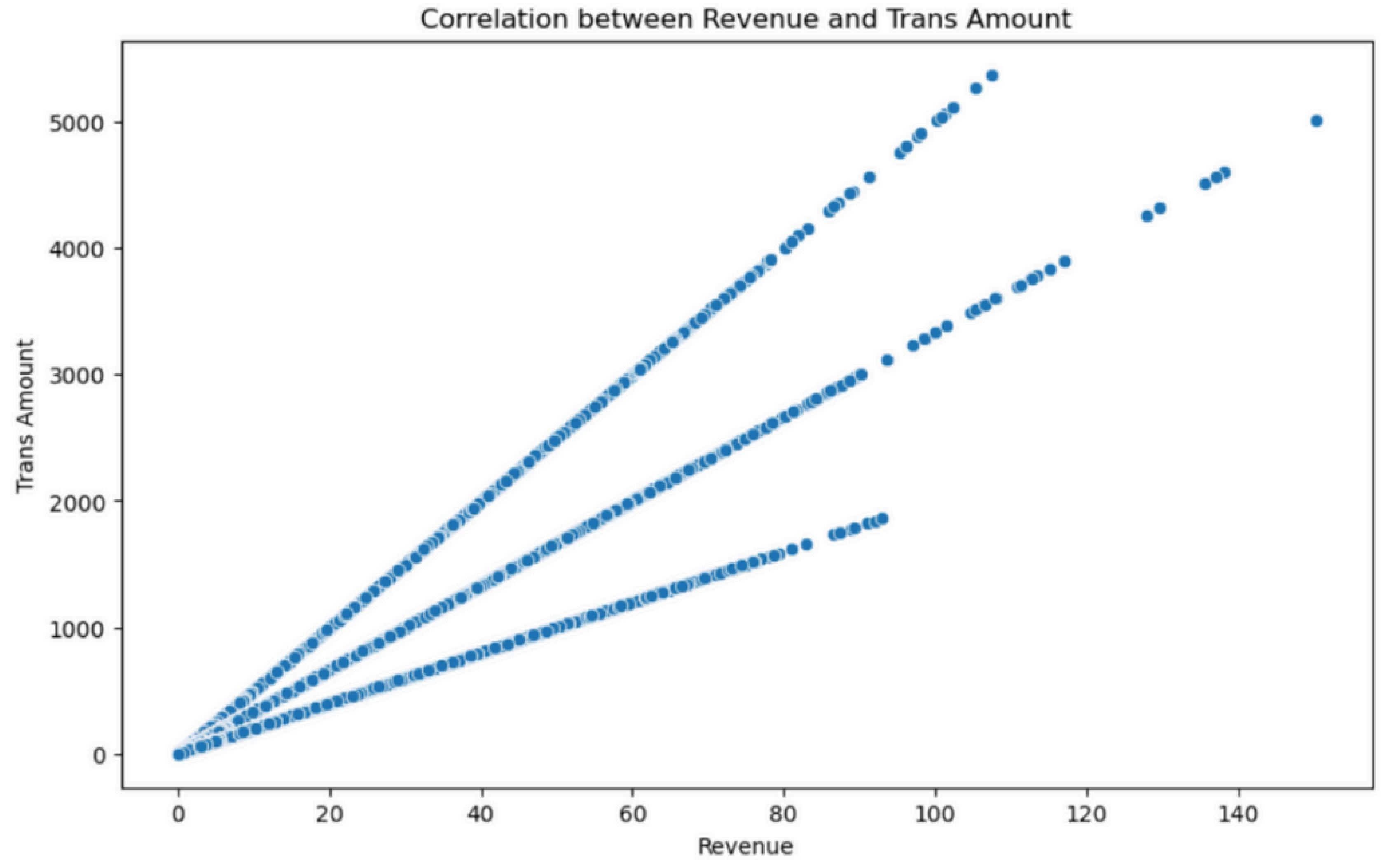
Menampilkan distribusi Trans Am

```
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
sns.histplot(credit_finance['Trans_Amount'], bins=30, kde=True)
plt.title('Distribution of Transaction Amount')

plt.subplot(1, 2, 2)
sns.histplot(credit_finance['Revenue'], bins=30, kde=True)
plt.title('Distribution of Revenue')

plt.tight_layout()
plt.show()
```

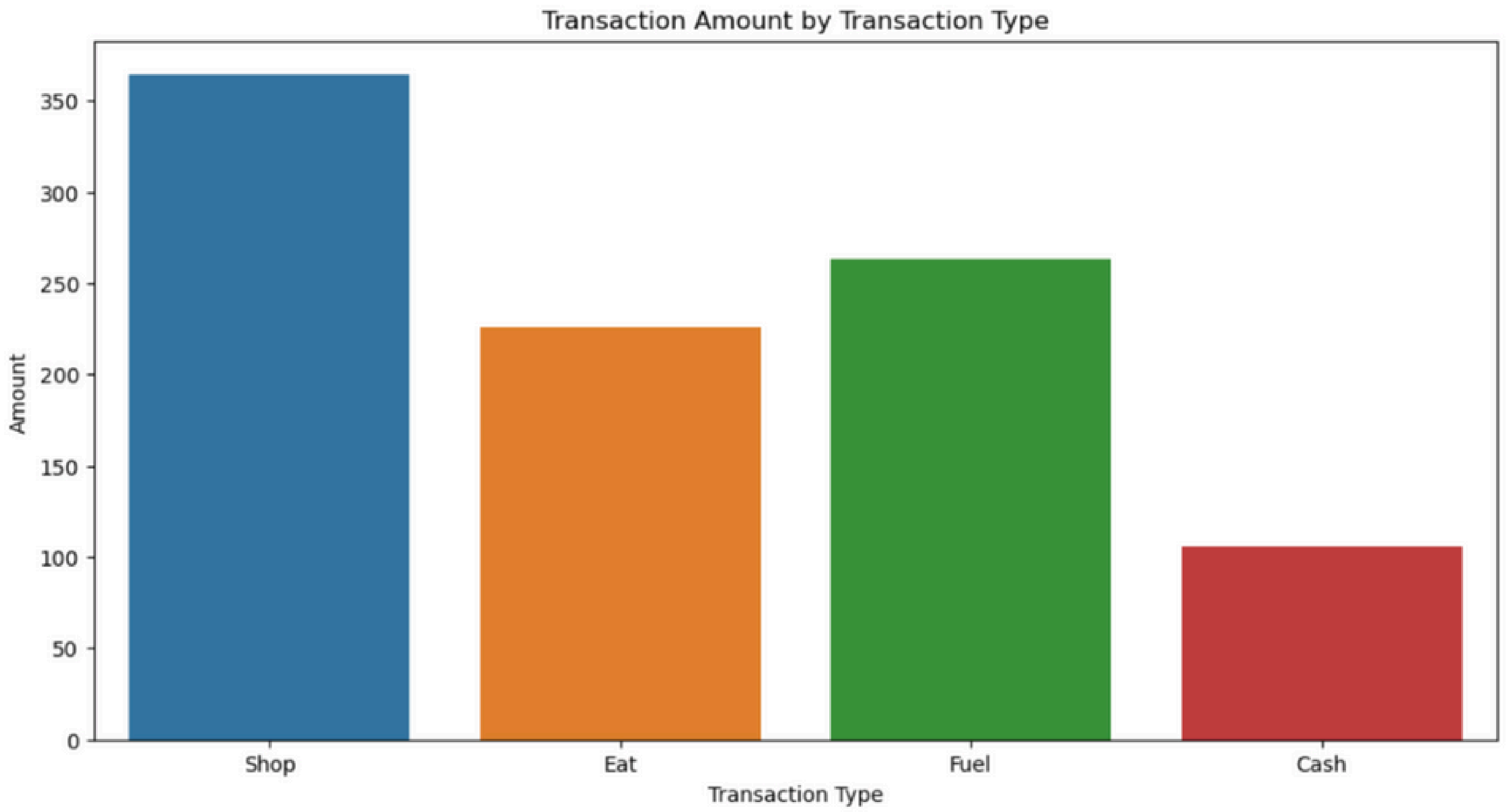


Korelasi antara Trans_Amount dengan Revenue

```python
# Scatter plot untuk korelasi antara Pendapatan dan Jumlah Transaksi
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Revenue', y='Trans_Amount', data=credit_finance, palette='coolwarm')
plt.title('Correlation between Revenue and Trans Amount')
plt.xlabel('Revenue')
plt.ylabel('Trans Amount')
plt.legend(title='Attrition Flag', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



Distribusi jumlah transaksi per type transaksi



Data Preparation

Mengambil Usia Tertinggi Untuk Record data Terakhir

```python
#mengubah tipe data umur pelanggan menjadi numerik untuk keperluan analisis selanjutnya
credit_info['Customer_Age'] = pd.to_numeric(credit_info['Customer_Age'], errors='coerce')

# dikarenakan pada dataFrame terdapat beberapa record yang sama CLIENTNUM nya maka disini akan difilter data yang di
# merupakan data terakhir (berarti umur CLIENT tertinggi)
credit_info_sorted = credit_info.sort_values(by='Customer_Age', ascending=False)
latest_records = credit_info_sorted.drop_duplicates(subset='CLIENTNUM', keep='first')

# melakukan group by supaya kolom yang bersifat agregasi tetap relevan
sum_trans_amount = credit_info.groupby('CLIENTNUM')['Months_Inactive_12_mon', 'Contacts_Count_12_mon',
                                                    'Total_Revolving_Bal', 'Total_Trans_Ct'].sum().reset_index()

# disini disesuaikan jika sebelumnya memakai sum, namun ada beberapa kolom bersifat rata-rata oleh karena itu
# pakai Average
average_values = credit_info.groupby('CLIENTNUM')['Avg_Open_To_Buy', 'Avg_Utilization_Ratio'].mean().reset_index()

#lalu di merge supaya data menjadi satu dan tetap relevan dengan data yang sudah difilter
latest_records = pd.merge(latest_records, sum_trans_amount, on='CLIENTNUM', how='left')
latest_records = pd.merge(latest_records, average_values, on='CLIENTNUM', how='left')

latest_records
```

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_b
0	712672083	Existing Customer	65	F	0	High School	Married	Less than $40K	Blue	
1	770721858	Existing Customer	65	M	1	Graduate	Married	40K−60K	Blue	
2	780689733	Existing Customer	65	M	0	College	Single	40K−60K	Silver	
3	778357458	Existing Customer	65	F	3	Graduate	Married	Less than $40K	Blue	
4	778247358	Existing Customer	65	M	1	Graduate	Single	40K−60K	Silver	
...	...	...	...	...	...	...	...	...	...	
11566	900203138	Existing Customer	24	F	2	College	Married	Less than $40K	Silver	
11567	900203137	Existing Customer	24	F	0	Graduate	Single	60K−80K	Silver	
11568	900203136	Existing Customer	24	F	0	College	Single	60K−80K	Silver	
11569	900201723	Existing Customer	24	F	0	High School	Single	40K−60K	Silver	
11570	900203146	Existing Customer	24	M	1	College	Single	Less than $40K	Blue	

11571 rows × 27 columns

Hapus kolom yang tidak diperlukan

```
# Karena adanya penambahan kolumn akibat agregasi maka kolom lama yang ada inisial x dibelakangnya akan dihapus
columntodrop = ['Months_Inactive_12_mon_x', 'Contacts_Count_12_mon_x', 'Total_Revolving_Bal_x',
                'Avg_Open_To_Buy_x', 'Total_Trans_Ct_x', 'Avg_Utilization_Ratio_x']

# mengubah nama dataFrame supaya relevan dan sesuai yang sudah difilter
credit_info = latest_records.drop(columns=columntodrop)
```

Rename kolom supaya relevan

```
# melakukan rename terhadap kolom yang tadinya hasil merger supaya menjadi relevan
rename_columns = {'Months_Inactive_12_mon_y': 'Months_Inactive',
               'Contacts_Count_12_mon_y': 'Contacts_Count',
                 'Total_Revolving_Bal_y': 'Total_Revolving_Bal',
                'Total_Trans_Ct_y': 'Total_Trans_Ct',
                 'Avg_Open_To_Buy_y': 'Avg_Open_To_Buy',
                 'Avg_Utilization_Ratio_y': 'Avg_Utilization_Ratio'}

# Mengganti nama banyak kolom sekaligus
credit_info = credit_info.rename(columns=rename_columns)
```

Memfilter ‘credit_finance’ dijumlahkan kolom yang bersifat agregasi

```
# disini akan memfilter dataFrame kedua yang mengandung beberapa transaksi yang dilakukan nasabah
# tentunya untuk menghindari data duplicate disini menggunakan group by berdasarkan CLIENTNUM (identitas) dan diambi
# 'Trans_Amount' dan 'Revenue'
filtered_finance = credit_finance.groupby('CLIENTNUM', as_index=False).agg({'Trans_Amount': 'sum',
                                                                           'Revenue': 'sum'})

# setelah sesuai maka digabungkanlah DataFrame 'cust_credit' dan 'grouped_finance' berdasarkan 'CLIENTNUM'
data_credit = pd.merge(credit_info, filtered_finance, on='CLIENTNUM', how='left')
```

Mengubah Tipe Data

```
#ubah tipe data
categorical_columns = ['Gender', 'Education_Level', 'Marital_Status', 'Income_Category', 'Card_Category']

for col in categorical_columns:
    data_credit[col] = data_credit[col].astype('category')

data_credit['CLIENTNUM'] = data_credit['CLIENTNUM'].astype(str)
```

Memanipulasi Kolom Date

```python
# Memisahkan tahun dan kuartal
data_credit[['Quarter', 'Year']] = data_credit['Date_Leave'].str.split(',', expand=True)

# Mengonversi kuartal menjadi nilai numerik (hilangkan 'none' dengan 0)
data_credit['Quarter'] = data_credit['Quarter'].apply(lambda x: 0 if x == 'none' else int(x[1:]))

# Penyesuaian untuk menghitung awal dari setiap kuartal
data_credit['Month_Start'] = data_credit.apply(lambda x: (x['Quarter'] - 1) * 3 + 1 if x['Quarter'] > 0 else None, a

# Ubah 'Year_Quarter' menjadi datetime dengan format 'YYYY-MM-DD'
data_credit['Year_Quarter'] = pd.to_datetime(data_credit.apply(lambda x: f"{x['Year']}-{'{:02d}'.format(int(x['Month
                                                              if  pd.notna(x['Month_Start']) else None, axis=1), er

# Hapus kolom yang tidak diperlukan
data_credit = data_credit.drop(columns=['Quarter', 'Year', 'Month_Start'])
```

Hasil :

```python
data_credit['Year_Quarter'].unique()
```

```
array([                          'NaT', '2018-10-01T00:00:00.000000000',
       '2018-07-01T00:00:00.000000000', '2018-01-01T00:00:00.000000000',
       '2018-04-01T00:00:00.000000000', '2019-07-01T00:00:00.000000000',
       '2019-04-01T00:00:00.000000000', '2019-01-01T00:00:00.000000000',
       '2019-10-01T00:00:00.000000000'], dtype='datetime64[ns]')
```

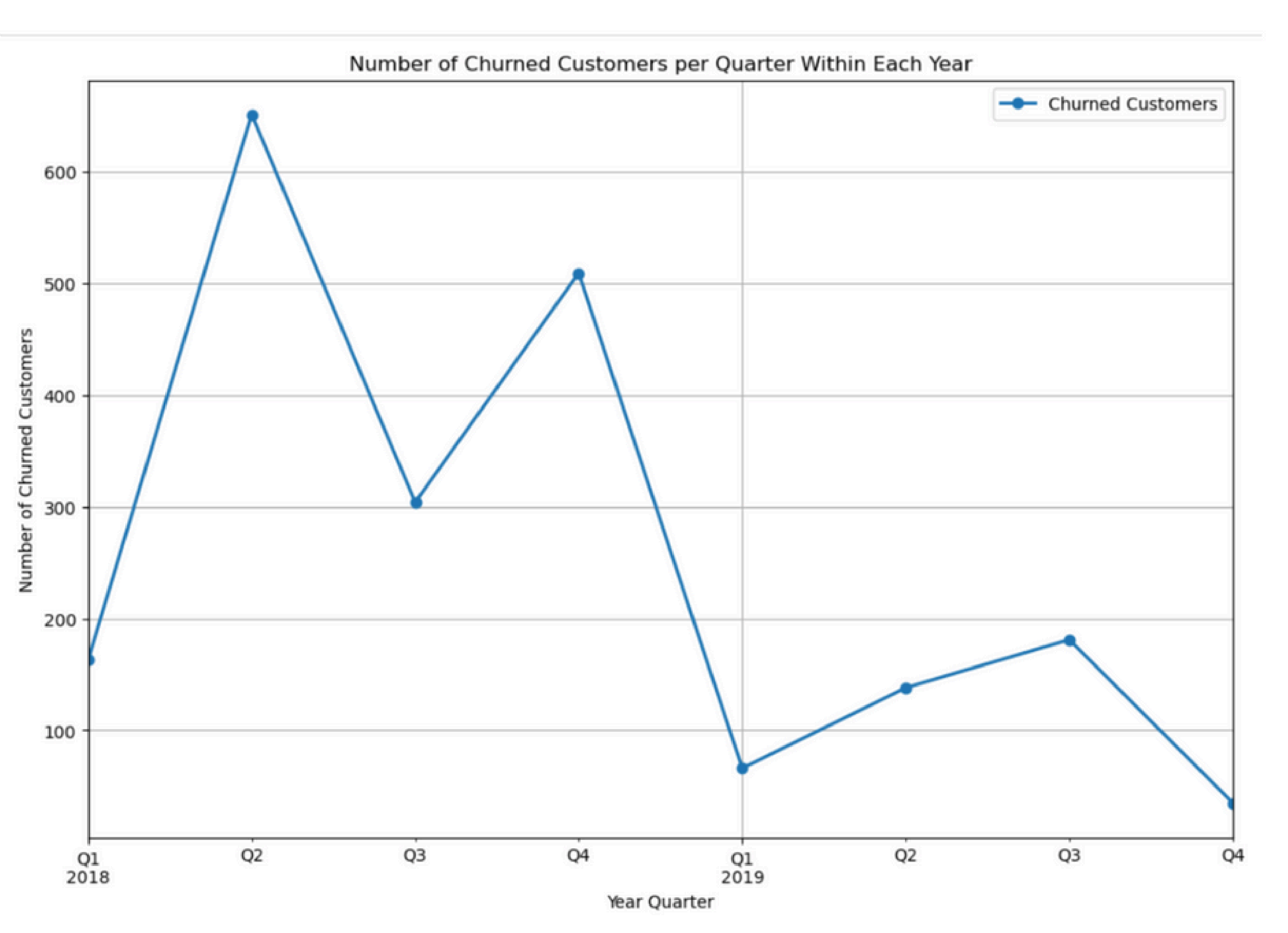
Cek informasi data terbaru

```
data_credit.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10038 entries, 0 to 11570
Data columns (total 22 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   CLIENTNUM                 10038 non-null  object
 1   Attrition_Flag            10038 non-null  object
 2   Customer_Age              10038 non-null  int64
 3   Gender                    10038 non-null  category
 4   Dependent_count           10038 non-null  int64
 5   Education_Level           10038 non-null  category
 6   Marital_Status            10038 non-null  category
 7   Income_Category           10038 non-null  category
 8   Card_Category             10038 non-null  category
 9   Months_on_book            10038 non-null  int64
 10  Total_Relationship_Count  10038 non-null  int64
 11  Credit_Limit              10038 non-null  float64
 12  Date_Leave                10038 non-null  object
 13  Months_Inactive           10038 non-null  int64
 14  Contacts_Count            10038 non-null  int64
 15  Total_Revolving_Bal       10038 non-null  int64
 16  Total_Trans_Ct            10038 non-null  int64
 17  Avg_Open_To_Buy           10038 non-null  float64
 18  Avg_Utilization_Ratio     10038 non-null  float64
 19  Trans_Amount              10038 non-null  float64
 20  Revenue                   10038 non-null  float64
 21  Year_Quarter              1806 non-null   datetime64[ns]
dtypes: category(5), datetime64[ns](1), float64(5), int64(8), object(3)
memory usage: 1.4+ MB
```

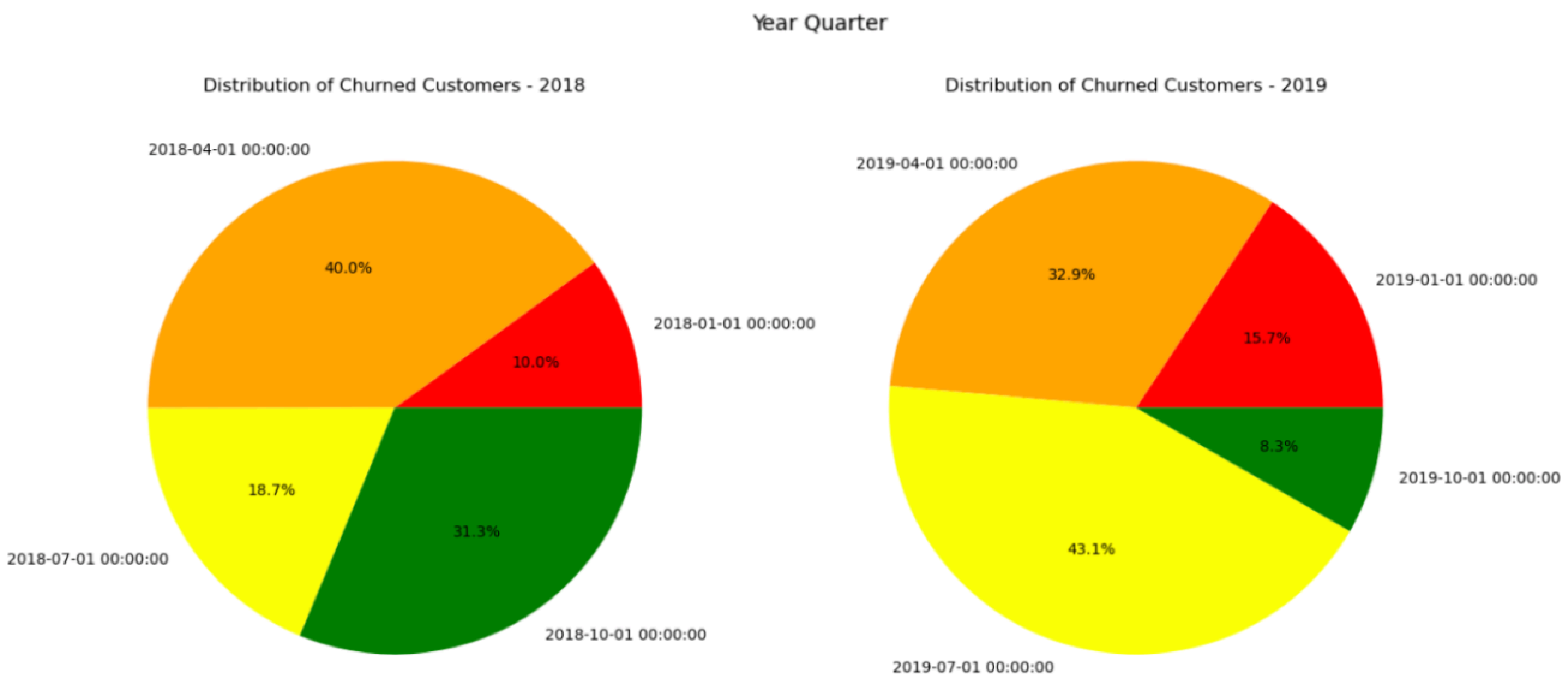
Visualisasi Trend Churn Per Quarter

```python
# Visualisasi untuk customer yang churn per quarter
churned_per_year_quarter = data_credit[data_credit['Attrition_Flag'] == 'Attrited Customer'].groupby(['Year_Quarter'

fig, ax = plt.subplots(figsize=(12, 8))
churned_per_year_quarter.plot(marker='o', linestyle='-', linewidth=2, label='Churned Customers')
ax.set_title('Number of Churned Customers per Quarter Within Each Year')
ax.set_xlabel('Year Quarter')
ax.set_ylabel('Number of Churned Customers')
ax.legend()
ax.grid(True)
plt.show()
```



Visualisasi Komposisi Customer yang churn

```
# Filter pelanggan yang churned
churned_customers = data_credit[data_credit['Attrition_Flag'] == 'Attrited Customer']

# Visualisasi: Pie chart - Distribution of churned customers across quarters for each year
unique_years = churned_customers['Year_Quarter'].dt.year.unique()

fig, axes = plt.subplots(nrows=1, ncols=len(unique_years), figsize=(16, 8))

for i, year in enumerate(unique_years):
    churned_per_year_quarter = churned_customers[churned_customers['Year_Quarter'].dt.year == year].groupby(['Year_Q
    axes[i].pie(churned_per_year_quarter, labels=churned_per_year_quarter.index, autopct='%1.1f%%', colors=['red', '
    axes[i].set_title(f'Distribution of Churned Customers - {year}')

plt.show()
```



Hapus Outlier

```python
# Menentukan kolom numerik yang akan dihapus outlier-nya
numeric_columns = ['Customer_Age', 'Dependent_count', 'Months_on_book', 'Total_Relationship_Count',
                   'Credit_Limit', 'Months_Inactive', 'Contacts_Count', 'Total_Revolving_Bal', 'Total_Trans_Ct',
                   'Avg_Open_To_Buy', 'Avg_Utilization_Ratio', 'Trans_Amount', 'Revenue']

# Menghitung IQR
Q1 = data_credit[numeric_columns].quantile(0.25)
Q3 = data_credit[numeric_columns].quantile(0.75)
IQR = Q3 - Q1

# deklarasi lower dan upper bound
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Proses membersihkan data outlier
data_credit_cleaned = data_credit.copy()
for column in numeric_columns:
    data_credit_cleaned = data_credit_cleaned[(data_credit_cleaned[column] >= lower_bound[column]) &
                                              (data_credit_cleaned[column] <= upper_bound[column])]

data_credit = data_credit_cleaned
```

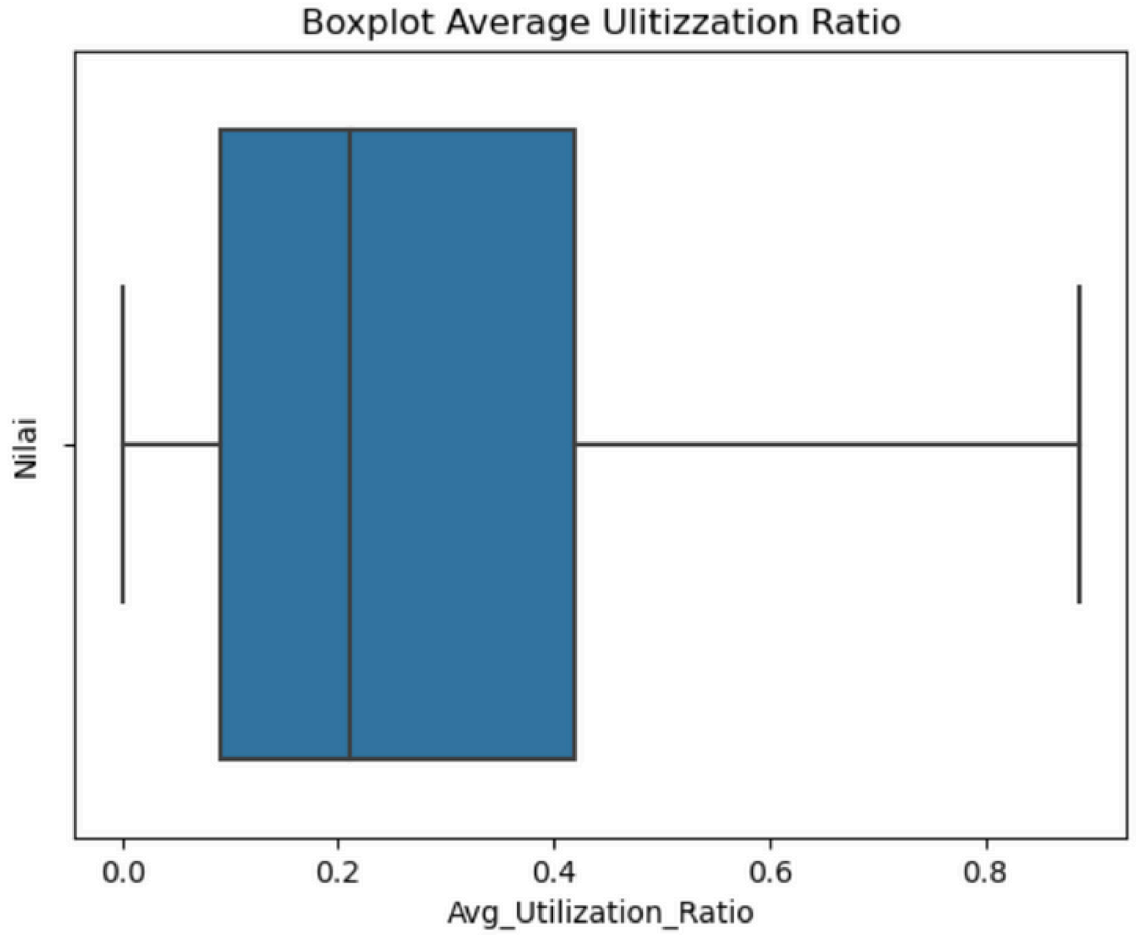
```python
data_credit
```

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_b
0	712672083	Existing Customer	65	F	0	High School	Married	Less than $40K	Blue	
1	770721858	Existing Customer	65	M	1	Graduate	Married	40K−60K	Blue	
2	780689733	Existing Customer	65	M	0	College	Single	40K−60K	Silver	
3	778357458	Existing Customer	65	F	3	Graduate	Married	Less than $40K	Blue	
4	778247358	Existing Customer	65	M	1	Graduate	Single	40K−60K	Silver	
...	...	...	...	...	...	...	...	...	...	
11566	900203138	Existing Customer	24	F	2	College	Married	Less than $40K	Silver	
11567	900203137	Existing Customer	24	F	0	Graduate	Single	60K−80K	Silver	
11568	900203136	Existing Customer	24	F	0	College	Single	60K−80K	Silver	
11569	900201723	Existing Customer	24	F	0	High School	Single	40K−60K	Silver	
11570	900203146	Existing Customer	24	M	1	College	Single	Less than $40K	Blue	

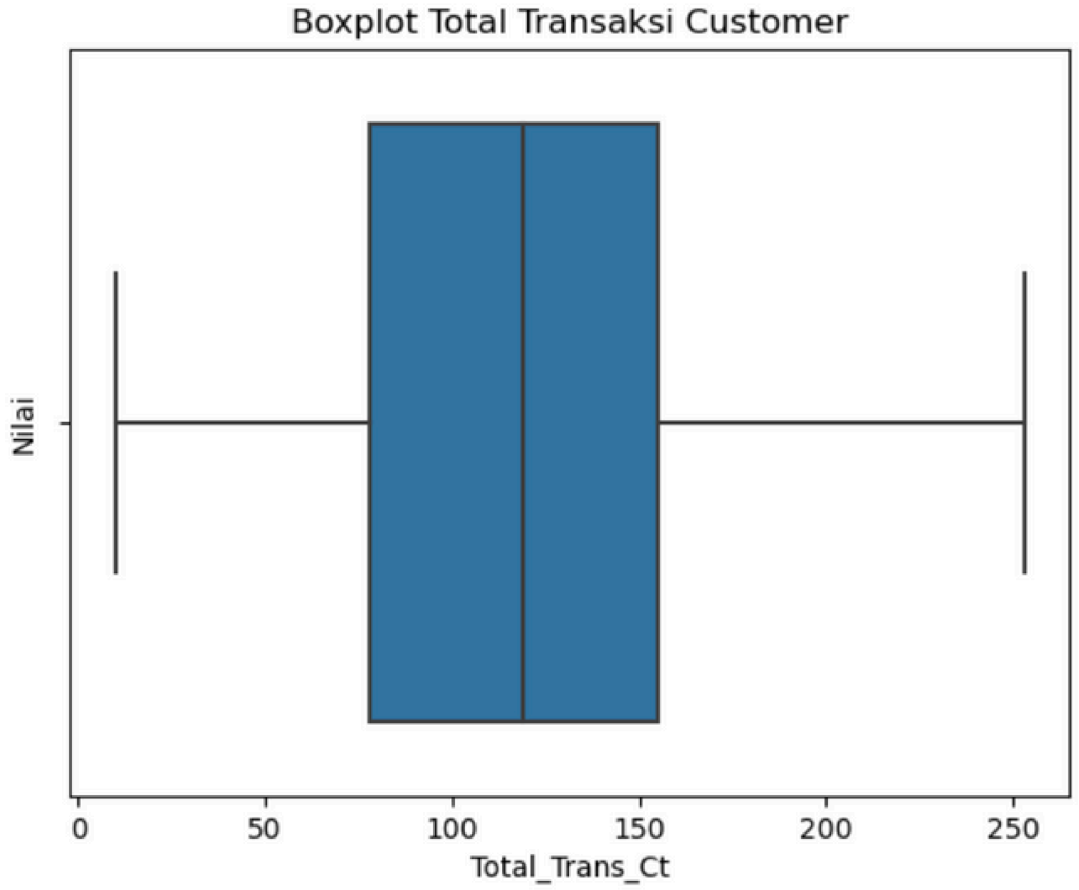
10038 rows × 22 columns

Cek Beberapa Kolom yang sudah dibersihkan outliernya

```
sns.boxplot(x='Avg_Utilization_Ratio', data=data_credit)
plt.title('Boxplot Average Ulitizzation Ratio')
plt.ylabel('Nilai')
plt.show()
```




```
sns.boxplot(x='Total_Trans_Ct', data=data_credit)
plt.title('Boxplot Total Transaksi Customer')
plt.ylabel('Nilai')
plt.show()
```



Encoding

```
selected_columns = ['Attrition_Flag', 'Gender', 'Education_Level','Marital_Status',
                    'Income_Category','Card_Category'
                   ]

# Menampilkan nilai unik di kolom-kolom tertentu
for column in selected_columns:
    unique_values = data_credit[column].unique()
    print(f"Unique values in column {column}:", unique_values)
```

```
Unique values in column Attrition_Flag: ['Existing Customer' 'Attrited Customer']
Unique values in column Gender: ['F', 'M']
Categories (2, object): ['F', 'M']
Unique values in column Education_Level: ['High School', 'Graduate', 'College', 'Uneducated', 'Post-Graduate', 'Doc
torate']
Categories (6, object): ['College', 'Doctorate', 'Graduate', 'High School', 'Post-Graduate', 'Uneducated']
Unique values in column Marital_Status: ['Married', 'Single', 'Divorced']
Categories (3, object): ['Divorced', 'Married', 'Single']
Unique values in column Income_Category: ['Less than $40K', '$40K - $60K', '$80K - $120K', '$60K - $80K', '$120K
+']
Categories (5, object): ['$120K +', '$40K - $60K', '$60K - $80K', '$80K - $120K', 'Less than $40K']
Unique values in column Card_Category: ['Blue', 'Silver', 'Gold', 'Platinum']
Categories (4, object): ['Blue', 'Gold', 'Platinum', 'Silver']
```

```
# Membuat peta encoding manual untuk setiap kolom
attrition_flag_mapping = {'Attrited Customer': 1, 'Existing Customer': 0}
gender_mapping = {'F': 0, 'M': 1}
education_level_mapping = {'High School': 0, 'Graduate': 1, 'College': 2, 'Uneducated': 3, 'Post-Graduate': 4, 'Doct
marital_status_mapping = {'Divorced': 0, 'Married': 1, 'Single': 2}
income_category_mapping = {'Less than $40K': 0, '$40K - $60K': 1, '$60K - $80K': 2, '$80K - $120K': 3, '$120K +': 4}
card_category_mapping = {'Blue': 0, 'Silver': 1, 'Gold': 2, 'Platinum': 3}

# Proses encoding memacu peta yang udah ditentukan
data_credit['Attrition_Flag'] = data_credit['Attrition_Flag'].map(attrition_flag_mapping)
data_credit['Gender'] = data_credit['Gender'].map(gender_mapping)
data_credit['Education_Level'] = data_credit['Education_Level'].map(education_level_mapping)
data_credit['Marital_Status'] = data_credit['Marital_Status'].map(marital_status_mapping)
data_credit['Income_Category'] = data_credit['Income_Category'].map(income_category_mapping)
data_credit['Card_Category'] = data_credit['Card_Category'].map(card_category_mapping)
```

```
# cek hasil
for column in ['Attrition_Flag', 'Gender', 'Education_Level', 'Marital_Status', 'Income_Category', 'Card_Category']:
    unique_values = data_credit[column].unique()
    print(f"Encoded values in column {column}:", unique_values)
```

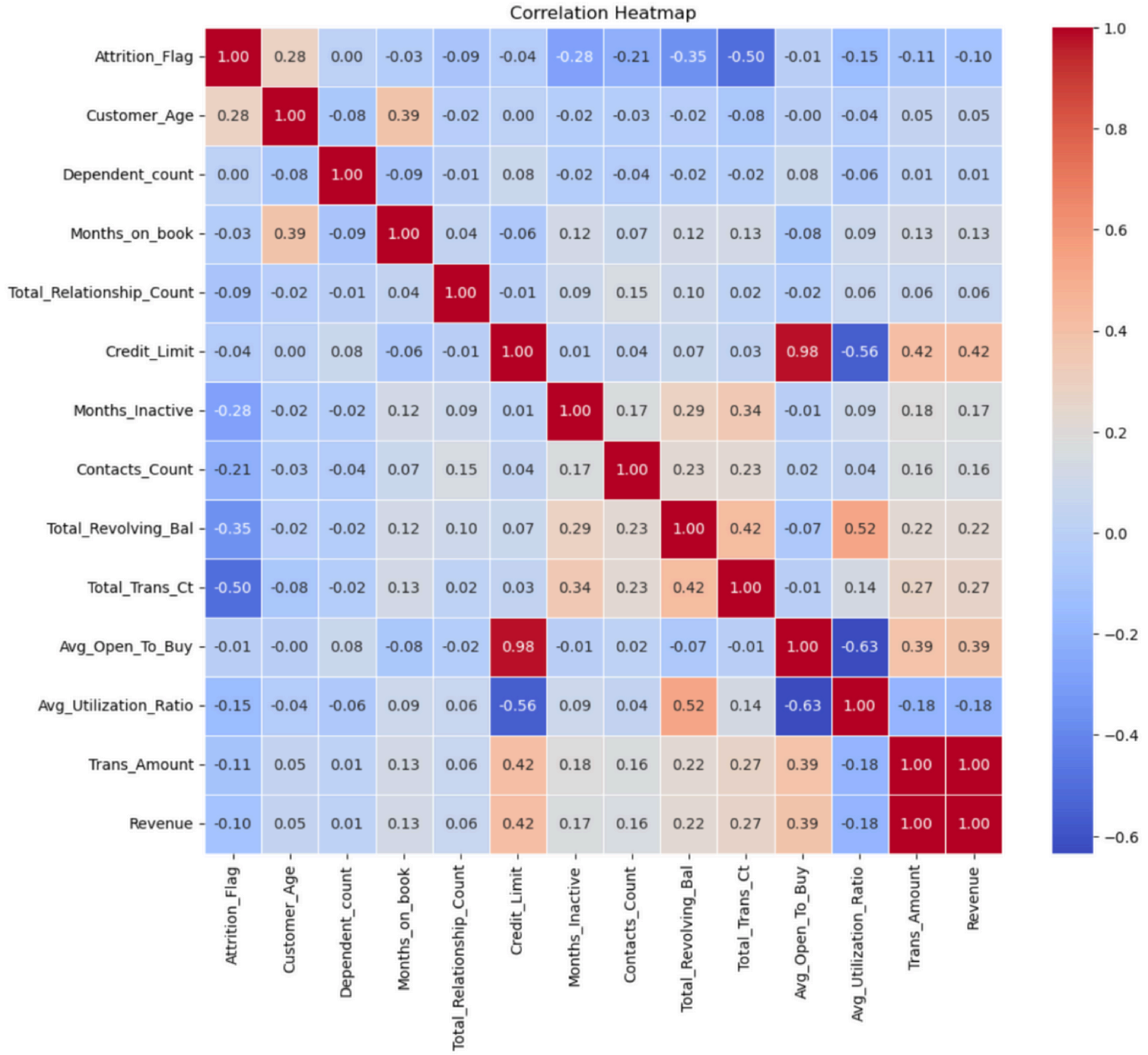
```
data_credit
```

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_b
0	712672083	0	65	0	0	0	1	0	0	
1	770721858	0	65	1	1	1	1	1	0	
2	780689733	0	65	1	0	2	2	1	1	
3	778357458	0	65	0	3	1	1	0	0	
4	778247358	0	65	1	1	1	2	1	1	
...	...	...	...	...	...	...	...	...	...	
11566	900203138	0	24	0	2	2	1	0	1	
11567	900203137	0	24	0	0	1	2	2	1	
11568	900203136	0	24	0	0	2	2	2	1	
11569	900201723	0	24	0	0	0	2	1	1	
11570	900203146	0	24	1	1	2	2	0	0	

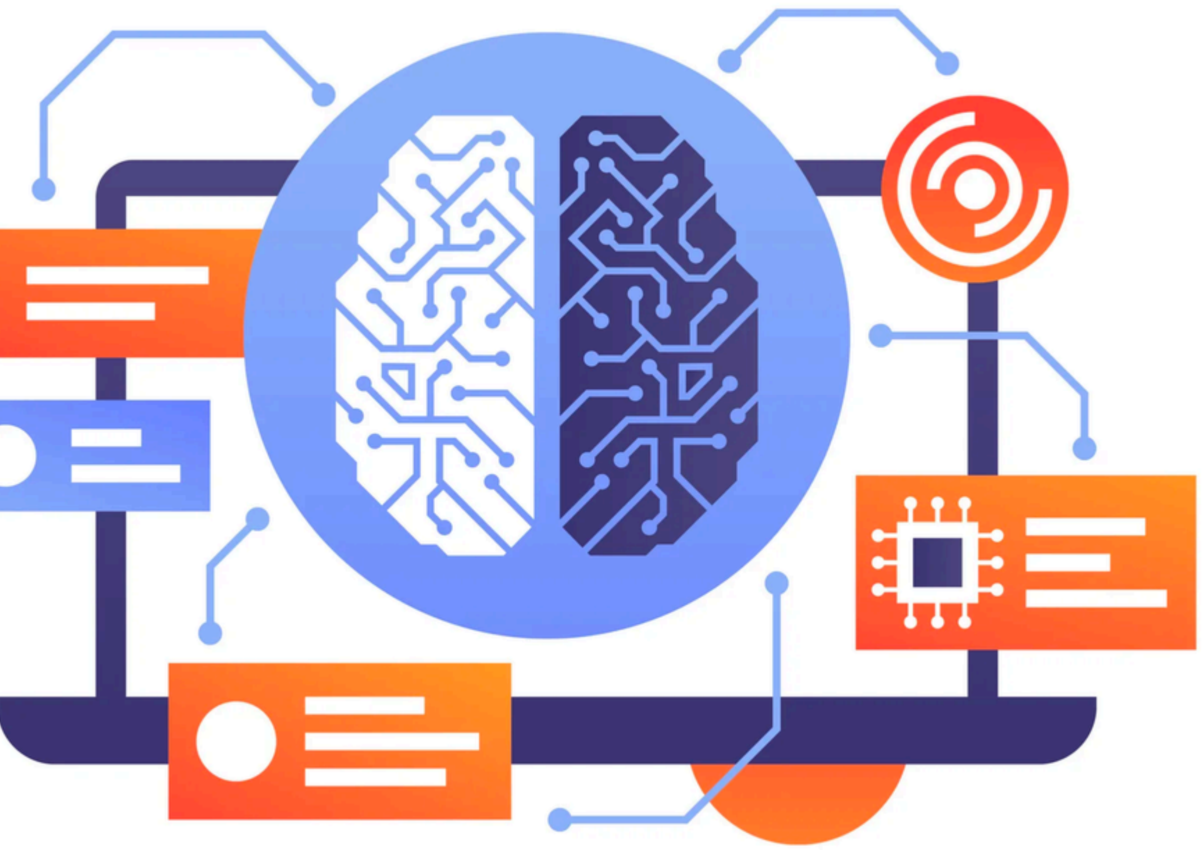
Visualisasi Korelasi

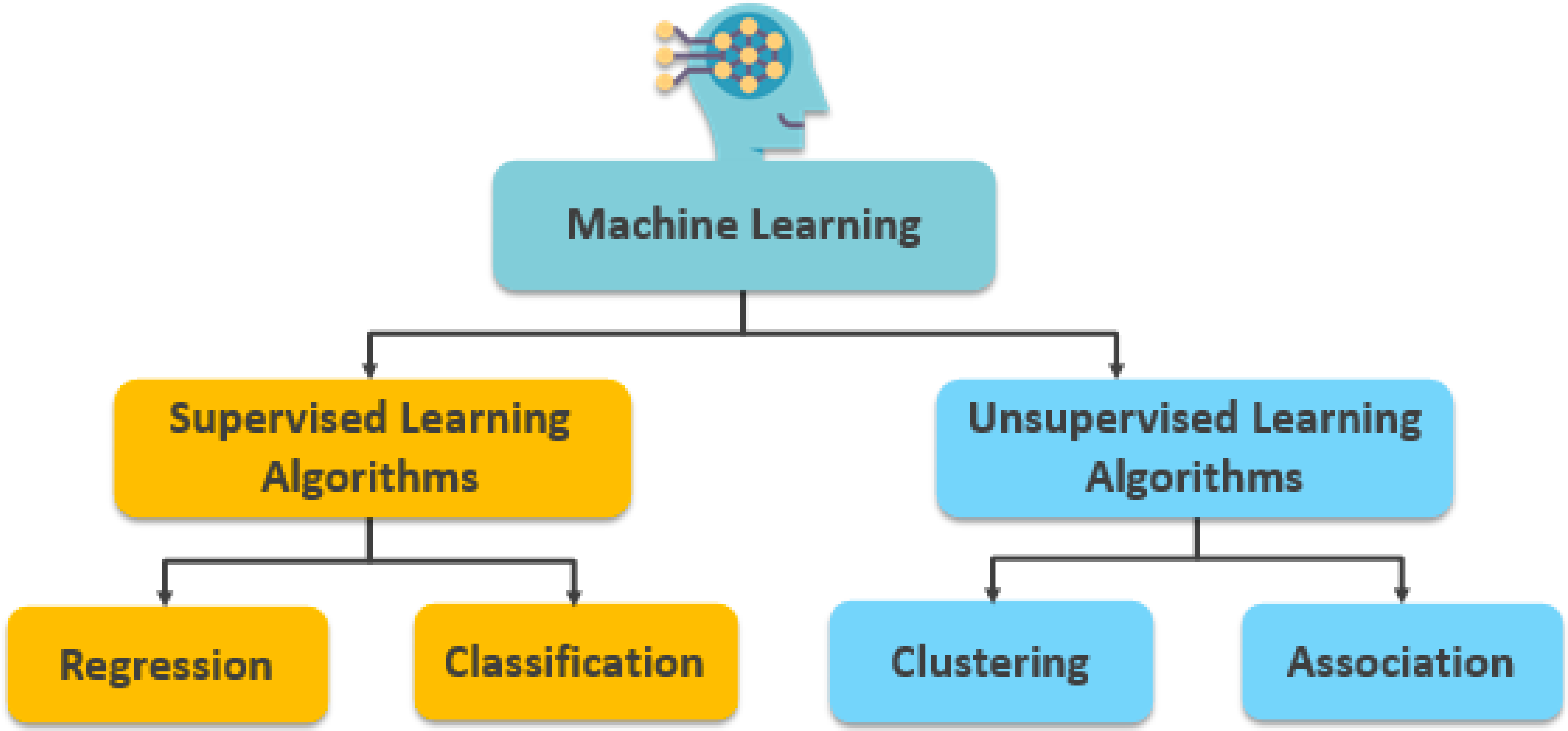
```
# Memilih variabel-variabel
selected_variables = ['CLIENTNUM', 'Attrition_Flag', 'Customer_Age', 'Gender',
       'Dependent_count', 'Education_Level', 'Marital_Status',
       'Income_Category', 'Card_Category', 'Months_on_book',
       'Total_Relationship_Count', 'Credit_Limit', 'Date_Leave',
       'Months_Inactive', 'Contacts_Count', 'Total_Revolving_Bal',
       'Total_Trans_Ct', 'Avg_Open_To_Buy', 'Avg_Utilization_Ratio',
       'Trans_Amount', 'Revenue', 'Year_Quarter']
# Subset dataset menggunakan variabel-variabel utama
selected_data = data_credit[selected_variables]

# Menghitung matriks korelasi
correlation_matrix = selected_data.corr()

# Membuat visual korelasi heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```

Heat map correlation



Modeling and Evaluation





Pemilihan fitur yang memiliki pengaruh tertinggi dengan Random Forest

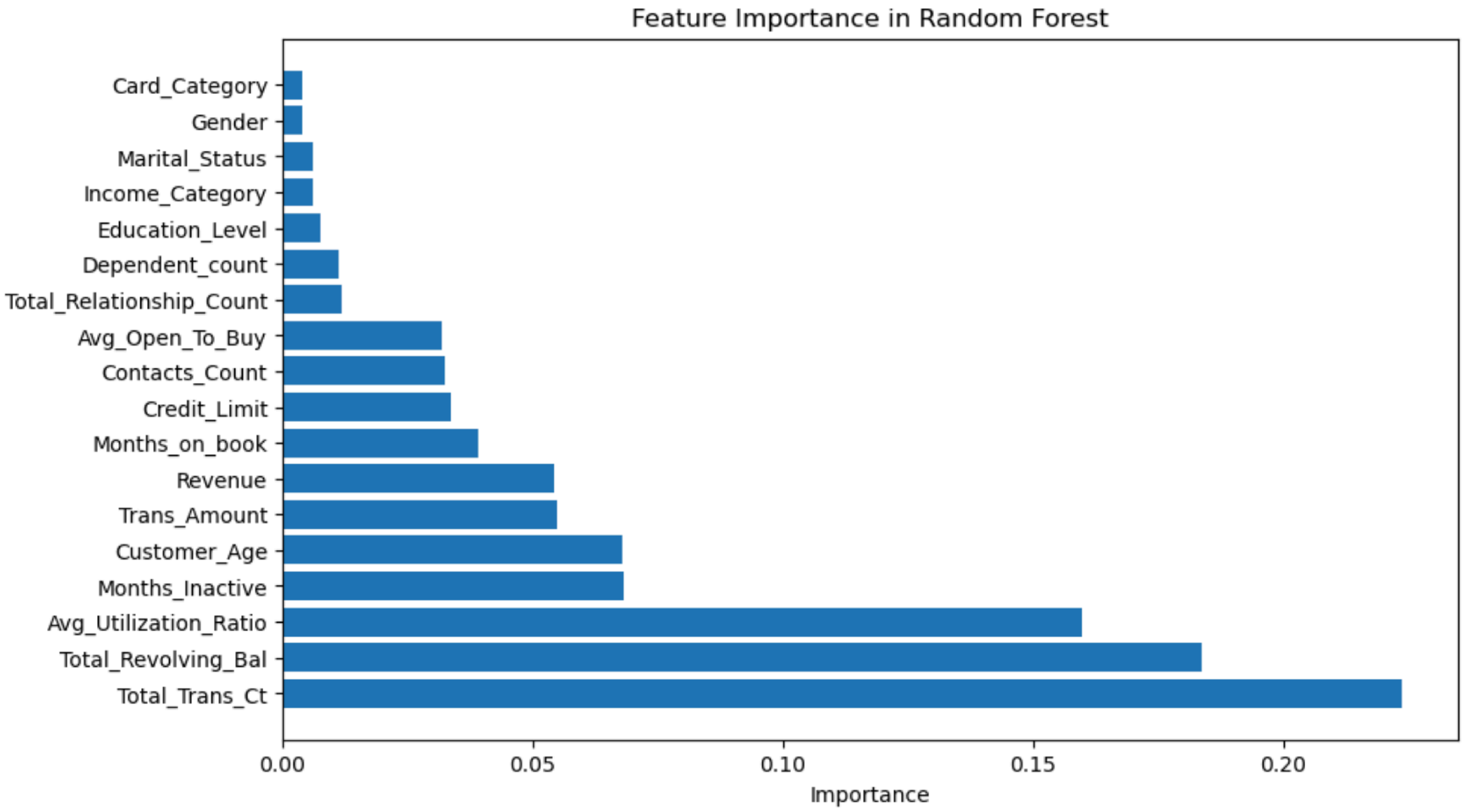
```
# pilih variabel fitur
features = ['Customer_Age', 'Gender', 'Dependent_count', 'Education_Level', 'Marital_Status',
            'Income_Category', 'Card_Category', 'Months_on_book', 'Total_Relationship_Count',
            'Credit_Limit', 'Months_Inactive', 'Contacts_Count', 'Total_Revolving_Bal',
            'Total_Trans_Ct', 'Avg_Open_To_Buy', 'Avg_Utilization_Ratio', 'Trans_Amount', 'Revenue']

target = 'Attrition_Flag'

X = data_credit[features]
y = data_credit[target]
```

```
# Pisahkan data menjadi set pelatihan dan pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Membuat model Random Forest
model_rf = RandomForestClassifier(n_estimators=100, random_state=42)
model_rf.fit(X_train, y_train)

# Menghitung feature importance
feature_importances = model_rf.feature_importances_
feature_importance_df = pd.DataFrame({'Feature': features, 'Importance': feature_importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)
```


```
# Menghitung feature importance
feature_importances = model_rf.feature_importances_
feature_importance_df = pd.DataFrame({'Feature': features, 'Importance': feature_importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)
```


```
# Visualisasi feature importance
plt.figure(figsize=(10, 6))
plt.barh(feature_importance_df['Feature'], feature_importance_df['Importance'])
plt.xlabel('Importance')
plt.title('Feature Importance in Random Forest')
plt.show()

# Evaluasi akurasi model pada data pengujian
accuracy_rf = model_rf.score(X_test, y_test)
print(f"Accuracy of Random Forest on Test Data: {accuracy_rf:.4f}")
```

Visualisasi Importance Feature




```
Accuracy of Random Forest on Test Data: 0.9502
```

Modeling dengan Decision Tree Algorithms
(Klasifikasi Apakah Pelanggan akan Churn atau tidak)

Modeling dengan Decision Tree

```
# Pilih fitur yang akan digunakan untuk prediksi
features = ['Total_Trans_Ct', 'Total_Revolving_Bal', 'Customer_Age', 'Avg_Utilization_Ratio', 'Months_Inactive']

# Pilih target (variabel yang ingin diprediksi)
target = 'Attrition_Flag'
```

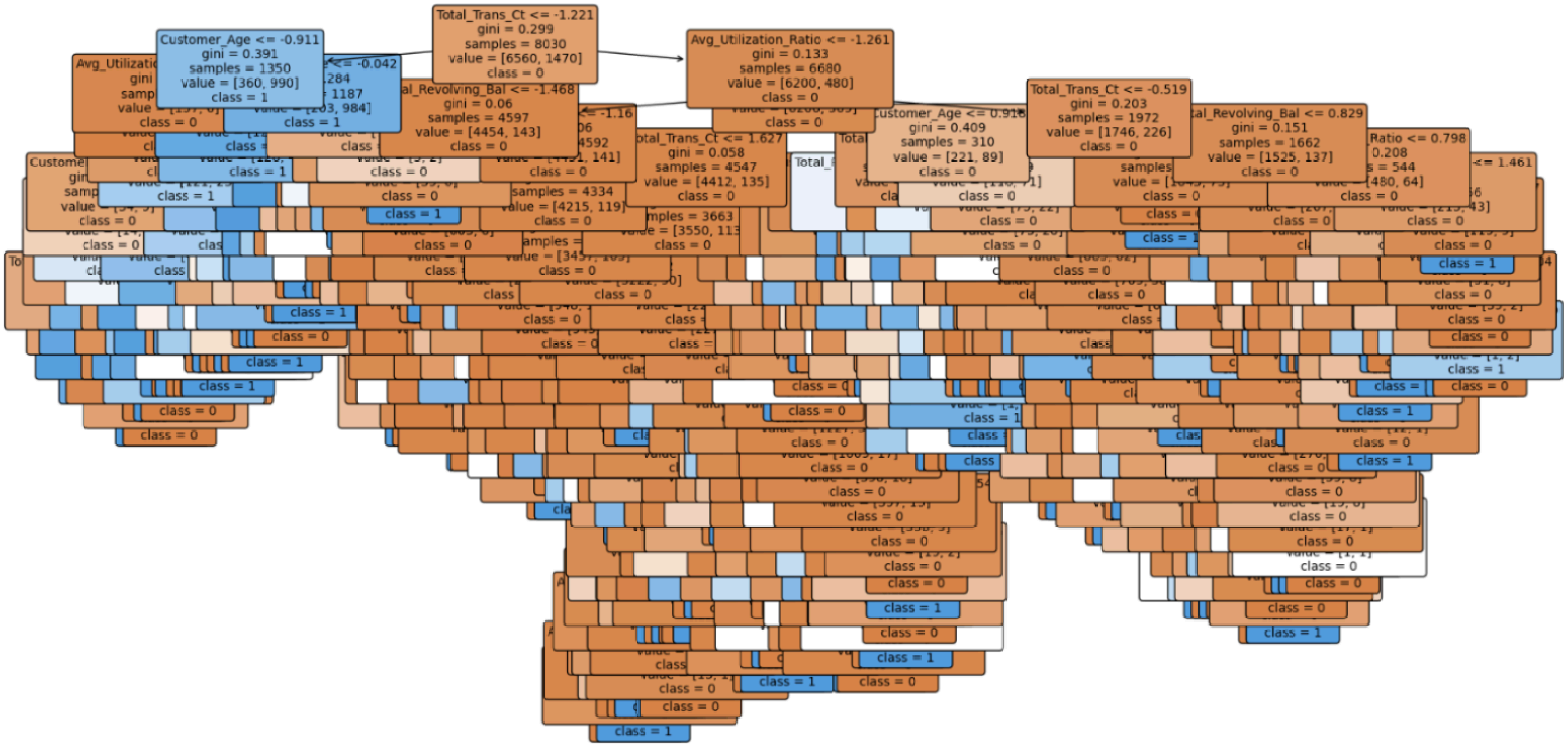
```
# Pisahkan data menjadi fitur (X) dan target (y)
X = data_credit[features]
y = data_credit[target]

# Pisahkan data menjadi set pelatihan dan pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Normalisasi data
scaler = StandardScaler()
X_train_normalized = scaler.fit_transform(X_train)
X_test_normalized = scaler.transform(X_test)
```

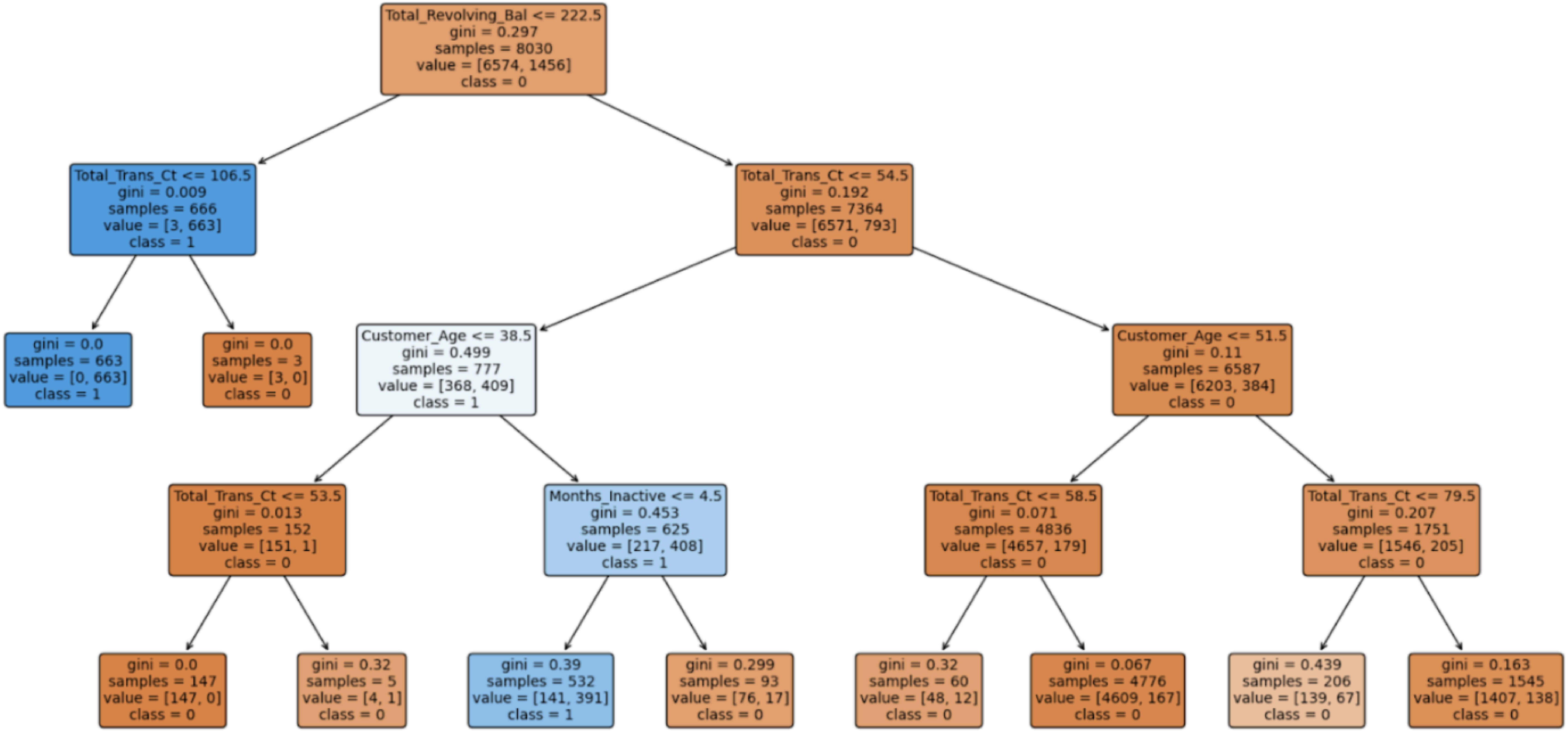
```
# Random sampling
X_train_resampled, y_train_resampled = resample(X_train_normalized, y_train, random_state=42)

# Membuat model Decision Tree
model = DecisionTreeClassifier(max_depth=4, random_state=42)
model.fit(X_train_resampled, y_train_resampled)
```

```
# Visualisasi Decision Tree
plt.figure(figsize=(20, 10))
plot_tree(model, feature_names=features, class_names=[str(cls) for cls in model.classes_], filled=True,
          rounded=True, fontsize=10)
plt.savefig('decision_tree.png')
plt.show()

# Melakukan prediksi pada set pengujian
y_pred = model.predict(X_test_normalized)
```

Hasil Descision Tree dengan Max_Depth = none



Hasil Descision Tree dengan Max_Depth = 4



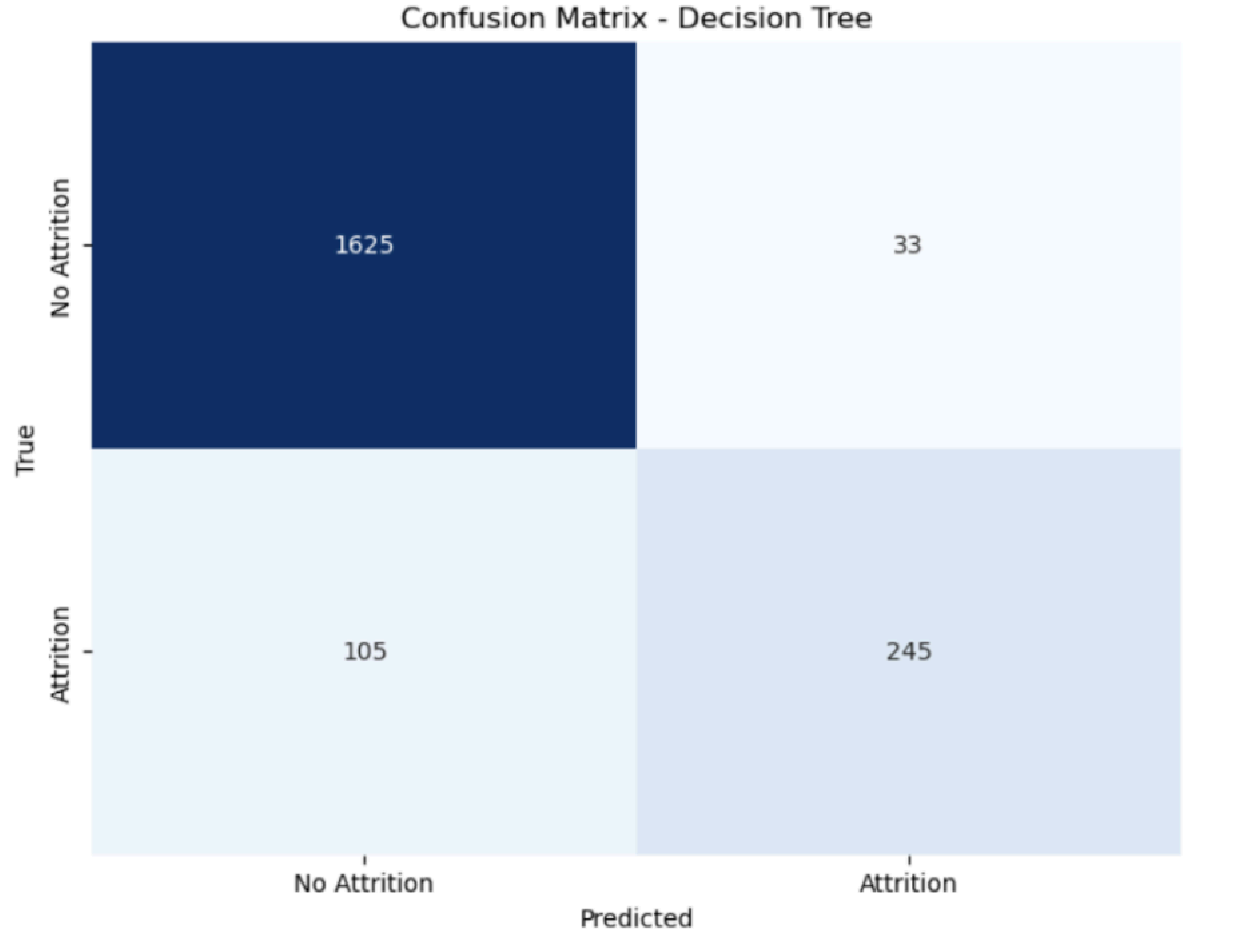
Confusion Matrix

```
# Evaluasi kinerja model
accuracy_DT = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

# Tampilkan hasil
print(f'Accuracy: {accuracy_DT:.4f}')
print(f'Confusion Matrix:\n{conf_matrix}')
print(f'Classification Report:\n{class_report}')

# Menghitung confusion matrix untuk Decision Tree
conf_matrix_dt = confusion_matrix(y_test, y_pred)

# Membuat heatmap untuk Decision Tree
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_dt, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['No Attrition', 'Attrition'],
            yticklabels=['No Attrition', 'Attrition'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix - Decision Tree')
plt.show()
```

Confusion Matrix

```
Accuracy: 0.9313
Confusion Matrix:
[[1625   33]
 [ 105  245]]
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.98      0.96      1658
           1       0.88      0.70      0.78       350

    accuracy                           0.93      2008
   macro avg       0.91      0.84      0.87      2008
weighted avg       0.93      0.93      0.93      2008
```

Heatmap Confusion Matrix Decision Tree



- True Positive (TP): 245 (Churn yang diprediksi benar)
- True Negative (TN): 1625 (Tidak Churn yang diprediksi benar)
- False Positive (FP): 33 (Tidak Churn yang diprediksi Churn)
- False Negative (FN): 105 (Churn yang diprediksi tidak Churn)

Modeling dengan SVM Algorithms (Klasifikasi Apakah Pelanggan akan Churn atau tidak)

SVM

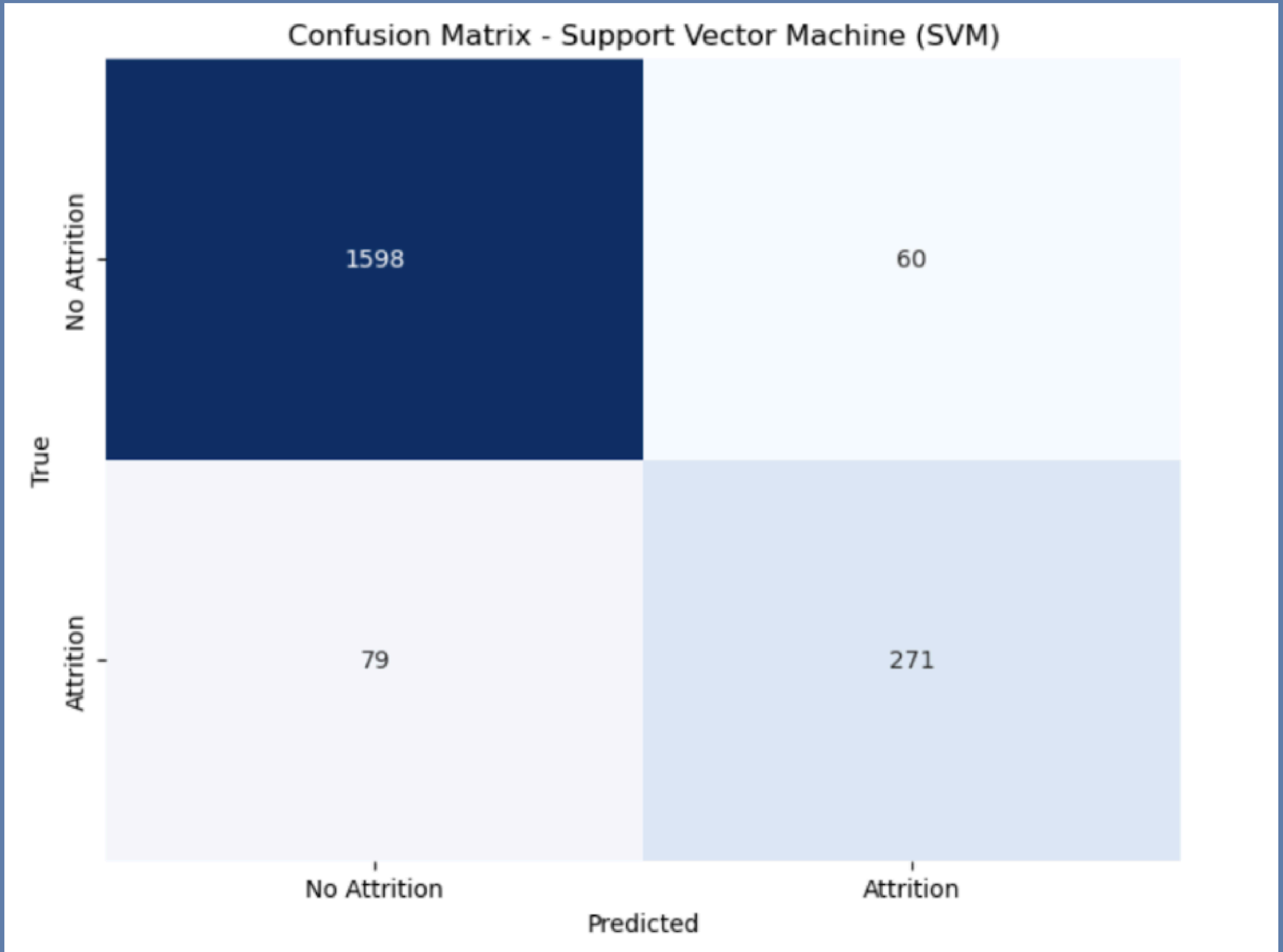
```
# Evaluasi model SVM
print("\nSupport Vector Machine (SVM) setelah oversampling:")
print("Accuracy:", accuracy_score(y_test, y_pred_svm))
print("Classification Report:\n", classification_report(y_test, y_pred_svm))
```

```
# Menghitung confusion matrix
conf_matrix_svm = confusion_matrix(y_test, y_pred_svm)

# Membuat heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_svm, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['No Attrition', 'Attrition'],
            yticklabels=['No Attrition', 'Attrition'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix - Support Vector Machine (SVM)')
plt.show()
```

Hasil Confusion Matrix SVM

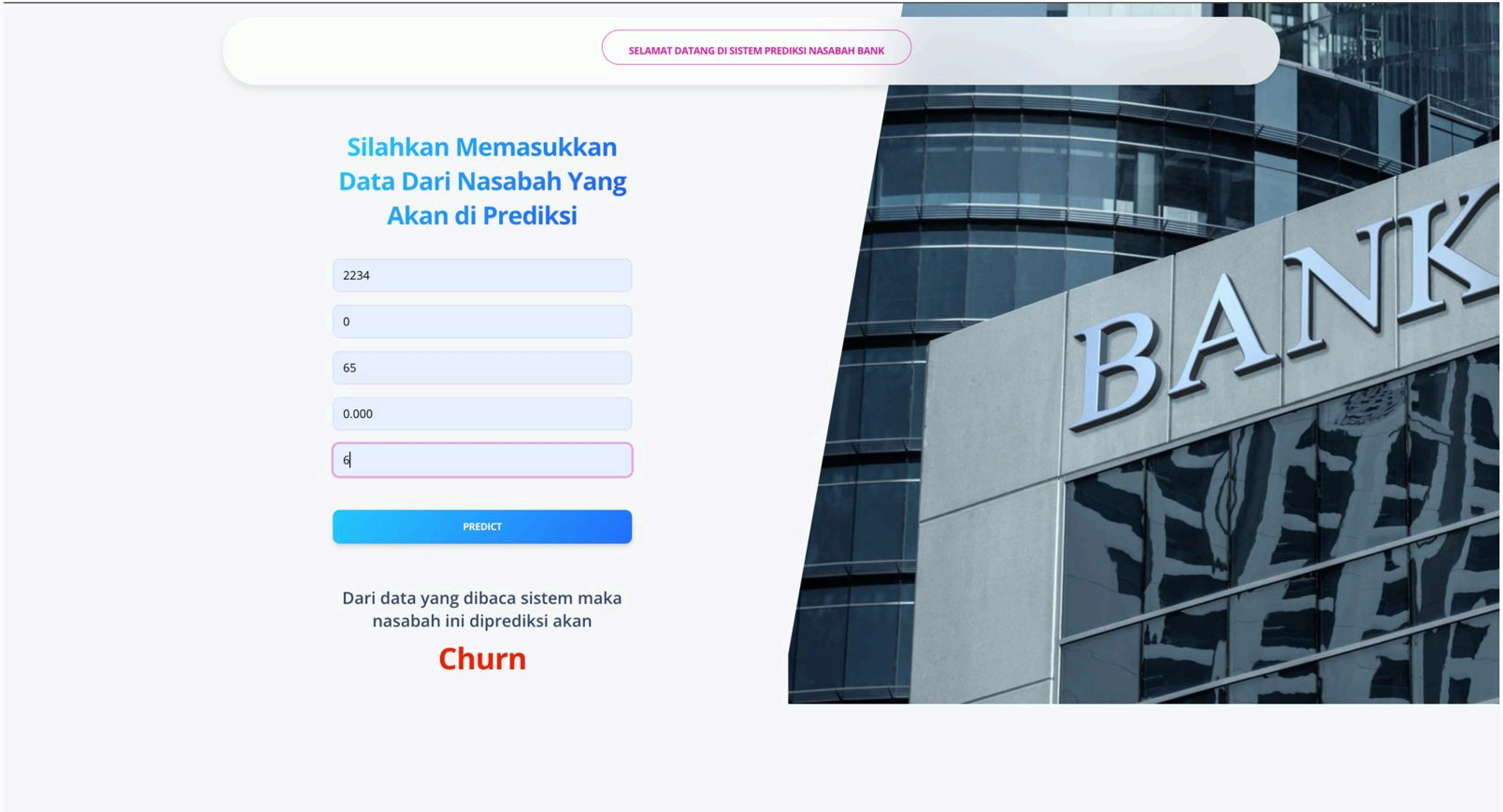
```
Support Vector Machine (SVM) setelah oversampling:
Accuracy: 0.9307768924302788
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.96      0.96      1658
           1       0.82      0.77      0.80       350

    accuracy                           0.93      2008
   macro avg       0.89      0.87      0.88      2008
weighted avg       0.93      0.93      0.93      2008
```

Heatmap Confusion Matrix SVM



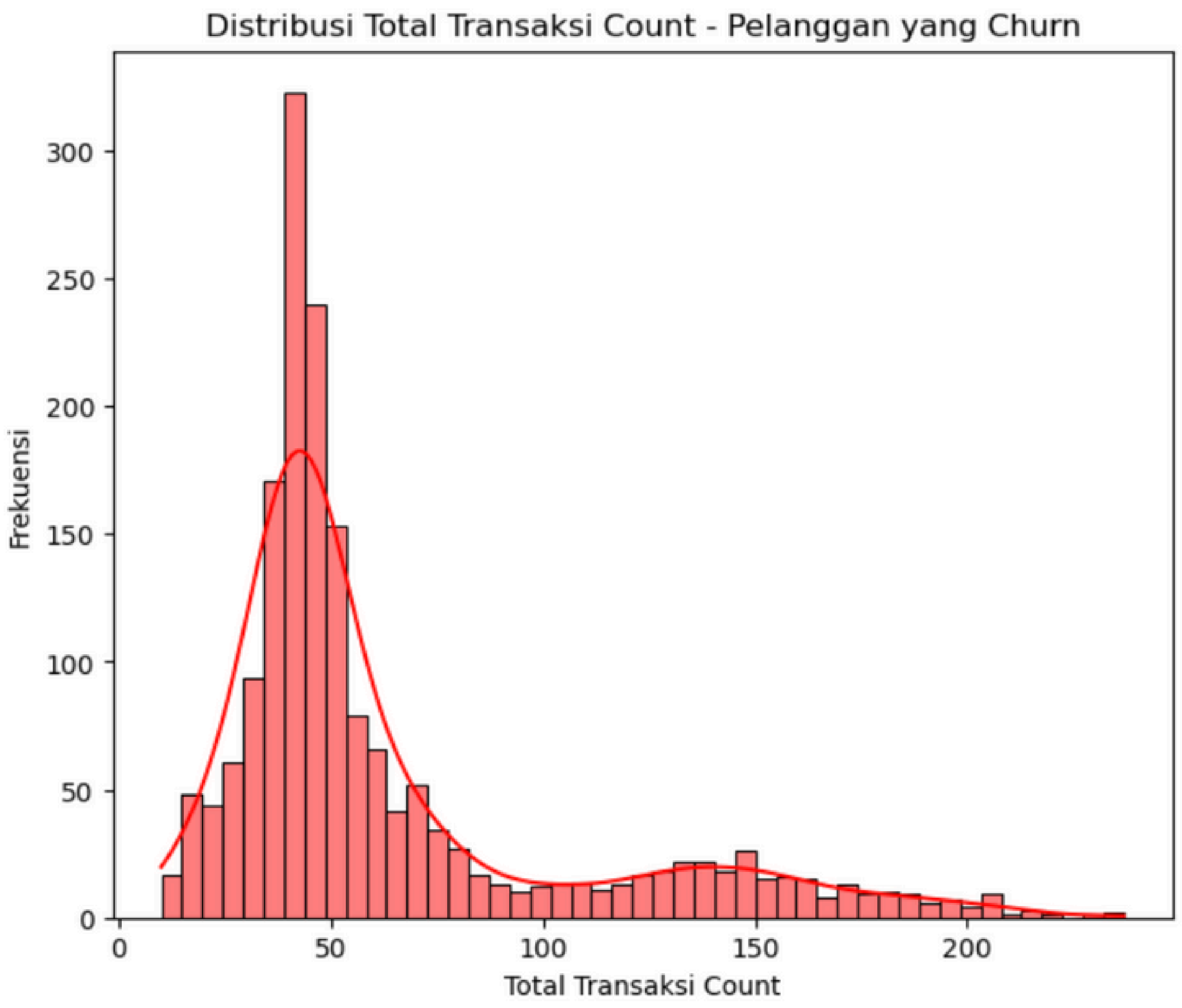
- True Positive (TP): 271 (Churn yang diprediksi benar)
- True Negative (TN): 1598 (Tidak Churn yang diprediksi benar)
- False Positive (FP): 60 (Tidak Churn yang diprediksi Churn)
- False Negative (FN): 79 (Churn yang diprediksi tidak Churn)

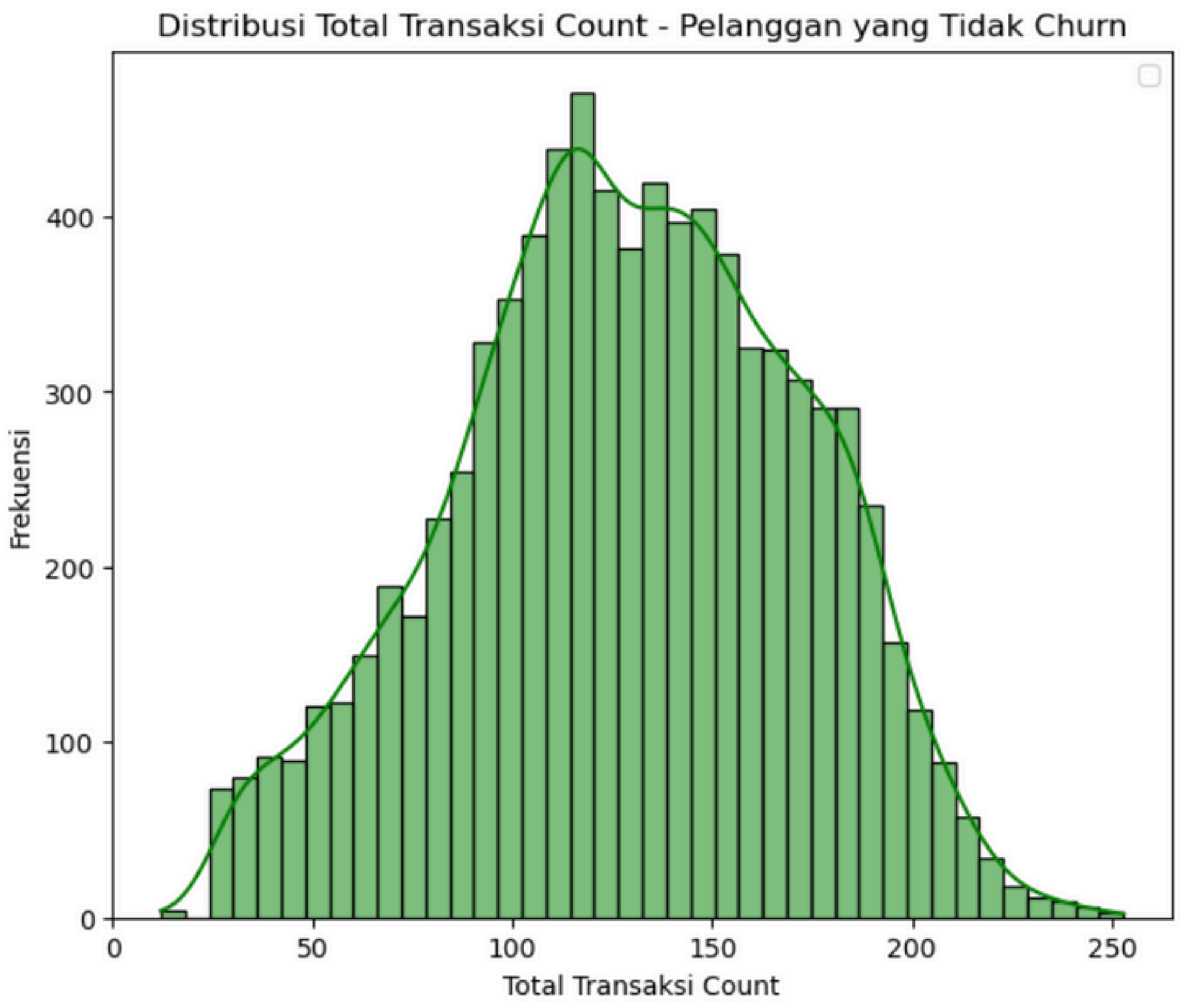
Deployment


SELAMAT DATANG DI SISTEM PREDIKSI NASABAH BANK
Silahkan Memasukkan Data Dari Nasabah Yang Akan di Prediksi
2234
0
65
0.000
6
PREDICT
Dari data yang dibaca sistem maka nasabah ini diprediksi akan
Churn
BANK

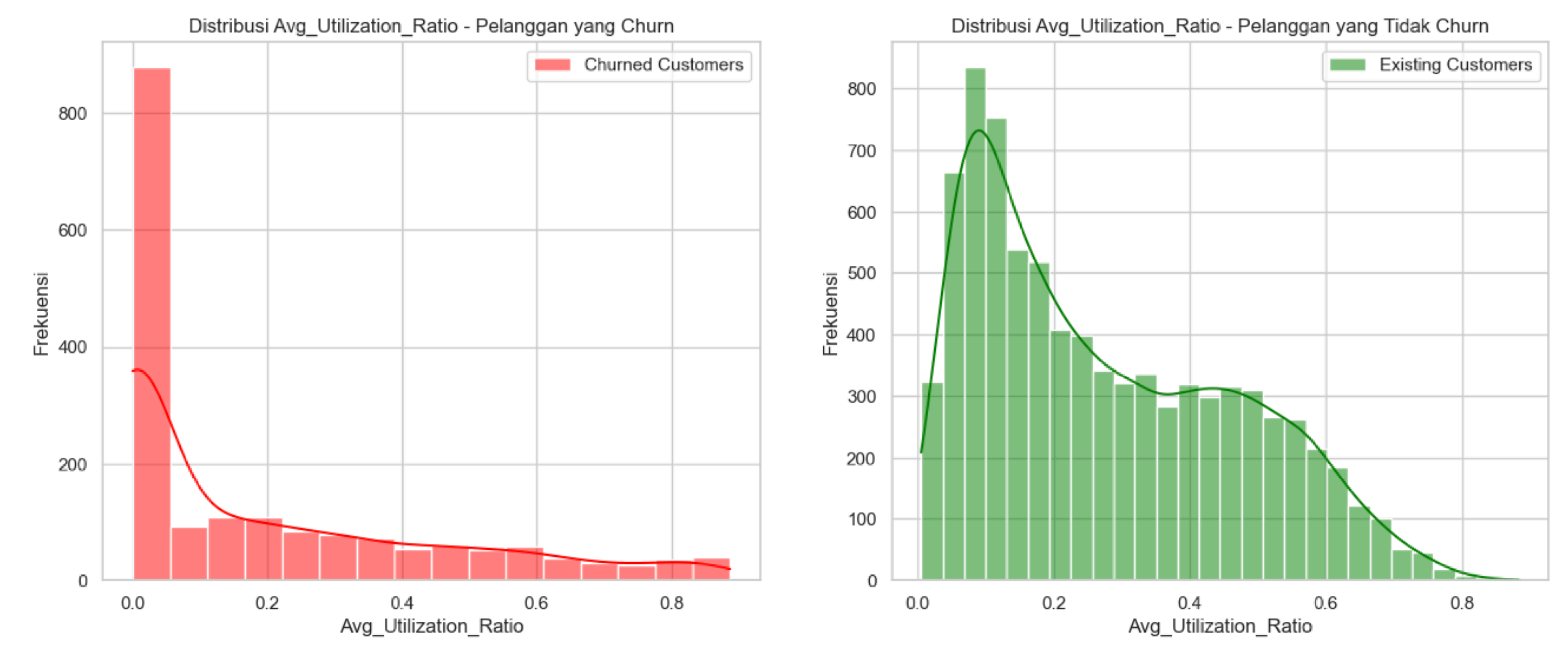
Insight

Distribusi Transaksi Customer





Distribusi Rasio Penggunaan Kartu



Strategi yang bisa dilakukan untuk Bank

- **Mengadakan event spesial untuk nasabah bank** agar transaksi customer meningkat, di Indonesia terdapat contoh seperti BCA Expo.
- **Memperluas layanan seperti pembayaran di banyak platform** belanja supaya meningkatkan keaktifan nasabah di layanan bank dan meningkatkan rasio penggunaan kartu kredit.
- **Memberikan program khusus untuk usia yang relatif berpotensi churn**, misal di studi kasus ini di usia tua yang tidak produktif kabanyakan churn. Program khusus ini bisa seperti pengolahan dana oleh wealth spesialist, investasi yang relevan untuk usia tua dan lain-lain.
- **meningkatkan komunikasi dan hubungan yang kuat dengan pelanggan yang lanjut usia atau yang berpotensi churn** dengan memberikan edukasi dan promosi yang sesuai dengan perilaku pelanggan dengan layanan yang ramah dan juga profesional.
- **memberikan produk atau layanan untuk nasabah bank yang lanjut usia yang relevan** seperti “Tabungan Pensiun” dengan memberikan bunga return yang tinggi dan juga kemudahan dalam mengatur dana pensiun.

Kesimpulan

Hasil pada pemodelan antara Decision Tree dan SVM untuk mengidentifikasi pelanggan yang potensial untuk churn menunjukkan bahwa keduanya berhasil memberikan performa yang baik, dengan Decision Tree sedikit lebih unggul dalam hal akurasi sekitar 93.13%, dibandingkan dengan 93.08% pada SVM. Namun, jika dilihat lebih dalam, bisa dilihat bahwa keduanya memiliki kelebihan masing-masing model. Decision Tree lebih baik dalam mengidentifikasi pelanggan yang benar-benar pindah dengan akurasi tinggi, sementara SVM memiliki kecenderungan yang lebih besar untuk mengidentifikasi pelanggan yang sebenarnya melakukan pindah (True Positives) dengan lebih baik daripada Decision Tree. Untuk meningkatkan kualitas prediksi, maka dapat dilakukan dengan cara menyempurnakan parameter-parameter model, seperti mengoptimalkan hyperparameter. Selain itu, mengeksplorasi fitur-fitur baru atau penyesuaian pada fitur yang sudah digunakan juga dapat memberikan hasil yang positif. Dalam pengembangannya juga dapat mempertimbangkan teknik augmentasi data untuk menangani ketidakseimbangan dalam dataset. Selain itu, dapat dilakukan juga implementasi cross validation dan interpretasi model akan membantu memahami bagaimana model membuat sebuah keputusan dan di mana kita dapat meningkatkan keakuratannya.

Setelah mengetahui beberapa faktor dari modelling classification tadi, Bank dapat mengimplementasikan berbagai strategi pemasaran untuk mencegah atau mengurangi churn rate dari nasabah yang berpotensi churn. Strategi-strategi tersebut meliputi mengadakan event spesial, memperluas layanan, memberikan program khusus, meningkatkan komunikasi dan hubungan, dan memberikan produk atau layanan yang relevan. Dengan strategi-strategi tersebut, bank dapat meningkatkan transaksi customer, menarik nasabah baru, memperkuat loyalitas nasabah lama, meningkatkan tingkat kepuasan dan loyalitas pelanggan, memberikan manfaat dan nilai tambah bagi nasabah, dan memberikan solusi finansial yang sesuai dengan kebutuhan dan preferensi nasabah.

Thank You