

Nama : Tirtayuda Munggarana

NIM : 1103202108

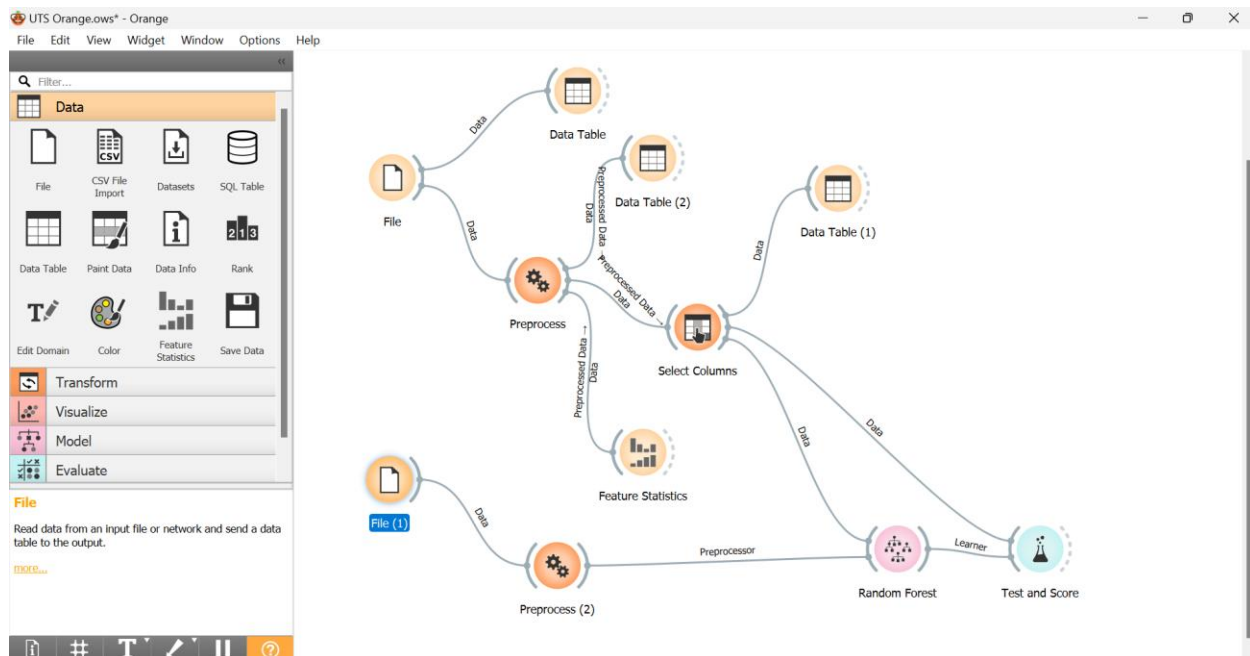
Kelas : TK-44-G04

UTS Machine Learning

Dataset Credit Score Classification

Diberikan informasi terkait kredit seseorang, buatlah model pembelajaran mesin yang dapat mengklasifikasikan skor kredit.

Dokumentasi



Train csv

File - Orange

Source

☒ File: Downloads\train.csv

☐ URL:

...

Reload








File Type

Automatically detect type

Info

100000 instances
14 features (1.2% missing values)
Data has no target variable.
14 meta attributes

Columns (Double click to edit)

	Name	Type	Role	Values
1	Month	 categorical	feature	April, August, February, January, July, June...
2	Occupation	 categorical	feature	Accountant, Architect, Developer, Doctor, ...
3	Monthly_Inhan...	 numeric	feature	
4	Num_Bank_Acc...	 numeric	feature	
5	Num_Credit_Card	 numeric	feature	
6	Interest_Rate	 numeric	feature	
7	Delay_from_due...	 numeric	feature	

Reset

Apply

Browse documentation datasets

100k

Data table sebelum preprocessing

Data Table - Orange

Info
100000 instances
13 features (1.3 % missing data)
Target with 7 values
14 meta attributes (2.4 % missing data)

Variables
☒ Show variable labels (if present)
☐ Visualize numeric values
☒ Color by instance classes

Selection
☒ Select full rows

Restore Original Order

☒ Send Automatically

	Payment_Behaviou	ID	Customer_ID	Name	Age
1	High_spent_Sm...	0x1602	CUS_0xd40	Aaron Maashoh	23
2	Low_spent_Larg...	0x1603	CUS_0xd40	Aaron Maashoh	23
3	Low_spent_Me...	0x1604	CUS_0xd40	Aaron Maashoh	-500
4	Low_spent_Sma...	0x1605	CUS_0xd40	Aaron Maashoh	23
5	High_spent_Me...	0x1606	CUS_0xd40	Aaron Maashoh	23
6	!@9#%8	0x1607	CUS_0xd40	Aaron Maashoh	23
7	Low_spent_Sma...	0x1608	CUS_0xd40	Aaron Maashoh	23
8	High_spent_Me...	0x1609	CUS_0xd40	?	23
9	Low_spent_Sma...	0x160e	CUS_0x21b1	Rick Rothackerj	28_
10	High_spent_Lar...	0x160f	CUS_0x21b1	Rick Rothackerj	28
11	High_spent_Lar...	0x1610	CUS_0x21b1	Rick Rothackerj	28
12	Low_spent_Me...	0x1611	CUS_0x21b1	Rick Rothackerj	28
13	Low_spent_Sma...	0x1612	CUS_0x21b1	Rick Rothackerj	28
14	High_spent_Lar...	0x1613	CUS_0x21b1	Rick Rothackerj	28
15	High_spent_Me...	0x1614	CUS_0x21b1	Rick Rothackerj	28
16	Low_spent_Sma...	0x1615	CUS_0x21b1	Rick Rothackerj	28
17	!@9#%8	0x161a	CUS_0x2dbc	Langep	34
18	High_spent_Sm...	0x161b	CUS_0x2dbc	?	34
19	High_spent_Sm...	0x161c	CUS_0x2dbc	Langep	34

100k 100k | 100k

Preprocessing

Preprocess - Orange

Preprocessors

- Discretize Continuous Variables
- Continuize Discrete Variables
- Impute Missing Values
- Select Relevant Features
- Select Random Features
- Normalize Features
- Randomize
- Remove Sparse Features
- Principal Component Analysis
- CUR Matrix Decomposition

Impute Missing Values

☐ Average/Most frequent
☐ Replace with random value
☒ Remove rows with missing values.

Normalize Features

☐ Standardize to $\mu=0$, $\sigma^2=1$
☐ Center to $\mu=0$
☐ Scale to $\sigma^2=1$
☐ Normalize to interval $[-1, 1]$
☒ Normalize to interval $[0, 1]$

☒ Apply Automatically

100k 83.3k

Data table sesudah dipreprocessing

Data Table (2) - Orange

Info
83316 instances
13 features
Target with 7 values
14 meta attributes (2.4 % missing data)

Variables
☒ Show variable labels (if present)
☐ Visualize numeric values
☒ Color by instance classes

Selection
☒ Select full rows

Restore Original Order

☒ Send Automatically

	Payment_Behaviour	ID	Customer_ID	Name	Age
1	High_spent_Sm...	0x1602	CUS_0xd40	Aaron Maashoh	23
2	High_spent_Me...	0x1606	CUS_0xd40	Aaron Maashoh	23
3	Low_spent_Sma...	0x1608	CUS_0xd40	Aaron Maashoh	23
4	High_spent_Me...	0x1609	CUS_0xd40	?	23
5	Low_spent_Sma...	0x160e	CUS_0x21b1	Rick Rothackerj	28
6	High_spent_Lar...	0x160f	CUS_0x21b1	Rick Rothackerj	28
7	High_spent_Lar...	0x1610	CUS_0x21b1	Rick Rothackerj	28
8	Low_spent_Sma...	0x1612	CUS_0x21b1	Rick Rothackerj	28
9	High_spent_Lar...	0x1613	CUS_0x21b1	Rick Rothackerj	28
10	Low_spent_Sma...	0x1615	CUS_0x21b1	Rick Rothackerj	28
11	!@9#%8	0x161a	CUS_0x2dbc	Langep	34
12	High_spent_Sm...	0x161b	CUS_0x2dbc	?	34
13	Low_spent_Me...	0x161d	CUS_0x2dbc	Langep	34
14	Low_spent_Larg...	0x161e	CUS_0x2dbc	Langep	34
15	High_spent_Me...	0x161f	CUS_0x2dbc	Langep	34
16	High_spent_Sm...	0x1620	CUS_0x2dbc	?	34
17	High_spent_Sm...	0x1621	CUS_0x2dbc	Langep	34
18	Low_spent_Larg...	0x1626	CUS_0xb891	Jasond	54
19	Low_spent_Sma...	0x1627	CUS_0xb891	Jasond	54

83.3k | 83.3k

Feature statistics sesudah di preprocessing

Feature Statistics - Orange

	Name	tribut	Mean	Mode	Median	Dispersion	Min.	Max.	Missi
N	Monthly_Inhan...		0.261036	0.133643	0.187419	0.818179	0.00	1	
N	Num_Bank_Acc...		0.009918	0.003891	0.003891	6.493477	0.00	1.0000	
N	Num_Credit_Card		0.014857	0.003336	0.004003	5.738480	0.00	1.0000	
N	Interest_Rate		0.012425	0.001208	0.002243	6.514259	0.00	1.0000	
N	Delay_from_due...		0.3626	0.2778	0.3194	0.5691	0.00	1.00	
N	Num_Credit_Inq...		0.0107216	0.0015402	0.0023104	6.9559705	0.00	1.00000	
N	Credit_Utilizatio...		0.409709	0.00	0.410823	0.416141	0.00	1	

Color: None

☒ Send Automatically

83.3k | 14

Pengambilan fitur – fitur penting untuk model

Select Columns - Orange

Ignored (5)

Filter

- C Occupation
- C Credit_Mix
- C Payment_of_Min_Amount
- C Month
- C Credit_Score

>

Features (8)

Filter

- N Monthly_Inhand_Salary
- N Num_Bank_Accounts
- N Num_Credit_Card
- N Interest_Rate
- N Delay_from_due_date
- N Num_Credit_Inquiries
- N Credit_Utilization_Ratio
- N Total_EMI_per_month

>

Target (1)

- C Payment_Behaviour

Metas (14)

- S ID
- S Customer_ID
- S Name
- S Age
- S SSN
- S Annual_Income
- S Num_of_Loan

Reset ☐ Ignore new variables by default ☒ Send Automatically

83.3k | 83.3k | 8

Random Forest

Random Forest - Orange

Name
Random Forest

Basic Properties

Number of trees: 10

☐ Number of attributes considered at each split: 5

☐ Replicable training

☐ Balance class distribution

Growth Control

☐ Limit depth of individual trees: 3

☒ Do not split subsets smaller than: 5

☒ Apply Automatically

83.3k | 83.3k | 1x83316

Test and score

Test and Score - Orange

☐ Cross validation
Number of folds: 10
☒ Stratified

☐ Cross validation by feature

☐ Random sampling
Repeat train/test: 10
Training set size: 66 %
☐ Stratified

☐ Leave one out

☒ Test on train data

☐ Test on test data

Evaluation results for target: (None, show average over classes)

Model	AUC	CA	F1	Prec	Recall	MCC
Random Forest	3.768	0.834	0.833	0.838	0.834	0.801

Compare models by: Area under ROC curve

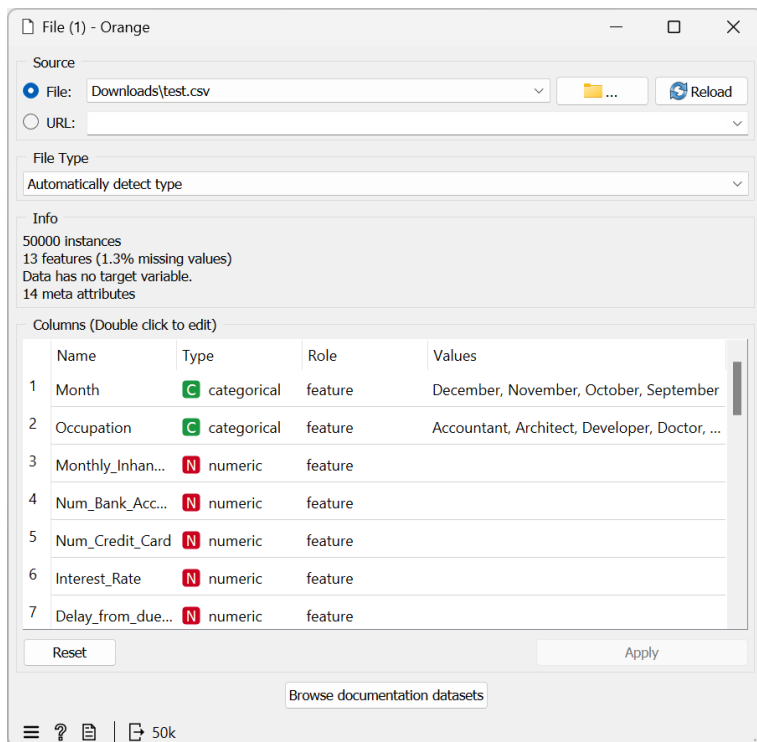
☐ Negligible diff.: 0.1

Model	Random ...
Random Forest	

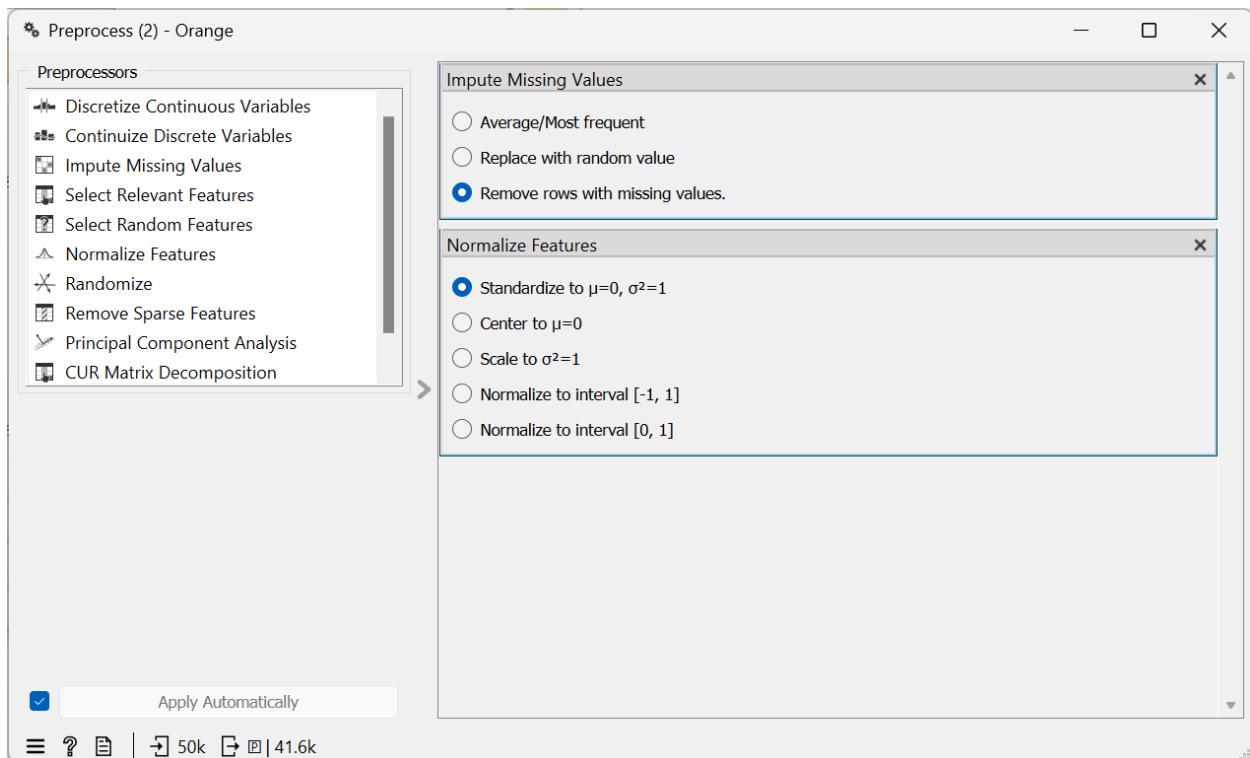
Table shows probabilities that the score for the model in the row is higher than that of the model in the column. Small numbers show the probability that the difference is negligible.

83.3k | 83.3k | 1x83316

Test CSV



Preprocessing



Google Colab

1. Import Library dan modul

```
import numpy as np # Mengimpor pustaka numpy
import pandas as pd # Mengimpor pustaka pandas
import plotly.express as px # Mengimpor modul express dari pustaka plotly
import plotly.graph_objects as go # Mengimpor modul graph pustaka plotly
import matplotlib.pyplot as plt # Mengimpor modul pyplot dari pustaka
matplotlib
import seaborn as sns # Mengimpor pustaka seaborn
from sklearn.preprocessing import OrdinalEncoder, LabelEncoder # Mengimpor
kelas OrdinalEncoder dan LabelEncoder dari modul preprocessing dalam
pustaka Scikit-learn
from sklearn.feature_selection import mutual_info_classif # Mengimpor
fungsi mutual_info_classif dari modul feature_selection dalam pustaka
Scikit-learn.
from sklearn.model_selection import train_test_split # Mengimpor fungsi
train_test_split dari modul model_selection dalam pustaka Scikit-learn
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
# Mengimpor kelas RandomForestClassifier dan RandomForestRegressor dari
modul ensemble dalam pustaka Scikit-learn
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score, mean_squared_error # Mengimpor fungsi-fungsi untuk
evaluasi model seperti classification_report, confusion_matrix,
accuracy_score untuk masalah klasifikasi, dan mean_squared_error untuk
masalah regresi dari modul metrics dalam pustaka Scikit-learn.
```

2. Menghubungkan ke Google Colab

```
## Load Dataset dari google drive
from google.colab import drive
drive.mount('/content/drive/')
```

3. Load dataset train dan test lalu mengubahnya menjadi dataframe

```
# load dataset train
train = pd.read_csv('/content/drive/MyDrive/DatasetML/train.csv',
dtype={'Column26': str})
# load dataset test
test = pd.read_csv('/content/drive/MyDrive/DatasetML/test.csv',
dtype={'Column26': str})
```


4. Menampilkan informasi terkait dataframe seperti tipe data

```
# untuk mendapatkan informasi mengenai struktur dan tipe data
train.info()
```

output :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 28 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   ID                                     100000 non-null  object
1   Customer_ID                           100000 non-null  object
2   Month                                 100000 non-null  object
3   Name                                  90015 non-null   object
4   Age                                   100000 non-null  object
5   SSN                                   100000 non-null  object
6   Occupation                             100000 non-null  object
7   Annual_Income                          100000 non-null  object
8   Monthly_Inhand_Salary                  84998 non-null   float64
9   Num_Bank_Accounts                      100000 non-null  int64
10  Num_Credit_Card                         100000 non-null  int64
11  Interest_Rate                           100000 non-null  int64
12  Num_of_Loan                             100000 non-null  object
13  Type_of_Loan                             88592 non-null   object
14  Delay_from_due_date                     100000 non-null  int64
15  Num_of_Delayed_Payment                  92998 non-null   object
16  Changed_Credit_Limit                   100000 non-null  object
17  Num_Credit_Inquiries                    98035 non-null   float64
18  Credit_Mix                             100000 non-null  object
19  Outstanding_Debt                       100000 non-null  object
20  Credit_Utilization_Ratio                100000 non-null  float64
21  Credit_History_Age                      90970 non-null   object
22  Payment_of_Min_Amount                   100000 non-null  object
23  Total_EMI_per_month                     100000 non-null  float64
24  Amount_invested_monthly                 95521 non-null   object
25  Payment_Behaviour                       100000 non-null  object
26  Monthly_Balance                         98800 non-null   object
27  Credit_Score                           100000 non-null  object
dtypes: float64(4), int64(4), object(20)
memory usage: 21.4+ MB
```

5. Data Formatting

```
train['Age'] =
train['Age'].fillna('0').str.extract('(\d+)').astype(float).astype(int)
train['Num_of_Loan'] =
train['Num_of_Loan'].fillna('0').str.extract('(\d+)').astype(float).
astype(int)
train['Num_of_Delayed_Payment'] =
train['Num_of_Delayed_Payment'].fillna('0').str.extract('(\d+)').ast
ype(float).astype(int)
```

- Kolom Age

- fillna('0'): Mengganti nilai yang hilang (NaN) dalam kolom 'Age' dengan string '0'.
- str.extract('(\d+)'): Mengekstraksi bagian numerik dari setiap nilai dalam kolom 'Age'.
- astype(float): Mengonversi nilai-nilai tersebut menjadi float.
- astype(int): Mengonversi float tersebut menjadi integer.

- Kolom Num Of Loan
 - Operasi yang dilakukan pada kolom 'Num_of_Loan' mirip dengan operasi pada kolom 'Age'. Nilai yang hilang diisi dengan '0', kemudian bagian numerik diekstraksi, dikonversi menjadi float, dan akhirnya menjadi integer.
- Kolom Num of Delayed Payment
 - Operasi yang dilakukan pada kolom 'Num_of_Delayed_Payment' juga mirip dengan operasi pada dua kolom sebelumnya. Nilai yang hilang diisi dengan '0', kemudian bagian numerik diekstraksi, dikonversi menjadi float, dan akhirnya menjadi integer.

6. Annual Income

```
train['Annual_Income'] = train['Annual_Income'].str.replace(r'^0-9.', '', regex=True)
train['Annual_Income'] = train['Annual_Income'].astype(float)
```

Kode ini secara efektif membersihkan kolom 'Annual_Income' dari karakter non-numerik dan mengonversi nilainya menjadi tipe data float, yang kemungkinan besar merupakan langkah persiapan data sebelum dilakukan analisis lebih lanjut atau pemodelan.

7. Change Credit Limit

```
train['Changed_Credit_Limit'] =
train['Changed_Credit_Limit'].replace('_', np.nan)
train['Changed_Credit_Limit'] =
pd.to_numeric(train['Changed_Credit_Limit'], errors='coerce')
train['Changed_Credit_Limit'] =
train['Changed_Credit_Limit'].fillna(0)
```

Secara keseluruhan, kode ini bertujuan untuk membersihkan kolom 'Changed_Credit_Limit' dari nilai yang tidak valid atau tidak berguna, seperti garis bawah ('_'), dan menggantikan nilai yang hilang dengan nilai 0. Hal ini kemungkinan dilakukan sebagai bagian dari persiapan data sebelum analisis lebih lanjut atau pemodelan.

8. Outstanding Debt

```
train['Outstanding_Debt'] = train['Outstanding_Debt'].astype(str)
train['Outstanding_Debt'] =
train['Outstanding_Debt'].str.replace(r'^0-9.', '', regex=True)
train['Outstanding_Debt'] = pd.to_numeric(train['Outstanding_Debt'],
errors='coerce')
train['Outstanding_Debt'] = train['Outstanding_Debt'].fillna(0)
```

Secara keseluruhan, kode ini bertujuan untuk membersihkan kolom 'Outstanding_Debt' dari karakter non-numerik, mengonversi nilainya menjadi tipe data numerik, dan mengganti nilai-nilai yang hilang dengan nilai 0. Hal ini biasanya dilakukan sebagai langkah persiapan data sebelum analisis lebih lanjut atau pemodelan.

9. Invested Monthly

```

train['Amount_invested_monthly'] =
train['Amount_invested_monthly'].astype(str)
train['Amount_invested_monthly'] =
train['Amount_invested_monthly'].replace(' ', '0')
train['Amount_invested_monthly'] =
train['Amount_invested_monthly'].str.replace(r'^0-9.', '')
train['Amount_invested_monthly'] =
pd.to_numeric(train['Amount_invested_monthly'], errors='coerce')
train['Amount_invested_monthly'] =
train['Amount_invested_monthly'].fillna(0)

```

Secara keseluruhan, kode ini memiliki tujuan yang serupa dengan kode sebelumnya, yaitu membersihkan kolom 'Amount_invested_monthly' dari karakter non-numerik, mengonversi nilainya menjadi tipe data numerik, dan mengganti nilai-nilai yang hilang dengan nilai 0. Hal ini biasanya dilakukan sebagai langkah persiapan data sebelum analisis lebih lanjut atau pemodelan.

10. Monthly Balance

```

train['Monthly_Balance'] = train['Monthly_Balance'].astype(str)
train['Monthly_Balance'] =
train['Monthly_Balance'].str.replace(r'^0-9.-]+', '')
train['Monthly_Balance'] = pd.to_numeric(train['Monthly_Balance'],
errors='coerce')
train['Monthly_Balance'] = train['Monthly_Balance'].fillna(0)

```

Kode ini bertujuan untuk membersihkan kolom 'Monthly_Balance' dari karakter non-numerik, mengonversi nilainya menjadi tipe data numerik, dan mengganti nilai-nilai yang hilang dengan nilai 0. Hal ini biasanya dilakukan sebagai langkah persiapan data sebelum analisis lebih lanjut atau pemodelan.

11. Credit History Age

```

def parse_years_and_months(age):
    if isinstance(age, str):
        age_parts = age.split(' Years and ')
        years = int(age_parts[0]) if 'Years' in age else 0
        months_str = age_parts[1].split(' Months')[0] if 'Months' in
age_parts[1] else '0'
        months = int(months_str)
        total_months = years * 12 + months
        return total_months
    else:
        return 0

train['Credit_History_Age_Months'] =
train['Credit_History_Age'].apply(parse_years_and_months)

```

Fungsi diatas yaitu bertujuan untuk mengkonversi umur dalam format years and months menjadi jumlah bulan

12. Duplicate

```
duplicates = train[train.duplicated()]
num_duplicates = duplicates.shape[0]

if num_duplicates == 0:
    print("Tidak ada nilai yang duplikasi")
else:
    print("ada nilai ", num_duplicates, "duplicates.")
```

output :

```
➡ Tidak ada nilai yang duplikasi 🙌
```

jadi, kode tersebut akan mencetak jumlah duplikat yang ditemukan dalam DataFrame train. Jika tidak ada duplikat yang ditemukan, akan mencetak pesan bahwa tidak ada nilai yang duplikat.

13. Data Scaling

```
train.describe().T
```

output :

	count	mean	std	min	25%	50%	75%	max
Age	100000.0	119.509700	6.847573e+02	14.000000	25.000000	34.000000	42.000000	8.698000e+03
Annual_Income	100000.0	176415.701298	1.429618e+06	7005.930000	19457.500000	37578.610000	72790.920000	2.419806e+07
Monthly_Inhand_Salary	84998.0	4194.170850	3.183686e+03	303.645417	1625.568229	3093.745000	5957.448333	1.520463e+04
Num_Bank_Accounts	100000.0	17.091280	1.174048e+02	-1.000000	3.000000	6.000000	7.000000	1.798000e+03
Num_Credit_Card	100000.0	22.474430	1.290574e+02	0.000000	4.000000	5.000000	7.000000	1.499000e+03
Interest_Rate	100000.0	72.466040	4.664226e+02	1.000000	8.000000	13.000000	20.000000	5.797000e+03
Num_of_Loan	100000.0	10.761960	6.178993e+01	0.000000	2.000000	3.000000	6.000000	1.496000e+03
Delay_from_due_date	100000.0	21.068780	1.486010e+01	-5.000000	10.000000	18.000000	28.000000	6.700000e+01
Num_of_Delayed_Payment	100000.0	28.779410	2.181148e+02	0.000000	8.000000	13.000000	18.000000	4.397000e+03
Changed_Credit_Limit	100000.0	10.171791	6.880628e+00	-6.490000	4.970000	9.250000	14.660000	3.697000e+01
Num_Credit_Inquiries	98035.0	27.754251	1.931773e+02	0.000000	3.000000	6.000000	9.000000	2.597000e+03
Outstanding_Debt	100000.0	1426.220376	1.155129e+03	0.230000	566.072500	1166.155000	1945.962500	4.998070e+03
Credit_Utilization_Ratio	100000.0	32.285173	5.116875e+00	20.000000	28.052567	32.305784	36.496663	5.000000e+01
Total_EMI_per_month	100000.0	1403.118217	8.306041e+03	0.000000	30.306660	69.249473	161.224249	8.233100e+04
Amount_invested_monthly	100000.0	178.363270	1.984724e+02	0.000000	58.325837	116.545252	220.039055	1.977326e+03
Monthly_Balance	100000.0	397.684413	2.171320e+02	0.000000	267.871374	334.806633	467.670597	1.602041e+03
Credit_History_Age_Months	100000.0	201.221460	1.143207e+02	0.000000	114.000000	208.000000	292.000000	4.040000e+02

Kode `train.describe().T` digunakan untuk menghasilkan ringkasan statistik deskriptif dari DataFrame train, yang akan ditransposisikan (diputar) sehingga fitur-fitur (kolom-kolom) menjadi indeks baris.

14. Remove outlier

```
# Removing ridiculous top outliers (2%)
```

```

selected_columns_train = train[['Num_Bank_Accounts',
'Interest_Rate', 'Annual_Income', 'Num_of_Delayed_Payment',
'Num_Credit_Inquiries', 'Total_EMI_per_month', 'Num_of_Loan',
'Num_Credit_Card']]

percentile_threshold = 0.98
percentiles = selected_columns_train.quantile(percentile_threshold)

for column in selected_columns_train.columns:
    train = train[train[column] <= percentiles[column]]

```

tujuan dari kode ini adalah untuk membersihkan dataset train dari outlier yang berada di bagian atas (2%) dari kolom-kolom yang telah dipilih. Ini dapat membantu meningkatkan kualitas data dan hasil analisis

15. Data Entry

```

# Menghapus nilai yang tidak jelas pada kolom payment behavior
train = train[train['Payment_Behaviour'] != '!@9#%8']

```

Tujuannya adalah untuk membersihkan dataset dari nilai yang tidak jelas atau anomali dalam kolom 'Payment_Behaviour', yang kemungkinan besar tidak berguna atau tidak bermakna dalam analisis data.

16. Drop Sebagian data di kolom pekerjaan

```

# Drop the rows containing the value '_____' in the 'Occupation'
column

train = train[train['Occupation'] != '_____']
print(train['Occupation'].unique())

```

Kode di atas bertujuan untuk menghapus baris-baris yang mengandung nilai '_____' dalam kolom 'Occupation' dari dataset train, dan kemudian mencetak nilai unik dari kolom 'Occupation' yang tersisa. Mari kita jelaskan langkah-langkahnya:

17. Drop baris pada data yang memiliki “_” pada kolom credit mix

```

# Drop rows where 'Credit_Mix' is '_'

train = train[train['Credit_Mix'] != '_']
print(train['Credit_Mix'].unique())

```

18. Mengatasi nilai negative

```

# Negative values
# Menghapus baris yang mengandung nilai negatif dalam kolom yang
sudah dipilih
selected_columns = ['Delay_from_due_date', 'Changed_Credit_Limit',
'Num_Bank_Accounts']

for column in selected_columns:

```

```
train = train[train[column] >= 0]
```

Tujuannya adalah untuk membersihkan dataset dari nilai yang tidak valid atau tidak relevan dalam kolom-kolom yang telah dipilih, dalam hal ini, nilai negatif. Setelah penghapusan, kita dapat memastikan bahwa hanya nilai-nilai yang valid dan relevan yang tersisa dalam kolom-kolom yang dipilih.

19. Drop kolom

```
# Drop kolom yang tidak berguna untuk training model

columns_to_drop = ['ID', 'Customer_ID', 'Month', 'Name', 'SSN',
                  'Credit_History_Age', 'Monthly_Inhand_Salary', 'Type_of_Loan']

train.drop(columns=columns_to_drop, inplace=True)
```

Tujuannya adalah untuk membersihkan dataset dari kolom-kolom yang tidak relevan atau tidak diperlukan untuk proses pelatihan model. Setelah penghapusan, kita dapat memastikan bahwa dataset hanya berisi kolom-kolom yang diperlukan untuk proses analisis atau pemodelan selanjutnya.

20. Mengatasi nilai yang hilang

```
total_missing_values = train.isnull().sum().sum()

if total_missing_values == 0:
    print("There are no missing values")
else:
    print("Total missing values:", total_missing_values)
```

There are no missing values 🎉

output :

Tujuannya adalah untuk memberikan informasi tentang keberadaan atau ketiadaan nilai yang hilang dalam dataset train. Jika tidak ada nilai yang hilang, akan dicetak pesan yang menunjukkan bahwa tidak ada nilai yang hilang. Jika ada nilai yang hilang, akan dicetak total jumlah nilai yang hilang.

21. Feature Engineering

```
# Mengambil semua kolom yang bertipe data int64 dan float64 dan
# diinputkan ke variabel numeric columns
numeric_columns = train.select_dtypes(include=['int64',
                                              'float64']).columns

# Jumlah kolom yang ingin ditampilkan
num_columns = 8

# Jumlah baris yang ditampilkan sesuai dengan jumlah kolom yang
# ditentukan diatas
```

```

num_rows = (len(numeric_columns) + num_columns - 1) // num_columns

# Ukuran subplot
fig, axes = plt.subplots(num_rows, num_columns, figsize=(16, 6))

axes = axes.flatten()

# Loop untuk membuat boxplot
for i, column in enumerate(numeric_columns):
    sns.boxplot(x=train[column], ax=axes[i])
    axes[i].set_title(column, fontsize=7)
    axes[i].set_xlabel('Value', fontsize=7)
    axes[i].set_ylabel('Count', fontsize=7)

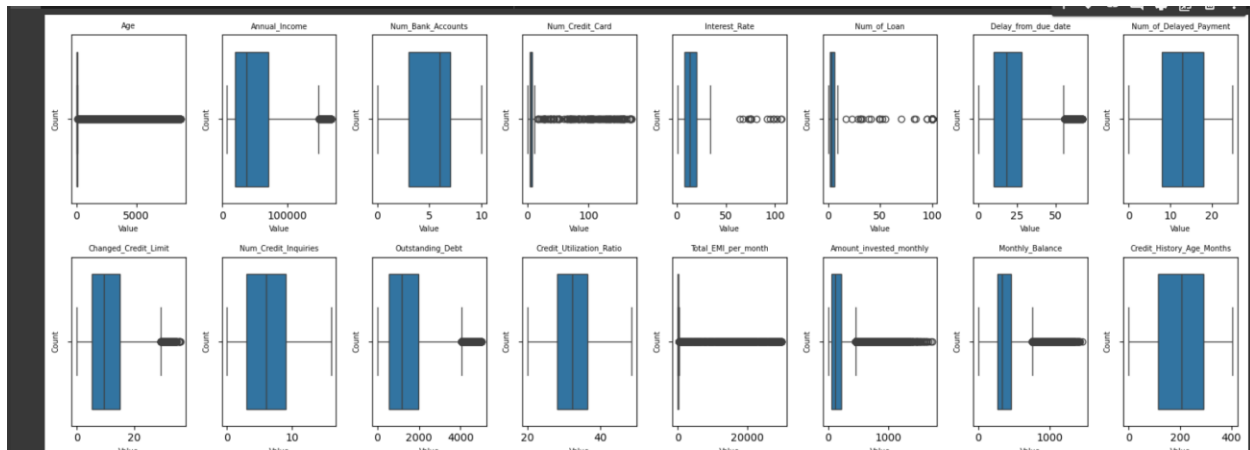
# Loop untuk menyembunyikan subplot yang tidak digunakan
for j in range(len(numeric_columns), num_columns*num_rows):
    axes[j].axis('off')

# Menata ulang tata letak plot agar sesuai
plt.tight_layout()

# Menampilkan Plot
plt.show()

```

output :



Tujuannya adalah untuk memvisualisasikan distribusi dan pola data dalam setiap kolom numerik melalui boxplot. Dengan menggunakan subplot, boxplot untuk setiap kolom numerik dapat ditampilkan dalam satu gambar dengan tata letak yang terorganisir.

22. Scaling

```

# Menskalakan pada tiap kolom dengan batasan yang sudah ditentukan
train = train[train['Age'] < 60]

```

```

train = train[train['Num_Credit_Card'] <= 10]
train = train[train['Interest_Rate'] <= 50]
train = train[train['Num_of_Loan'] <= 12]
train = train[train['Num_Bank_Accounts'] <= 10]
train = train[train['Delay_from_due_date'] <= 60]
train = train[train['Changed_Credit_Limit'] <= 30]
train = train[train['Num_Credit_Inquiries'] <= 12]
train = train[train['Total_EMI_per_month'] <= 200]
train = train[train['Outstanding_Debt'] <= 1500]

```

Setelah diterapkan batasan tersebut, baris-baris yang melanggar batasan tersebut akan dihapus dari dataset train. Tujuannya adalah untuk memastikan bahwa data tetap dalam rentang yang wajar dan sesuai dengan kondisi yang diharapkan.

23. Encoding

```

# Label Encoder
categories = ['Poor', 'Standard', 'Good']

encoder = OrdinalEncoder(categories=[categories])

train['Credit_Score_Encoded'] =
encoder.fit_transform(train[['Credit_Score']])

```

Dengan menggunakan OrdinalEncoder, kategori 'Poor' akan dikodekan menjadi 0, 'Standard' menjadi 1, dan 'Good' menjadi 2. Ini memungkinkan penggunaan data kategori dalam analisis atau pemodelan yang membutuhkan input numerik.

24. Encoding Occupation

```

# Encoding Occupation
label_encoder = LabelEncoder()
train['Occupation_Encoded'] =
label_encoder.fit_transform(train['Occupation'])

```

Dengan menggunakan LabelEncoder, setiap nilai unik dalam kolom 'Occupation' akan dikodekan menjadi nilai numerik secara otomatis. Ini memungkinkan penggunaan data kategori dalam analisis atau pemodelan yang membutuhkan input numerik.

25. Ordinal Encoder

```

# Ordinal Encoder
categories = ['Bad', 'Standard', 'Good']

encoder = OrdinalEncoder(categories=[categories])

train['Credit_Mix_Encoded'] =
encoder.fit_transform(train[['Credit_Mix']])

```


Dengan menggunakan OrdinalEncoder, kategori 'Bad' akan dikodekan menjadi 0, 'Standard' menjadi 1, dan 'Good' menjadi 2. Ini memungkinkan penggunaan data kategori dalam analisis atau pemodelan yang membutuhkan input numerik.

```
categories_payment_behaviour = [  
    'Low_spent_Small_value_payments',  
    'Low_spent_Medium_value_payments',  
    'Low_spent_Large_value_payments',  
    'High_spent_Small_value_payments',  
    'High_spent_Medium_value_payments',  
    'High_spent_Large_value_payments'  
]  
  
encoder_payment_behaviour =  
OrdinalEncoder(categories=[categories_payment_behaviour])  
  
train['Payment_Behaviour_Encoded'] =  
encoder_payment_behaviour.fit_transform(train[['Payment_Behaviour']])
```

Dengan menggunakan OrdinalEncoder, kategori-kategori yang telah ditentukan akan dikodekan menjadi angka ordinal. Ini memungkinkan penggunaan data kategori dalam analisis atau pemodelan yang membutuhkan input numerik.

26. Dropping Unencoded Columns

```
columns_to_drop = [ 'Payment_Behaviour', 'Credit_Mix',  
    'Occupation', 'Credit_Score']  
train.drop(columns=columns_to_drop, inplace=True)
```

Tujuannya adalah untuk membersihkan dataset dari kolom-kolom yang tidak diperlukan atau tidak relevan untuk analisis atau pemodelan selanjutnya. Setelah penghapusan, dataset train akan berisi hanya kolom-kolom yang relevan untuk proses selanjutnya.

```
# Calculate the total number of accounts (Bank Accounts + Credit Cards)  
  
train['Total_Num_Accounts'] = train['Num_Bank_Accounts'] +  
train['Num_Credit_Card']  
  
# Calculate the total outstanding debt per account  
  
train['Debt_Per_Account'] = train['Outstanding_Debt'] /  
train['Total_Num_Accounts']  
  
# Calculate the ratio of outstanding debt to annual income
```

```

train['Debt_to_Income_Ratio'] = train['Outstanding_Debt'] /
train['Annual_Income']

# Calculate the total number of delayed payments per account

train['Delayed_Payments_Per_Account'] = train['Num_of_Delayed_Payment'] /
train['Total_Num_Accounts']

# Calculate the total monthly expenses (EMI + Monthly Investments)

train['Total_Monthly_Expenses'] = train['Total_EMI_per_month'] +
train['Amount_invested_monthly']

```

27. Mutual Information Score

```

categorical_columns =
train.select_dtypes(include=['object']).columns

data_encoded = train.copy()

encoder = OrdinalEncoder()
data_encoded[categorical_columns] =
encoder.fit_transform(data_encoded[categorical_columns])

y = data_encoded['Credit_Score_Encoded']
X = data_encoded.drop(columns=['Credit_Score_Encoded'])

mi_scores = mutual_info_classif(X, y)

for i, score in enumerate(mi_scores):
    print(f"Feature '{X.columns[i]}': Mutual Information Score =
{score}")

```

Output :

```

Feature 'Age': Mutual Information Score = 0.008068851664653653
Feature 'Annual_Income': Mutual Information Score = 0.4284873379299763
Feature 'Num_Bank_Accounts': Mutual Information Score = 0.05815364544646062
Feature 'Num_Credit_Card': Mutual Information Score = 0.07049223428645313
Feature 'Interest_Rate': Mutual Information Score = 0.10119734895144439
Feature 'Num_of_Loan': Mutual Information Score = 0.01797484601916266
Feature 'Delay_from_due_date': Mutual Information Score = 0.07049624263412979
Feature 'Num_of_Delayed_Payment': Mutual Information Score = 0.050432053015048384
Feature 'Changed_Credit_Limit': Mutual Information Score = 0.10319230440962235
Feature 'Num_Credit_Inquiries': Mutual Information Score = 0.03197765297492983
Feature 'Outstanding_Debt': Mutual Information Score = 0.4299239010979161
Feature 'Credit_Utilization_Ratio': Mutual Information Score = 0.000894060062685711
Feature 'Payment_of_Min_Amount': Mutual Information Score = 0.06680892845199238
Feature 'Total_EMI_per_month': Mutual Information Score = 0.3502706736702732
Feature 'Amount_invested_monthly': Mutual Information Score = 0.004089937026318324
Feature 'Monthly_Balance': Mutual Information Score = 0.004956969851415138
Feature 'Credit_History_Age_Months': Mutual Information Score = 0.02029470616539708
Feature 'Occupation_Encoded': Mutual Information Score = 0.0
Feature 'Credit_Mix_Encoded': Mutual Information Score = 0.18435896029176524
Feature 'Payment_Behaviour_Encoded': Mutual Information Score = 0.0028136568072885115
Feature 'Total_Num_Accounts': Mutual Information Score = 0.07611153445857588
Feature 'Debt_Per_Account': Mutual Information Score = 0.4306556137849169
Feature 'Debt_to_Income_Ratio': Mutual Information Score = 0.4336673708746479
Feature 'Delayed_Payments_Per_Account': Mutual Information Score = 0.05346183962411333
Feature 'Total_Monthly_Expenses': Mutual Information Score = 0.004287649821107076

```

Tujuannya adalah untuk mengevaluasi pentingnya setiap fitur dalam memprediksi variabel target 'Credit_Score_Encoded' menggunakan skor informasi mutual. Semakin tinggi skor informasi mutual, semakin penting fitur tersebut dalam prediksi variabel target.

```
sorted_mi_scores = sorted(zip(X.columns, mi_scores), key=lambda x: x[1],
reverse=True)
sorted_columns = [x[0] for x in sorted_mi_scores]
sorted_scores = [x[1] for x in sorted_mi_scores]

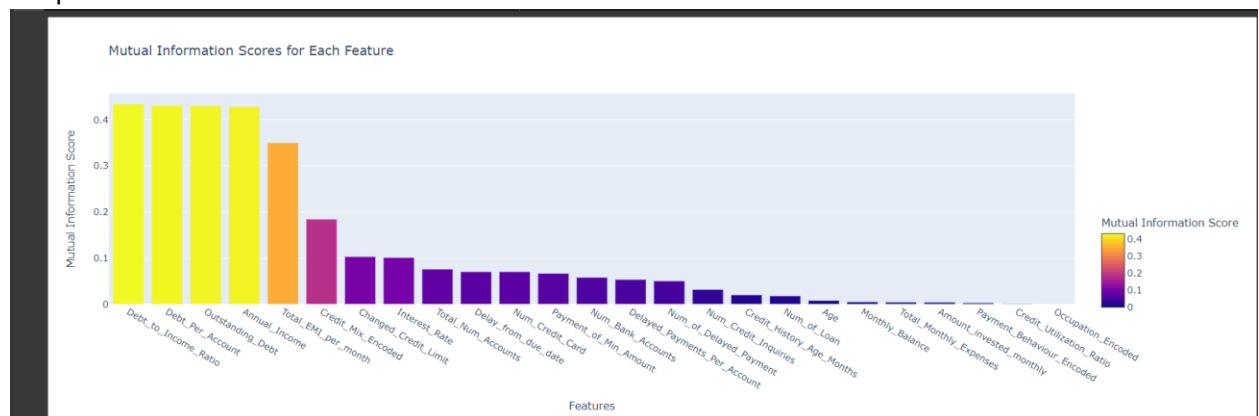
colorscale = 'Viridis'

fig = go.Figure(data=[go.Bar(x=sorted_columns, y=sorted_scores,
marker=dict(color=sorted_scores, colorbar=dict(title='Mutual Information
Score', len=0.5, y=0.2))))

fig.update_layout(title='Mutual Information Scores for Each Feature',
axis_title='Features',
axis_title='Mutual Information Score')

fig.show()
```

Output :



Kode di atas bertujuan untuk membuat diagram batang yang menampilkan skor informasi mutual untuk setiap fitur dalam dataset. Tujuannya adalah untuk secara visual menampilkan skor informasi mutual untuk setiap fitur dalam dataset, dengan fitur-fitur yang paling informatif di bagian atas plot. Ini membantu dalam pemahaman yang lebih baik tentang pentingnya masing-masing fitur dalam memprediksi variabel target.

28. Correlation Matrix

```
# Calculate Correlation Matrix

corr = train.select_dtypes(include=['float64',
'int64']).corr(method='pearson')
mask = np.triu(np.ones_like(corr, dtype=bool))
```

```
plt.figure(figsize=(16, 12))
sns.heatmap(corr, mask=mask, vmax=0.9, square=True, annot=True)
plt.title('Correlation Matrix')
plt.show()
```

Visualisasi ini membantu memahami hubungan antara berbagai fitur numerik dalam dataset. Korelasi yang kuat (baik positif maupun negatif) antara fitur-fitur dapat menunjukkan adanya multicollinearity potensial, yang dapat memengaruhi kinerja beberapa model pembelajaran mesin. Mengidentifikasi korelasi semacam itu penting untuk seleksi fitur dan pembangunan model.

29. Boxplot visualization

```
numeric_columns = train.select_dtypes(include=['int64',
'float64']).columns

num_columns = 8
num_rows = (len(numeric_columns) + num_columns - 1) // num_columns

fig, axes = plt.subplots(num_rows, num_columns, figsize=(16, 6))

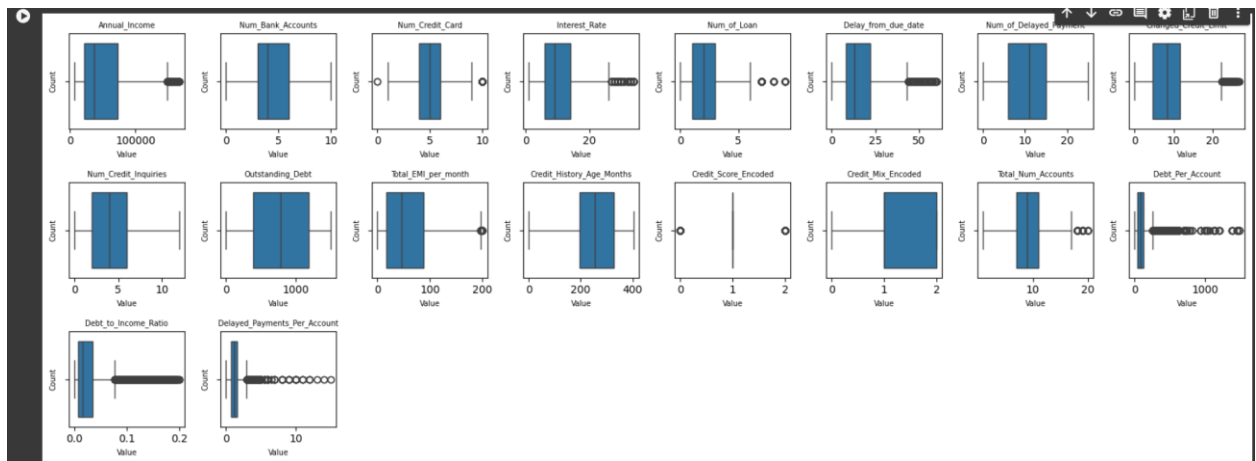
axes = axes.flatten()

for i, column in enumerate(numeric_columns):
    sns.boxplot(x=train[column], ax=axes[i])
    axes[i].set_title(column, fontsize=7)
    axes[i].set_xlabel('Value', fontsize=7)
    axes[i].set_ylabel('Count', fontsize=7)

for j in range(len(numeric_columns), num_columns*num_rows):
    axes[j].axis('off')

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

Output :



30. Membagi dataset

```
# Menentukan target yang hal ini credit score
y = train['Credit_Score_Encoded']

# Menentukan fitur apa saja yang berguna dalam membuat model
X = train[['Annual_Income', 'Num_Bank_Accounts', 'Num_Credit_Card',
           'Interest_Rate', 'Num_of_Loan', 'Delay_from_due_date',
           'Num_of_Delayed_Payment', 'Changed_Credit_Limit',
           'Num_Credit_Inquiries', 'Outstanding_Debt',
           'Total_EMI_per_month',
           'Credit_History_Age_Months', 'Credit_Mix_Encoded',
           'Total_Num_Accounts',
           'Debt_Per_Account', 'Debt_to_Income_Ratio',
           'Delayed_Payments_Per_Account']]

# Membagi dataset menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2, random_state=77)
```

31. Random forest regressor

```
model = RandomForestRegressor(n_estimators=500, bootstrap=True,
                              random_state=77)
model.fit(X_train, y_train)
```

output:



RandomForestRegressor

RandomForestRegressor(n_estimators=500, random_state=77)

32. MSE

```
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

Output :

```
Mean Squared Error: 0.1621489922101923
```

33. Accuracy

```
rf_classifier = RandomForestClassifier(n_estimators=500,
bootstrap=True)
rf_classifier.fit(X_train, y_train)
```

```
y_pred = rf_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy on original test set:", accuracy)

matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 6))
sns.heatmap(matrix, annot=True, cbar=False, cmap='twilight',
linewidth=0.5, fmt="d")
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix for RandomForestClassifier on original test
set')

print('\nClassification report for original test set:\n',
classification_report(y_test, y_pred))
```

Output :

```
➡ Accuracy on original test set: 0.8094852703140175

Classification report for original test set:
              precision    recall  f1-score   support

     0.0         0.75      0.67      0.71       901
     1.0         0.84      0.87      0.85      3810
     2.0         0.77      0.74      0.75      1467

 accuracy          0.81          0.81          0.81       6178
 macro avg         0.79          0.76          0.77       6178
 weighted avg         0.81          0.81          0.81       6178
```



Confusion Matrix for RandomForestClassifier on original test set

