

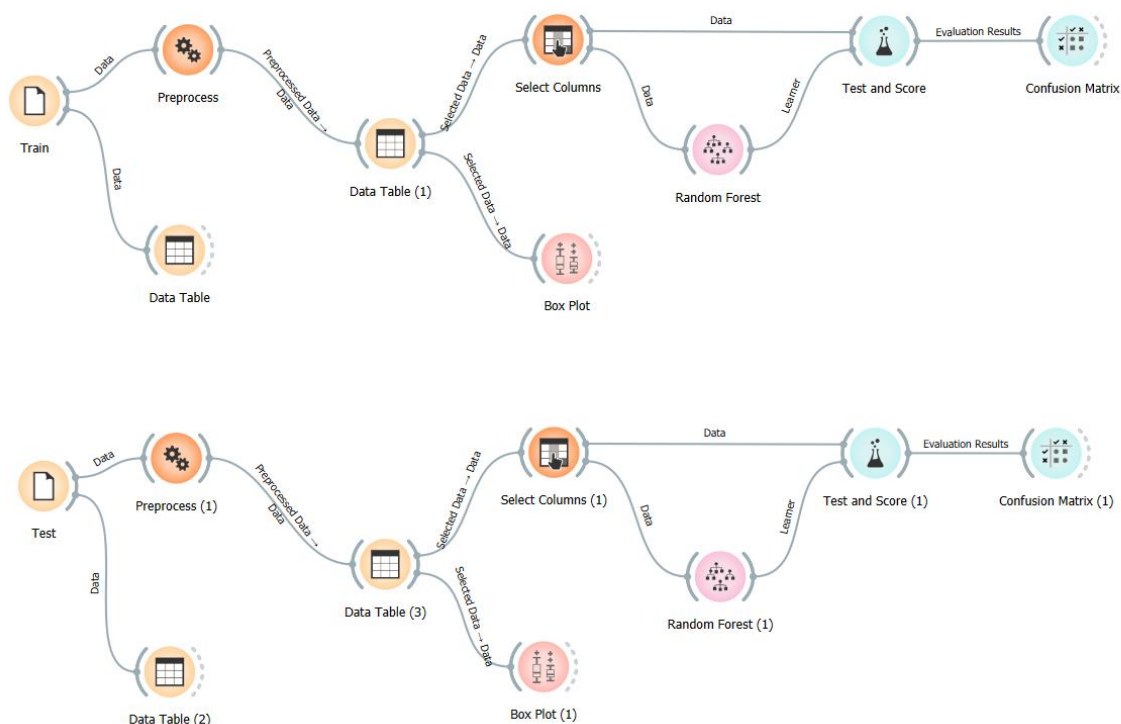
Nama : Mochamad Phillia Wibowo
NIM : 1103204191
Kelas : TK-44-G04
Model : Random Forest
Dataset : <https://www.kaggle.com/datasets/parisrohan/credit-score-classification/download?datasetVersionNumber=1>

UTS Machine Learning : Report Presentation

- About Dataset :
 - Pernyataan Masalah
Anda bekerja sebagai ilmuwan data di sebuah perusahaan keuangan global. Selama bertahun-tahun, perusahaan telah mengumpulkan detail bank dasar dan mengumpulkan banyak informasi terkait kredit. Manajemen ingin membangun sebuah sistem cerdas untuk memisahkan orang-orang ke dalam kelompok skor kredit untuk mengurangi upaya manual.
 - Tugas
Diberikan informasi terkait kredit seseorang, buatlah sebuah model pembelajaran mesin yang dapat mengklasifikasikan skor kredit.

Orange Data Mining :

Dokumentasi :



Train.csv

The screenshot shows the 'Train' widget in the Orange Data Mining software. The 'Source' tab is selected, showing the file 'Credit score classification/train.csv'. The 'File Type' is set to 'Automatically detect type'. The 'Info' section displays: 100000 instances, 14 features (1.2% missing values), Data has no target variable, and 14 meta attributes. The 'Columns' table is as follows:

Name	Type	Role	Values
Delay_from_due...	numeric	feature	
Num_Credit_Inq...	numeric	feature	
Credit_Mix	categorical	feature	Bad, Good, Standard, _
Credit_Utilizatio...	numeric	feature	
Payment_of_Mi...	categorical	feature	NM, No, Yes
Total_EMI_per_...	numeric	feature	
Payment_Behav...	categorical	feature	!@9#%8, High_spent_Large_value_payments, High_spent_Medium_value_payments, High_spent_Small_value_payments, Low_spent_Large_value_payments, ...
Credit_Score	categorical	target	Good, Poor, Standard
ID	text	meta	
Customer_ID	text	meta	

Buttons at the bottom include 'Reset', 'Apply', and 'Browse documentation datasets'.

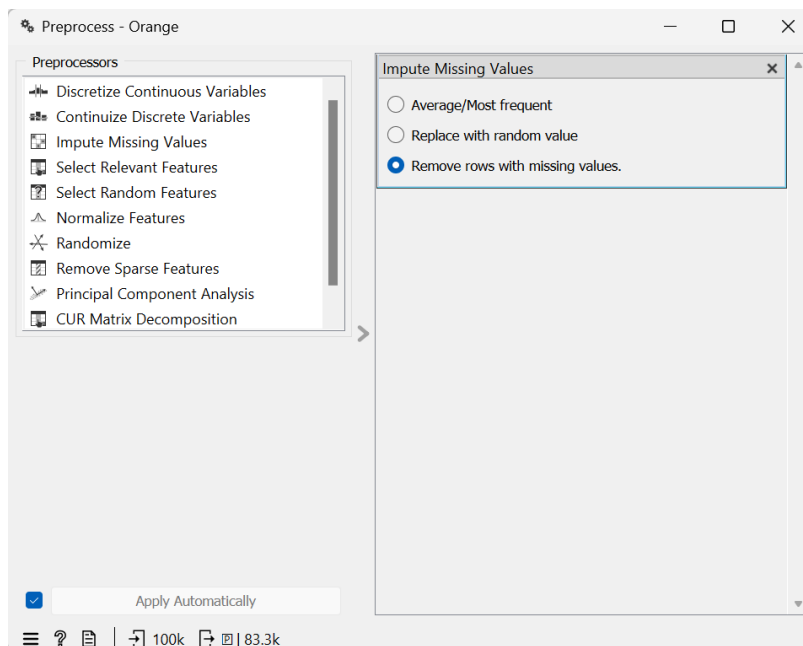
Before preprocessing

The screenshot shows the 'Data Table' widget in the Orange Data Mining software. The 'Info' section displays: 100000 instances, 13 features (1.3 % missing data), Target with 3 values, and 14 meta attributes (2.4 % missing data). The 'Variables' section has checkboxes for 'Show variable labels (if present)', 'Visualize numeric values', and 'Color by instance classes', all of which are checked. The 'Selection' section has a checkbox for 'Select full rows', which is also checked. A 'Restore Original Order' button is present. The 'Send Automatically' checkbox is checked. The data table shows the following columns: Credit_Score, ID, Customer_ID, and a fifth column with names. The data is as follows:

	Credit_Score	ID	Customer_ID	
1	Good	0x1602	CUS_0xd40	Aaror
2	Good	0x1603	CUS_0xd40	Aaror
3	Good	0x1604	CUS_0xd40	Aaror
4	Good	0x1605	CUS_0xd40	Aaror
5	Good	0x1606	CUS_0xd40	Aaror
6	Good	0x1607	CUS_0xd40	Aaror
7	Good	0x1608	CUS_0xd40	Aaror
8	Standard	0x1609	CUS_0xd40	?
9	Standard	0x160e	CUS_0x21b1	Rick F
10	Good	0x160f	CUS_0x21b1	Rick F
11	Standard	0x1610	CUS_0x21b1	Rick F
12	Good	0x1611	CUS_0x21b1	Rick F
13	Good	0x1612	CUS_0x21b1	Rick F
14	Good	0x1613	CUS_0x21b1	Rick F
15	Good	0x1614	CUS_0x21b1	Rick F
16	Good	0x1615	CUS_0x21b1	Rick F
17	Good	0x161a	CUS_0x2dbc	Lange
18	Good	0x161b	CUS_0x2dbc	?
19	Good	0x161c	CUS_0x2dbc	Lange

Buttons at the bottom include 'Restore Original Order', 'Send Automatically', and a status bar showing '100k | 100k | 100k'.

Preprocessing



After preprocessing

Data Table (1) - Orange

Info

83316 instances
13 features
Target with 3 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

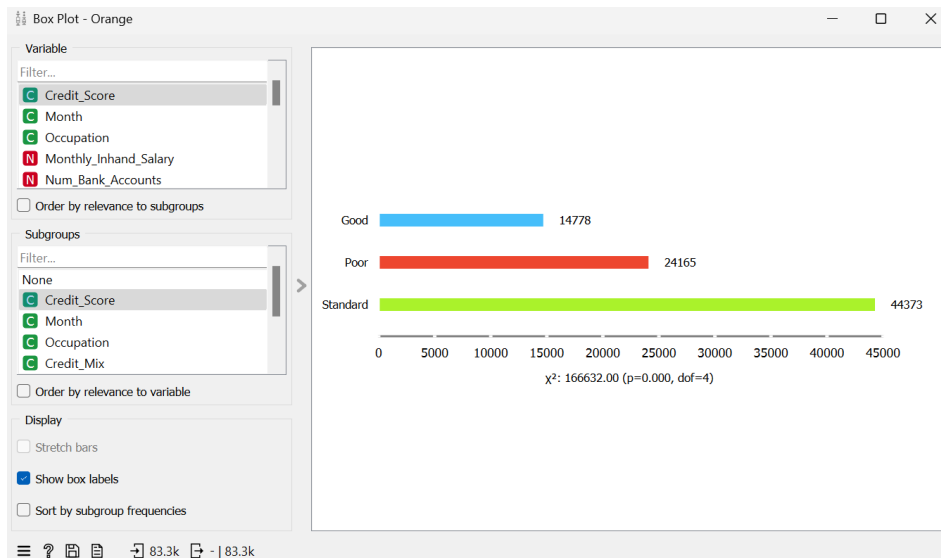
Restore Original Order

☒ Send Automatically

	Credit_Score	ID	Customer_ID	Target
1	Good	0x1602	CUS_0xd40	Aaron
2	Good	0x1606	CUS_0xd40	Aaron
3	Good	0x1608	CUS_0xd40	Aaron
4	Standard	0x1609	CUS_0xd40	?
5	Standard	0x160e	CUS_0x21b1	Rick Ro
6	Good	0x160f	CUS_0x21b1	Rick Ro
7	Standard	0x1610	CUS_0x21b1	Rick Ro
8	Good	0x1612	CUS_0x21b1	Rick Ro
9	Good	0x1613	CUS_0x21b1	Rick Ro
10	Good	0x1615	CUS_0x21b1	Rick Ro
11	Good	0x161a	CUS_0x2dbc	Lange
12	Good	0x161b	CUS_0x2dbc	?
13	Good	0x161d	CUS_0x2dbc	Lange
14	Good	0x161e	CUS_0x2dbc	Lange
15	Good	0x161f	CUS_0x2dbc	Lange
16	Standard	0x1620	CUS_0x2dbc	?
17	Standard	0x1621	CUS_0x2dbc	Lange
18	Standard	0x1626	CUS_0xb891	Jasond
19	Standard	0x1627	CUS_0xb891	Jasond

83.3k 83.3k | 83.3k

Box Plot



Select Columns

The 'Select Columns - Orange' window allows for selecting and filtering variables. It is divided into three main sections: Ignored (19), Features (8), and Target (1). The 'Ignored' section lists 19 variables, including SSN, Age, ID, Customer_ID, Name, Monthly_Balance, Amount_invested_monthly, Credit_History_Age, Outstanding_Debt, Type_of_Loan, Num_of_Delayed_Payment, Changed_Credit_Limit, Num_of_Loan, Annual_Income, Credit_Mix, Payment_Behaviour, Payment_of_Min_Amount, and Occupation. The 'Features' section lists 8 variables: Monthly_Inhand_Salary, Num_Bank_Accounts, Num_Credit_Card, Interest_Rate, Delay_from_due_date, Num_Credit_Inquiries, Credit_Utilization_Ratio, and Total_EMI_per_month. The 'Target' section contains 1 variable: Credit_Score. The 'Metas' section is currently empty. At the bottom, there are buttons for 'Reset', 'Ignore new variables by default', and 'Send Automatically' (checked).

Ignored (19)

Filter

- S SSN
- S Age
- S ID
- S Customer_ID
- S Name
- S Monthly_Balance
- S Amount_invested_monthly
- S Credit_History_Age
- S Outstanding_Debt
- S Type_of_Loan
- S Num_of_Delayed_Payment
- S Changed_Credit_Limit
- S Num_of_Loan
- S Annual_Income
- C Credit_Mix
- C Payment_Behaviour
- C Payment_of_Min_Amount
- C Occupation

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 Credit_Score

Metas

Reset ☐ Ignore new variables by default ☒ Send Automatically

83.3k | 83.3k | 8

Random Forest model with test and score

Random Forest - Orange

Name
Random Forest

Basic Properties

Number of trees: 20

☒ Number of attributes considered at each split: 8

☒ Replicable training

☐ Balance class distribution

Growth Control

☐ Limit depth of individual trees: 3

☒ Do not split subsets smaller than: 2

☒ Apply Automatically

83.3k | -

Test and Score - Orange

☐ Cross validation
Number of folds: 5
☒ Stratified

☐ Cross validation by feature

☐ Random sampling
Repeat train/test: 10
Training set size: 75 %
☒ 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	0.994	0.994	0.994	0.994	0.994	0.991

Compare models by: Area under ROC curve

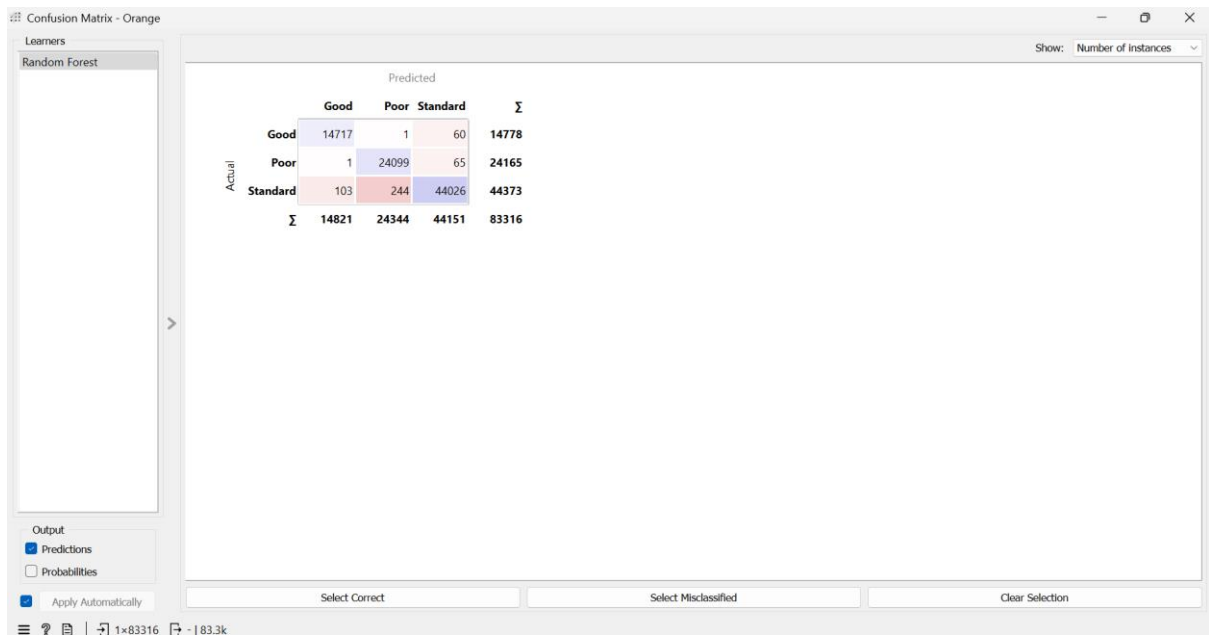
☐ Negligible diff.: 0.1

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 | 1×83316

Confusion Matrix



Test.csv

Test - Orange

Source: ☒ File: Credit score classification\test.csv ☐ URL:

File Type: Automatically detect type

Info: 50000 instances
13 features (1.3% missing values)
Data has no target variable.
14 meta attributes

Columns (Double click to edit)

	Name	Type	Role	Values
13	Payment_Beha...	C categorical	target	!@9#%8, High_spent_Large_value_payments, ...
14	ID	S text	meta	
15	Customer_ID	S text	meta	
16	Name	S text	meta	
17	Age	S text	meta	
18	SSN	S text	meta	

Reset Apply

Browse documentation datasets

50k

Before Preprocessing

Data Table (2) - Orange

Info
50000 instances
12 features (1.4 % 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_Behaviour	ID	Customer_ID	Target
1	Low_spent_Sma...	0x160a	CUS_0xd40	Aaron
2	High_spent_Me...	0x160b	CUS_0xd40	Aaron
3	Low_spent_Me...	0x160c	CUS_0xd40	Aaron
4	High_spent_Me...	0x160d	CUS_0xd40	Aaron
5	High_spent_Lar...	0x1616	CUS_0x21b1	Rick R
6	Low_spent_Larg...	0x1617	CUS_0x21b1	Rick R
7	High_spent_Lar...	0x1618	CUS_0x21b1	Rick R
8	!@9#%8	0x1619	CUS_0x21b1	Rick R
9	Low_spent_Me...	0x1622	CUS_0x2dbc	Lange
10	Low_spent_Larg...	0x1623	CUS_0x2dbc	Lange
11	Low_spent_Me...	0x1624	CUS_0x2dbc	?
12	High_spent_Me...	0x1625	CUS_0x2dbc	Lange
13	High_spent_Me...	0x162e	CUS_0xb891	Jason
14	High_spent_Lar...	0x162f	CUS_0xb891	Jason
15	Low_spent_Me...	0x1630	CUS_0xb891	Jason
16	Low_spent_Sma...	0x1631	CUS_0xb891	Jason
17	High_spent_Me...	0x163a	CUS_0x1cdb	Deepa
18	High_spent_Me...	0x163b	CUS_0x1cdb	Deepa
19	High_spent_Me...	0x163c	CUS_0x1cdb	Deepa

Preprocessing

Preprocess (1) - 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.

Apply Automatically

After Preprocessing

Data Table (3) - Orange

Info
41614 instances
12 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

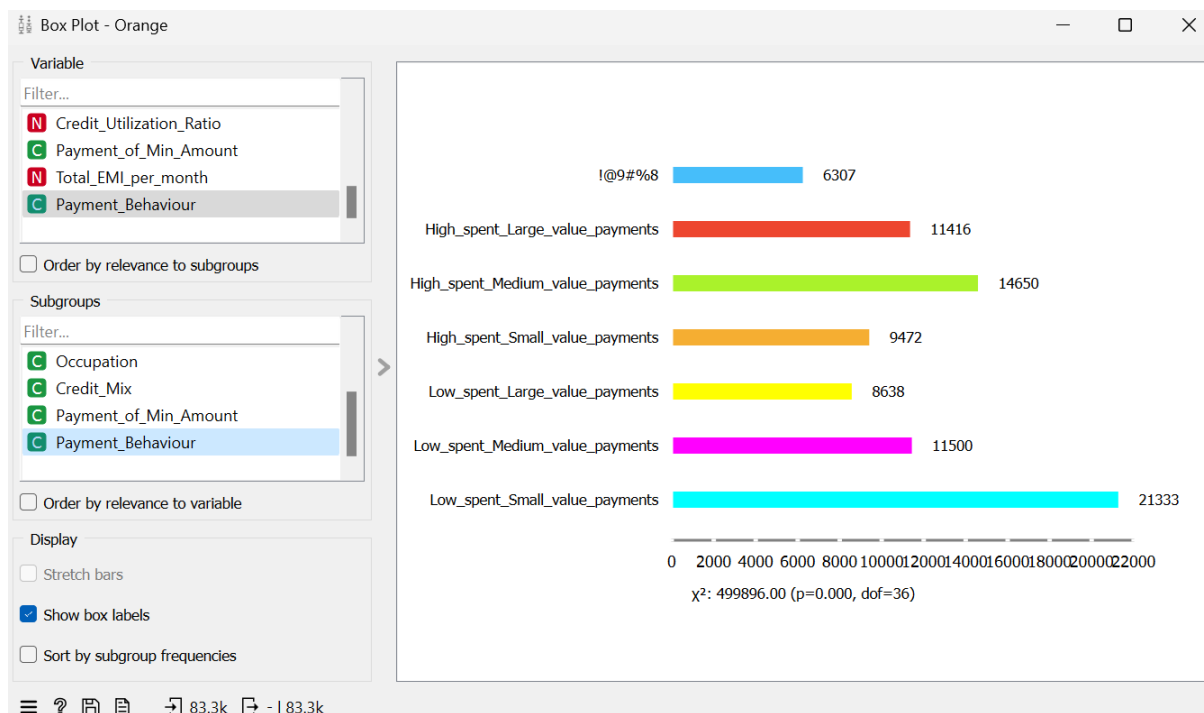
Restore Original Order

☒ Send Automatically

	Payment_Behaviour	ID	Customer_ID	I
1	Low_spent_Sma...	0x160a	CUS_0xd40	Aaron
2	High_spent_Me...	0x160b	CUS_0xd40	Aaron
3	Low_spent_Me...	0x160c	CUS_0xd40	Aaron
4	High_spent_Lar...	0x1616	CUS_0x21b1	Rick Ro
5	Low_spent_Larg...	0x1617	CUS_0x21b1	Rick Ro
6	High_spent_Lar...	0x1618	CUS_0x21b1	Rick Ro
7	!@9#%8	0x1619	CUS_0x21b1	Rick Ro
8	Low_spent_Larg...	0x1623	CUS_0x2dbc	Lange
9	Low_spent_Me...	0x1624	CUS_0x2dbc	?
10	High_spent_Me...	0x1625	CUS_0x2dbc	Lange
11	High_spent_Me...	0x162e	CUS_0xb891	Jasond
12	High_spent_Lar...	0x162f	CUS_0xb891	Jasond
13	Low_spent_Me...	0x1630	CUS_0xb891	Jasond
14	Low_spent_Sma...	0x1631	CUS_0xb891	Jasond
15	High_spent_Me...	0x163a	CUS_0x1cdb	Deepa
16	High_spent_Me...	0x163b	CUS_0x1cdb	Deepa
17	High_spent_Me...	0x163c	CUS_0x1cdb	Deepa
18	Low_spent_Sma...	0x163d	CUS_0x1cdb	Deepa
19	High_spent_Lar...	0x1646	CUS_0x95ee	Np

41.6k | 41.6k | 41.6k

Box Plot



Selected Columns

Select Columns (1) - Orange

Ignored (18)

Filter

S

 SSN

S

 Age

S

 ID

S

 Customer_ID

S

 Name

S

 Monthly_Balance

S

 Amount_invested_monthly

S

 Credit_History_Age

S

 Outstanding_Debt

S

 Type_of_Loan

S

 Num_of_Delayed_Payment

S

 Changed_Credit_Limit

S

 Num_of_Loan

S

 Annual_Income

C

 Credit_Mix

C

 Payment_of_Min_Amount

C

 Occupation

C

 Month

>

>

>

Reset

☐ Ignore new variables by default

☒ Send Automatically

41.6k | -

41.6k | 8

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

Random Forest Model with test and score

Random Forest (1) - Orange ? X

Name
Random Forest (1)

Basic Properties

Number of trees: 20

☒ Number of attributes considered at each split: 8

☒ Replicable training

☐ Balance class distribution

Growth Control

☐ Limit depth of individual trees: 3

☒ Do not split subsets smaller than: 2

☒ Apply Automatically

41.6k | -

Test and Score (1) - Orange - □ X

☐ Cross validation
Number of folds: 5
☒ Stratified

☐ Cross validation by feature

☐ Random sampling
Repeat train/test: 10
Training set size: 75 %
☒ Stratified

☐ Leave one out

☒ Test on train data

☐ Test on test data

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

Model	AUC	CA	F1	Prec	Recall	MCC
Random Forest (1)	1.000	0.997	0.997	0.997	0.997	0.997

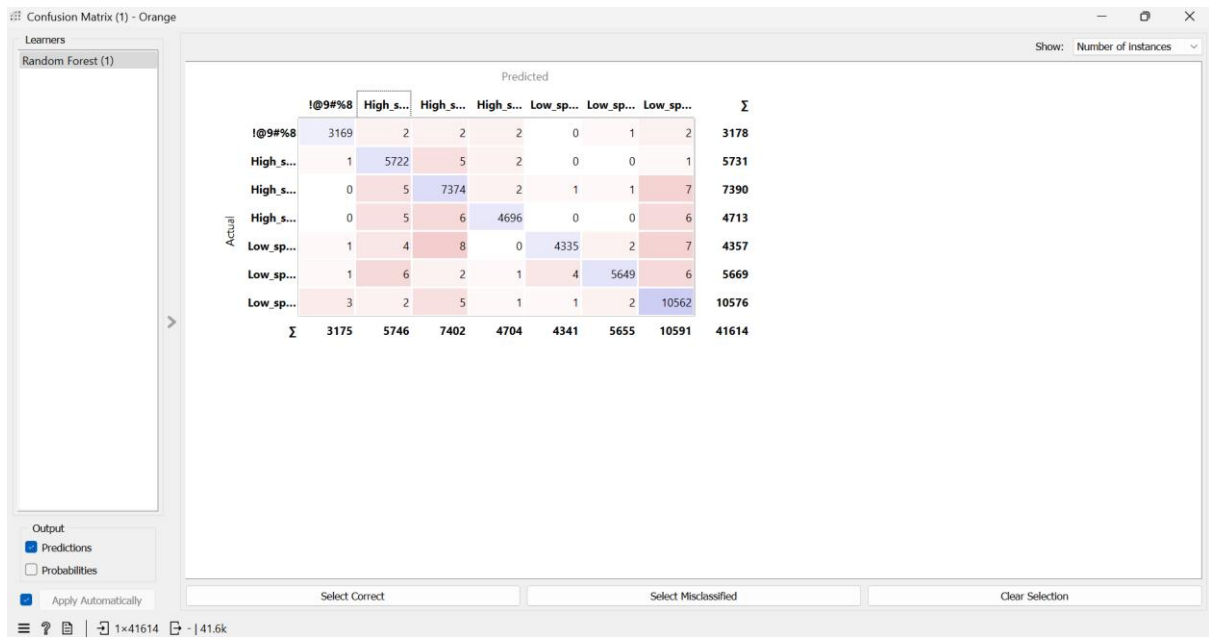
Compare models by: Area under ROC curve v ☐ Negligible diff.: 0.1

Random ...
Random Forest (1)

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.

41.6k | - | 41.6k | 1×41614

Confusion Matrix



Google Colab :

- Persiapan Data
 - Memuat Library yang diperlukan

```
# Digunakan untuk operasi array dan manipulasi numerik.
import numpy as np

# Digunakan untuk manipulasi data dan analisis data.
import pandas as pd

# Library untuk membuat visualisasi data interaktif.
import plotly.express as px

# Library lain untuk membuat visualisasi data interaktif.
import plotly.graph_objects as go

# Library untuk membuat visualisasi data statis.
import matplotlib.pyplot as plt

# Library untuk membuat visualisasi data statistik yang lebih menarik.
import seaborn as sns

# Untuk mengkodekan variabel kategorikal menjadi numerik.
from sklearn.preprocessing import OrdinalEncoder, LabelEncoder

# Untuk pemilihan fitur berdasarkan informasi mutal.
from sklearn.feature_selection import mutual_info_classif

# Untuk membagi data menjadi data latih dan data uji.
from sklearn.model_selection import train_test_split

# Model pohon keputusan ensambel.
from sklearn.ensemble import RandomForestClassifier,
RandomForestRegressor

# Untuk evaluasi model.
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score, mean_squared_error

# Untuk menangani ketidakseimbangan kelas dalam data klasifikasi.
from imblearn.over_sampling import SMOTE

# Implementasi algoritma XGBoost untuk regresi.
from xgboost import XGBRegressor

# Untuk menangani peringatan yang tidak perlu.
import warnings
```

```
# Mengabaikan peringatan FutureWarning.  
warnings.filterwarnings("ignore", category=FutureWarning)
```

Kode ini adalah contoh penggunaan beberapa pustaka umum yang digunakan dalam analisis dan pemodelan data. Mari kita lihat penjelasannya satu per satu:

1. ``import numpy as np``: NumPy adalah pustaka yang digunakan untuk komputasi numerik. Ini menyediakan dukungan untuk array dan matriks, serta berbagai fungsi matematika.

2. ``import pandas as pd``: Pandas adalah pustaka untuk manipulasi dan analisis data. Ini menyediakan struktur data yang kuat dan mudah digunakan yang disebut DataFrame.

3. ``import plotly.express as px``: Plotly Express adalah pustaka untuk membuat visualisasi data interaktif dengan mudah dan cepat.

4. ``import plotly.graph_objects as go``: Plotly Graph Objects adalah pustaka yang memberikan kontrol yang lebih besar dalam membuat visualisasi data dengan Plotly.

5. ``import matplotlib.pyplot as plt``: Matplotlib adalah pustaka yang sering digunakan untuk membuat visualisasi statis dalam Python.

6. ``import seaborn as sns``: Seaborn adalah pustaka yang dibangun di atas Matplotlib untuk membuat visualisasi data statistik yang menarik dan informatif.

7. ``from sklearn.preprocessing import OrdinalEncoder, LabelEncoder``: Sklearn adalah pustaka yang digunakan untuk machine learning di Python. OrdinalEncoder dan LabelEncoder digunakan untuk mengkodekan variabel kategorikal menjadi angka.

8. ``from sklearn.feature_selection import mutual_info_classif``: Ini adalah pustaka dari Sklearn yang digunakan untuk pemilihan fitur berdasarkan informasi mutual antara fitur dan target.

9. ``from sklearn.model_selection import train_test_split``: Ini digunakan untuk membagi dataset menjadi subset pelatihan dan pengujian.

10. ``from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor``: Ini adalah pustaka dari Sklearn yang digunakan untuk membangun model RandomForest untuk klasifikasi dan regresi.

11. ``from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, mean_squared_error``: Ini adalah pustaka dari Sklearn yang digunakan untuk mengevaluasi kinerja model, seperti mencetak laporan klasifikasi, matriks kebingungan, dan menghitung akurasi.

12. ``from imblearn.over_sampling import SMOTE``: Ini adalah pustaka untuk menangani ketidakseimbangan kelas dalam data dengan oversampling menggunakan metode SMOTE.

13. ``from xgboost import XGBRegressor``: XGBoost adalah pustaka yang digunakan untuk implementasi algoritma boosting yang efisien, biasanya digunakan untuk regresi dan klasifikasi.

14. ``import warnings``: Warnings adalah modul bawaan Python yang digunakan untuk mengelola dan menangani pesan peringatan.

Dengan mengimpor semua pustaka ini, Anda siap untuk melakukan analisis dan pemodelan data dengan berbagai teknik dan algoritma. Selanjutnya, Anda dapat menggunakannya untuk mengimpor dataset, membersihkan data, melakukan visualisasi, membangun dan mengevaluasi model machine learning, dan banyak lagi.

- Menghubungkan GDrive ke GColab

```
# Modul untuk menghubungkan Google Drive dengan Colab.
from google.colab import drive

# Menghubungkan Google Drive dengan Colab.
drive.mount('/content/gdrive')
```

Output :

```
Mounted at /content/gdrive
```

- Membaca dan menampilkan data

```
# Membaca file 'train.csv' dari Google Drive.
```

```
train = pd.read_csv('/content/gdrive/MyDrive/Credit score
classification/train.csv', dtype={'Column26': str})

# Menampilkan lima baris pertama dari dataframe 'train'.
train.head()
```

Output :

<ipython-input-3-60b4835c1834>:2: DtypeWarning: Columns (26) have mixed types. Specify dtype option on import or set low_memory=False.

```
train = pd.read_csv('/content/gdrive/MyDrive/Credit score classification/train.csv', dtype={'Column26': str})
```

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	...	Credit_Mix	Outstanding_Debt	Credit_Utilization_Ratio	Credit_Hi
0	0x1602	CUS_0xd40	January	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333	3	...	—	809.98	26.822620	22.Y
1	0x1603	CUS_0xd40	February	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	NaN	3	...	Good	809.98	31.944960	
2	0x1604	CUS_0xd40	March	Aaron Maashoh	500	821-00-0265	Scientist	19114.12	NaN	3	...	Good	809.98	28.609352	22.Y
3	0x1605	CUS_0xd40	April	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	NaN	3	...	Good	809.98	31.377862	22.Y
4	0x1606	CUS_0xd40	May	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333	3	...	Good	809.98	24.797347	22.Y

5 rows x 28 columns

- Pra Prosesan Data
 - Membersihkan data

```
# Perintah menampilkan struktur dataframe train
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
```

Output dari `train.info()` memberikan informasi tentang struktur DataFrame train, termasuk jumlah entri, jumlah kolom, nama kolom, jumlah nilai non-null di setiap kolom, dan tipe data masing-masing kolom.

1. `<class 'pandas.core.frame.DataFrame'>`: Ini menunjukkan bahwa objek yang sedang ditangani adalah DataFrame dari pustaka Pandas.

2. RangeIndex: 100000 entries, 0 to 99999: DataFrame memiliki 100.000 entri dan indeks yang membentang dari 0 hingga 99.999.

3. Data columns (total 28 columns): Terdapat total 28 kolom dalam DataFrame.

4. Column dan Dtype: Daftar kolom dan tipe data masing-masing kolom. Dalam hal ini, terdapat kolom dengan tipe data float64, int64, dan object.

5. Non-Null Count: Menunjukkan jumlah nilai non-null (non-kosong) di setiap kolom. Kolom dengan jumlah non-null yang lebih kecil dari jumlah entri menunjukkan adanya nilai yang hilang (missing values).

- Data Format

- Kolom Age / Num_of_Loan / Num_of_Delayed_Payment

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

1. `train['Age'] = train['Age'].fillna('0').str.extract('(\d+)').astype(float).astype(int):`

- `fillna('0')`: Mengisi nilai yang hilang (missing values) dalam kolom 'Age' dengan string '0'.
- `str.extract('(\d+)')`: Menggunakan ekspresi reguler untuk mengekstrak angka dari setiap nilai dalam kolom 'Age'.
- `astype(float)`: Mengonversi nilai-nilai yang diekstrak menjadi tipe data float.
- `astype(int)`: Mengonversi nilai-nilai float menjadi tipe data integer.
- Hasilnya adalah kolom 'Age' yang sekarang berisi nilai integer yang mewakili usia.

2. `train['Num_of_Loan'] =`

`train['Num_of_Loan'].fillna('0').str.extract('(\d+)').astype(float).astype(int):`

- `fillna('0')`: Mengisi nilai yang hilang (missing values) dalam kolom 'Num_of_Loan' dengan string '0'.
- `str.extract('(\d+)')`: Menggunakan ekspresi reguler untuk mengekstrak angka dari setiap nilai dalam kolom 'Num_of_Loan'.

- `astype(float)`: Mengonversi nilai-nilai yang diekstrak menjadi tipe data float.
- `astype(int)`: Mengonversi nilai-nilai float menjadi tipe data integer.
- Hasilnya adalah kolom 'Num_of_Loan' yang sekarang berisi nilai integer yang mewakili jumlah pinjaman.

3. `train['Num_of_Delayed_Payment'] = train['Num_of_Delayed_Payment'].fillna('0').str.extract('(\d+').astype(float).astype(int):`

- `fillna('0')`: Mengisi nilai yang hilang (missing values) dalam kolom 'Num_of_Delayed_Payment' dengan string '0'.
- `str.extract('(\d+)')`: Menggunakan ekspresi reguler untuk mengekstrak angka dari setiap nilai dalam kolom 'Num_of_Delayed_Payment'.
- `astype(float)`: Mengonversi nilai-nilai yang diekstrak menjadi tipe data float.
- `astype(int)`: Mengonversi nilai-nilai float menjadi tipe data integer.
- Hasilnya adalah kolom 'Num_of_Delayed_Payment' yang sekarang berisi nilai integer yang mewakili jumlah pembayaran yang tertunda.

▪ Kolom 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 untuk menghapus karakter non-digit dari nilai-nilai dalam kolom 'Annual_Income' menggunakan ekspresi reguler, dan kemudian mengonversi nilai-nilai tersebut menjadi tipe data float.

▪ Kolom Changed_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)
```

Kode tersebut mengganti nilai '_' dengan NaN (nilai kosong) dalam kolom 'Changed_Credit_Limit', kemudian mengonversi nilai-nilai tersebut menjadi numerik, dengan parameter `errors='coerce'` yang mengubah nilai yang tidak dapat diubah menjadi NaN. Akhirnya, nilai-nilai NaN diganti dengan '0'.

- Kolom 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)
```

Kode ini untuk mengonversi nilai-nilai dalam kolom 'Outstanding_Debt' menjadi string, kemudian menghapus karakter non-digit dari nilai-nilai tersebut menggunakan ekspresi reguler. Selanjutnya, nilai-nilai tersebut dikonversi menjadi numerik, dengan parameter errors='coerce' yang mengubah nilai yang tidak dapat diubah menjadi NaN. Akhirnya, nilai-nilai NaN diganti dengan '0'.

- Kolom Amount 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)
```

Kode ini untuk mengonversi nilai-nilai dalam kolom 'Amount_invested_monthly' menjadi string, kemudian mengganti nilai kosong dengan '0'. Selanjutnya, karakter non-digit dihapus dari nilai-nilai tersebut menggunakan ekspresi reguler. Nilai-nilai tersebut kemudian dikonversi menjadi numerik, dengan parameter errors='coerce' yang mengubah nilai yang tidak dapat diubah menjadi NaN. Akhirnya, nilai-nilai NaN diganti dengan '0'.

- Kolom 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 untuk mengonversi nilai-nilai dalam kolom 'Monthly_Balance' menjadi string, kemudian menghapus karakter non-digit (kecuali '-') dari nilai-nilai tersebut menggunakan ekspresi reguler. Setelah itu, nilai-nilai tersebut dikonversi menjadi numerik, dengan parameter `errors='coerce'` yang mengubah nilai yang tidak dapat diubah menjadi NaN. Akhirnya, nilai-nilai NaN diganti dengan '0'.

- Kolom 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 `parse_years_and_months` ini mengonversi nilai usia dalam format "tahun" dan "bulan" menjadi total jumlah bulan. Berikut adalah penjelasan singkatnya:

- Fungsi menerima satu argumen, `age`, yang diasumsikan sebagai string.
- Fungsi memeriksa apakah `age` merupakan string atau bukan menggunakan `isinstance()`.
- Jika `age` adalah string, ia akan membaginya berdasarkan string ' Years and ' dan ' Months' untuk mendapatkan bagian tahun dan bulan.
- Nilai tahun diubah menjadi integer, sedangkan nilai bulan diambil dari bagian string kedua dan juga diubah menjadi integer.
- Kemudian, total jumlah bulan dihitung dengan mengalikan tahun dengan 12 dan menambahkannya dengan jumlah bulan.
- Fungsi mengembalikan total jumlah bulan.
- Jika `age` bukan string, fungsi mengembalikan nilai 0.

Penerapan fungsi ini pada kolom 'Credit_History_Age' dari DataFrame `train` akan menghasilkan kolom baru 'Credit_History_Age_Months' yang berisi total jumlah bulan dari usia kredit.

○ Duplikat

```
# Metode duplicated()
duplicates = train[train.duplicated()]

# Menampung jumlah baris yang duplikat menggunakan shape[0]
num_duplicates = duplicates.shape[0]

# Pengujian kondisional
if num_duplicates == 0:
    print("Tidak ada duplikat")
else:
    print("Masih ada", num_duplicates, "duplikat.")
```

Output :

```
Tidak ada duplikat
```

○ Data Scalling

```
# Mendeskripsikan didalam dataframe train
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

- count: Jumlah entri non-null dalam setiap kolom.
- mean: Rata-rata dari setiap kolom.
- std: Standar deviasi dari setiap kolom.

- min: Nilai minimum dari setiap kolom.
- 25%: Kuartil pertama (Q1), atau nilai yang membagi data menjadi 25% terbawah.
- 50%: Median (Q2), atau nilai tengah dari data.
- 75%: Kuartil ketiga (Q3), atau nilai yang membagi data menjadi 25% teratas.
- max: Nilai maksimum dari setiap kolom.

Dengan output diatas, maka akan coba kita hilangkan outlier pada dataframe dengan men-scalling lagi tiap baris yang masih di atas persentil treshhold.

```
# Memilih kolom tertentu
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']]

# Menentukan persentil untuk setiap kolom yang dipilih
percentile_threshold = 0.98
percentiles = selected_columns_train.quantile(percentile_threshold)

# Perulangan pada baris kolom untuk menghapus nilai untuk setiap kolom
yang masih diatas persentil yang ditentukan sebelumnya
for column in selected_columns_train.columns:
    train = train[train[column] <= percentiles[column]]
```

Kode ini memiliki dua tujuan utama:

1. Memilih kolom tertentu dari DataFrame train.
2. Menghapus baris-baris dari DataFrame train yang memiliki nilai di atas persentil tertentu untuk setiap kolom yang dipilih.

```
# Mendeskripsikan didalam dataframe train
train.describe().T
```

Output :

	count	mean	std	min	25%	50%	75%	max
Age	85806.0	120.486003	690.067511	14.000000	25.000000	34.000000	42.000000	8698.000000
Annual_Income	85806.0	49378.620726	36478.390299	7005.930000	19294.460000	36780.030000	70825.760000	166837.640000
Monthly_Inhand_Salary	72894.0	4105.781241	3036.319177	303.645417	1623.778333	3069.091667	5899.970000	14131.123333
Num_Bank_Accounts	85806.0	5.372759	2.585031	-1.000000	3.000000	6.000000	7.000000	10.000000
Num_Credit_Card	85806.0	5.779048	5.727016	0.000000	4.000000	5.000000	7.000000	171.000000
Interest_Rate	85806.0	14.551174	8.824173	1.000000	7.000000	13.000000	20.000000	128.000000
Num_of_Loan	85806.0	7.318381	18.881665	0.000000	2.000000	3.000000	6.000000	100.000000
Delay_from_due_date	85806.0	21.033483	14.786223	-5.000000	10.000000	18.000000	28.000000	67.000000
Num_of_Delayed_Payment	85806.0	12.348321	6.845002	0.000000	8.000000	13.000000	18.000000	25.000000
Changed_Credit_Limit	85806.0	10.189597	6.867078	-6.490000	4.990000	9.280000	14.710000	36.970000
Num_Credit_Inquiries	85806.0	5.757558	3.808858	0.000000	3.000000	5.000000	8.000000	16.000000
Outstanding_Debt	85806.0	1420.471486	1150.336514	0.230000	565.375000	1163.330000	1933.980000	4998.070000
Credit_Utilization_Ratio	85806.0	32.250542	5.098798	20.000000	28.024589	32.263129	36.468620	48.489852
Total_EMI_per_month	85806.0	287.201729	1900.010819	0.000000	29.489975	66.648703	148.997495	29974.000000
Amount_invested_monthly	85806.0	174.614986	190.440132	0.000000	58.040175	115.485412	216.812779	1775.048037
Monthly_Balance	85806.0	393.844633	208.552121	0.000000	268.094008	334.580384	464.420853	1497.941923
Credit_History_Age_Months	85806.0	201.374484	114.274695	0.000000	114.000000	209.000000	292.000000	404.000000

Data sudah terbebas dari outlier.

- Data Entry Plus with Filtering

```
# Untuk menghilangkan nilai '!@9#%8' pada kolom Payment_Behaviour
train = train[train['Payment_Behaviour'] != '@9#%8']
```

```
# Untuk menghilangkan nilai '_____' pada kolom Occupation
train = train[train['Occupation'] != '_____]

# Mencetak nilai unik dari kolom Occupation setelah penghapusan
dilakukan
print(train['Occupation'].unique())
```

Output :

```
['Scientist' 'Teacher' 'Engineer' 'Entrepreneur' 'Developer' 'Lawyer'
'Media_Manager' 'Doctor' 'Journalist' 'Manager' 'Accountant' 'Musician'
'Mechanic' 'Writer' 'Architect']
```

Kode ini melakukan dua tindakan:

1. Menghapus baris di mana kolom 'Payment_Behaviour' memiliki nilai '@9#%8'.
2. Menghapus baris di mana kolom 'Occupation' memiliki nilai '_____', kemudian mencetak nilai unik dari kolom 'Occupation' yang tersisa.

Penjelasan detail :

1. `train = train[train['Payment_Behaviour'] != '@9#%8']`: Ini menghasilkan DataFrame baru di mana baris-baris dengan nilai '@9#%8' dalam kolom 'Payment_Behaviour' dihapus.
2. `train = train[train['Occupation'] != '_____']`: Ini menghasilkan DataFrame baru di mana baris-baris dengan nilai '_____' dalam kolom 'Occupation' dihapus.
3. `print(train['Occupation'].unique())`: Ini mencetak nilai unik dari kolom 'Occupation' setelah penghapusan dilakukan. Hasilnya adalah array yang berisi berbagai jenis pekerjaan yang tersisa setelah proses filtering.

```
# Untuk menghasilkan dataframe baru pada Credit_Mix yang tidak ada nilai '_'
train = train[train['Credit_Mix'] != '_']

# Mencetak nilai unik pada kolom Credit_Mix
print(train['Credit_Mix'].unique())
```

Output :

```
['Good' 'Standard' 'Bad']
```

Kode ini memfilter DataFrame train berdasarkan kolom 'Credit_Mix', yaitu menghapus baris di mana nilai kolom 'Credit_Mix' sama dengan '_'. Kemudian, setelah filtering, kode ini mencetak nilai unik dari kolom 'Credit_Mix' yang tersisa.

Penjelasan detail :

1. `train = train[train['Credit_Mix'] != '_']`: Ini menghasilkan DataFrame baru di mana baris-baris dengan nilai '_' dalam kolom 'Credit_Mix' dihapus.
2. `print(train['Credit_Mix'].unique())`: Ini mencetak nilai unik dari kolom 'Credit_Mix' setelah filtering dilakukan. Hasilnya adalah array yang berisi nilai 'Good', 'Standard', dan 'Bad', yang merupakan nilai unik yang tersisa setelah proses filtering.

- Negative Values

```
# Seleksi kolom untuk filtering
selected_columns = ['Delay_from_due_date', 'Changed_Credit_Limit',
                    'Num_Bank_Accounts']

# Perulangan untuk selain kurang dari 0 dihapus dari dataframe
for column in selected_columns:
    train = train[train[column] >= 0]
```

Filtering baris berdasarkan kolom tertentu :

- Dalam loop for, setiap kolom yang tercantum dalam `selected_columns` ('Delay_from_due_date', 'Changed_Credit_Limit', 'Num_Bank_Accounts') diperiksa.

- Baris-baris di mana nilai dalam kolom tersebut kurang dari 0 dihapus dari DataFrame train.

```
# Dropping Columns (daftar yang ingin di drop)
columns_to_drop = ['ID', 'Customer_ID', 'Month', 'Name', 'SSN',
                  'Credit_History_Age', 'Monthly_Inhand_Salary', 'Type_of_Loan']

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

Menghapus kolom tertentu :

- Kolom-kolom yang tercantum dalam columns_to_drop ('ID', 'Customer_ID', 'Month', 'Name', 'SSN', 'Credit_History_Age', 'Monthly_Inhand_Salary', 'Type_of_Loan') dihapus dari DataFrame train menggunakan metode drop() dengan parameter columns=columns_to_drop.

- inplace=True digunakan untuk mengubah DataFrame train secara langsung tanpa perlu menyimpan hasilnya ke variabel baru.

- Missing Values

```
# Memeriksa total jumlah data yang kosong dalam dataframe
total_missing_values = train.isnull().sum().sum()

# Melakukan pengujian kondisional
if total_missing_values == 0:
    print("Tidak ada data yang kosong")
else:
    print("Jumlah data yang kosong :", total_missing_values)
```

Output :

```
Tidak ada data yang kosong
```

- Feature Engineering
 - Visualisasi

```
# Mengidentifikasi jenis-jenis tipe data dan dimasukkan kedalam numerik kolom
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)

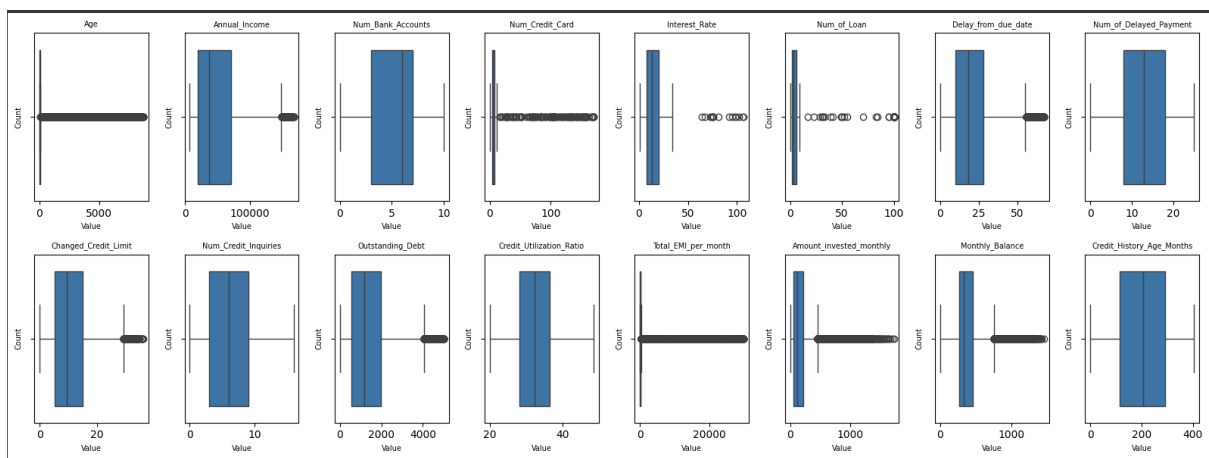
# Loop untuk menyembunyikan subplot yang tidak digunakan
for j in range(len(numeric_columns), num_columns*num_rows):
    axes[j].axis('off')

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

# Menampilkan Plot
plt.show()

```

Output :



Kode ini digunakan untuk membuat beberapa boxplot untuk setiap kolom numerik dalam DataFrame train. Boxplot ini membantu dalam visualisasi distribusi nilai dan deteksi outlier.

Penjelasan singkat:

- `numeric_columns = train.select_dtypes(include=['int64', 'float64']).columns:`
Mengidentifikasi kolom-kolom numerik dalam DataFrame train dan menyimpannya dalam variabel `numeric_columns`.

- num_columns = 8: Menentukan jumlah kolom (grafik) yang ingin ditampilkan dalam setiap baris.
- num_rows = (len(numeric_columns) + num_columns - 1) // num_columns: Menghitung jumlah baris yang diperlukan berdasarkan jumlah kolom numerik dan num_columns.
- fig, axes = plt.subplots(num_rows, num_columns, figsize=(16, 6)): Membuat subplots dengan ukuran dan jumlah baris yang sudah dihitung sebelumnya.
- axes = axes.flatten(): Mengubah array multidimensi axes menjadi array 1 dimensi.
- Loop for digunakan untuk membuat boxplot untuk setiap kolom numerik:
 - sns.boxplot(x=train[column], ax=axes[i]): Membuat boxplot untuk kolom column di subplot ke-i.
 - axes[i].set_title(column, fontsize=7): Menentukan judul subplot dengan nama kolom.
 - axes[i].set_xlabel('Value', fontsize=7): Menentukan label sumbu x.
 - axes[i].set_ylabel('Count', fontsize=7): Menentukan label sumbu y.
- Loop for berikutnya digunakan untuk menyembunyikan subplot yang tidak digunakan.
- plt.tight_layout(): Menata ulang tata letak plot agar sesuai.
- plt.show(): Menampilkan plot.

Dengan kode ini, Anda dapat melihat distribusi nilai dari setiap kolom numerik dalam bentuk boxplot, serta mendeteksi adanya outlier.

○ Scalling

```
# Tune scale pada tiap kolom
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]
```

Kode ini bertujuan untuk melakukan scaling pada beberapa kolom tertentu dalam DataFrame train dengan batasan tertentu.

Penjelasan singkat:

- Setiap baris dalam DataFrame train dipertahankan hanya jika nilai dalam kolom yang tercantum di bawah ini memenuhi batasan tertentu:

- 'Age' kurang dari 60.

'Num_Credit_Card' kurang dari atau sama dengan 10.

- 'Interest_Rate' kurang dari atau sama dengan 50.
- 'Num_of_Loan' kurang dari atau sama dengan 12.
- 'Num_Bank_Accounts' kurang dari atau sama dengan 10.
- 'Delay_from_due_date' kurang dari atau sama dengan 60.
- 'Changed_Credit_Limit' kurang dari atau sama dengan 30.
- 'Num_Credit_Inquiries' kurang dari atau sama dengan 12.
- 'Total_EMI_per_month' kurang dari atau sama dengan 200.
- 'Outstanding_Debt' kurang dari atau sama dengan 1500.

Dengan melakukan ini, Anda membatasi nilai dalam kolom-kolom yang tercantum di atas sesuai dengan batasan yang telah ditentukan.

- Encoding

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

encoder = OrdinalEncoder(categories=[categories])

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

Label Encoder untuk 'Credit_Score':

- Encoder Ordinal digunakan untuk melakukan encoding pada kolom 'Credit_Score'.
- Encoder diinisialisasi dengan daftar kategori yang diinginkan, yaitu ['Poor', 'Standard', 'Good'].
- Hasil dari encoding disimpan dalam kolom baru 'Credit_Score_Encoded'.

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

Label Encoder untuk 'Occupation':

- LabelEncoder dari scikit-learn digunakan untuk melakukan encoding pada kolom 'Occupation'.
- Hasil encoding disimpan dalam kolom baru 'Occupation_Encoded'.

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

encoder = OrdinalEncoder(categories=[categories])

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

Kode di atas menggunakan encoder ordinal untuk melakukan encoding pada kolom 'Credit_Mix' dalam DataFrame 'train'.

Encoder diinisialisasi dengan daftar kategori yang diinginkan, yaitu ['Bad', 'Standard', 'Good'].

Hasil dari encoding disimpan dalam kolom baru 'Credit_Mix_Encoded'. Dengan menggunakan encoder ordinal, setiap kategori dalam kolom 'Credit_Mix' akan diberi label numerik berdasarkan urutan yang ditentukan dalam daftar kategori tersebut.

```
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']])
```

Ordinal Encoder untuk 'Payment_Behaviour':

- Encoder Ordinal digunakan untuk melakukan encoding pada kolom 'Payment_Behaviour'.
- Encoder diinisialisasi dengan daftar kategori yang diinginkan, yaitu categories_payment_behaviour.
- Hasil dari encoding disimpan dalam kolom baru 'Payment_Behaviour_Encoded'.

Kode di atas menggunakan encoder ordinal untuk melakukan encoding pada kolom 'Payment_Behaviour' dalam DataFrame 'train'.

Encoder diinisialisasi dengan daftar kategori yang diinginkan, yaitu
['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'].

Hasil dari encoding disimpan dalam kolom baru 'Payment_Behaviour_Encoded'. Dengan menggunakan encoder ordinal, setiap kategori dalam kolom 'Payment_Behaviour' akan diberi label numerik berdasarkan urutan yang ditentukan dalam daftar kategori tersebut.

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

Menghapus Kolom yang Sudah diencode Sebelumnya:

- Kolom-kolom yang sudah diencode sebelumnya, yaitu 'Payment_Behaviour', 'Credit_Mix', 'Occupation', dan 'Credit_Score', dihapus dari DataFrame train menggunakan metode drop().
- inplace=True digunakan untuk mengubah DataFrame train secara langsung tanpa perlu menyimpan hasilnya ke variabel baru.

- New Features

```
# Menghitung total jumlah (Bank Accounts + Credit Cards)

train['Total_Num_Accounts'] = train['Num_Bank_Accounts'] +
train['Num_Credit_Card']

# Menghitung total jumlah hutang per akun

train['Debt_Per_Account'] = train['Outstanding_Debt'] /
train['Total_Num_Accounts']

# Menghitung perbandingan hutang dan pemasukan

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

# Menghitung total jumlah pembayaran yang telat per akun

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

# Menghitung pengeluaran bulanan (EMI + Investasi bulanan)
```

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

Kode di atas melakukan perhitungan tambahan pada DataFrame train untuk menciptakan fitur-fitur baru yang mungkin berguna dalam analisis atau pemodelan data. Berikut adalah penjelasan dari setiap perhitungan:

1. Total_Num_Accounts:

Menghitung total jumlah akun dengan menjumlahkan 'Num_Bank_Accounts' dan 'Num_Credit_Card'. Fitur ini memberikan gambaran tentang total akun yang dimiliki oleh pelanggan.

2. Debt_Per_Account:

Menghitung rata-rata hutang per akun dengan membagi 'Outstanding_Debt' dengan 'Total_Num_Accounts'. Fitur ini memberikan informasi tentang seberapa besar rata-rata hutang yang dimiliki oleh setiap akun.

3. Debt_to_Income_Ratio:

Menghitung rasio hutang terhadap pendapatan dengan membagi 'Outstanding_Debt' dengan 'Annual_Income'. Fitur ini membantu dalam mengevaluasi kemampuan seseorang untuk membayar hutang berdasarkan pendapatannya.

4. Delayed_Payments_Per_Account:

Menghitung jumlah pembayaran yang telat per akun dengan membagi 'Num_of_Delayed_Payment' dengan 'Total_Num_Accounts'. Fitur ini memberikan informasi tentang seberapa sering pembayaran telat terjadi per akun.

5. Total_Monthly_Expenses:

Menghitung total pengeluaran bulanan dengan menjumlahkan 'Total_EMI_per_month' dan 'Amount_invested_monthly'. Fitur ini memberikan gambaran tentang total pengeluaran bulanan pelanggan termasuk cicilan dan investasi.

Dengan menciptakan fitur-fitur ini, Anda dapat memperkaya data Anda dengan informasi tambahan yang dapat digunakan dalam analisis lebih lanjut atau dalam membangun model prediksi.

○ Mutual Information Scores (MI Score)

```
# Mengidentifikasi kolom-kolom kategorikal dalam DataFrame 'train'  
categorical_columns = train.select_dtypes(include=['object']).columns  
  
# Membuat salinan data 'train' untuk proses encoding  
data_encoded = train.copy()
```

```

# Membuat encoder Ordinal
encoder = OrdinalEncoder()

# Melakukan encoding pada kolom-kolom kategorikal menggunakan encoder Ordinal
data_encoded[categorical_columns] =
encoder.fit_transform(data_encoded[categorical_columns])

# Memisahkan target (y) dan fitur-fitur (X) dari data yang telah diencode
y = data_encoded['Credit_Score_Encoded']
X = data_encoded.drop(columns=['Credit_Score_Encoded'])

# Menghitung skor informasi mutual antara setiap fitur dalam X dan target y
mi_scores = mutual_info_classif(X, y)

# Mencetak skor informasi mutual untuk setiap fitur
for i, score in enumerate(mi_scores):
    print(f"Feature '{X.columns[i]}': Mutual Information Score = {score}")

```

Output :

```

Feature 'Age': Mutual Information Score = 0.01098262062094002
Feature 'Annual_Income': Mutual Information Score = 0.427485666172875
Feature 'Num_Bank_Accounts': Mutual Information Score = 0.06140714265320235
Feature 'Num_Credit_Card': Mutual Information Score = 0.07414252477232597
Feature 'Interest_Rate': Mutual Information Score = 0.09931801640079341
Feature 'Num_of_Loan': Mutual Information Score = 0.02259043893278556
Feature 'Delay_from_due_date': Mutual Information Score = 0.07110212703176733
Feature 'Num_of_Delayed_Payment': Mutual Information Score = 0.054170701460416026
Feature 'Changed_Credit_Limit': Mutual Information Score = 0.09860654680483916
Feature 'Num_Credit_Inquiries': Mutual Information Score = 0.027454255643937886
Feature 'Outstanding_Debt': Mutual Information Score = 0.4320970898018073
Feature 'Credit_Utilization_Ratio': Mutual Information Score = 0.0008940600062685711
Feature 'Payment_of_Min_Amount': Mutual Information Score = 0.07198116200383686
Feature 'Total_EMI_per_month': Mutual Information Score = 0.35146274524439014
Feature 'Amount_invested_monthly': Mutual Information Score = 0.0017832864526063918
Feature 'Monthly_Balance': Mutual Information Score = 0.005211776983816252
Feature 'Credit_History_Age_Months': Mutual Information Score = 0.016993367154926053
Feature 'Occupation_Encoded': Mutual Information Score = 0.0007323001161445575
Feature 'Credit_Mix_Encoded': Mutual Information Score = 0.1802606977966852
Feature 'Payment_Behaviour_Encoded': Mutual Information Score = 0.0028598592021713554
Feature 'Total_Num_Accounts': Mutual Information Score = 0.07872281224039046
Feature 'Debt_Per_Account': Mutual Information Score = 0.4320345518442008
Feature 'Debt_to_Income_Ratio': Mutual Information Score = 0.43457974094499585
Feature 'Delayed_Payments_Per_Account': Mutual Information Score = 0.05648610008981558
Feature 'Total_Monthly_Expenses': Mutual Information Score = 0.005116200567692353

```

Kode di atas digunakan untuk menghitung skor informasi mutual (mutual information score) antara setiap fitur dalam dataset (X) dan target (y) yang telah diencode. Skor informasi mutual mengukur seberapa banyak informasi tentang target yang dapat diperoleh dari suatu

fitur. Hasilnya dicetak dalam loop untuk setiap fitur dalam dataset. Skor ini dapat membantu dalam pemilihan fitur untuk model pembelajaran mesin, di mana fitur-fitur dengan skor yang lebih tinggi cenderung lebih informatif dalam memprediksi target.

```
# Mengurutkan skor informasi mutual dan nama fitur secara terbalik
sorted_mi_scores = sorted(zip(X.columns, mi_scores), key=lambda x:
x[1], reverse=True)

# Mendapatkan nama fitur yang sudah diurutkan
sorted_columns = [x[0] for x in sorted_mi_scores]

# Mendapatkan skor informasi mutual yang sudah diurutkan
sorted_scores = [x[1] for x in sorted_mi_scores]

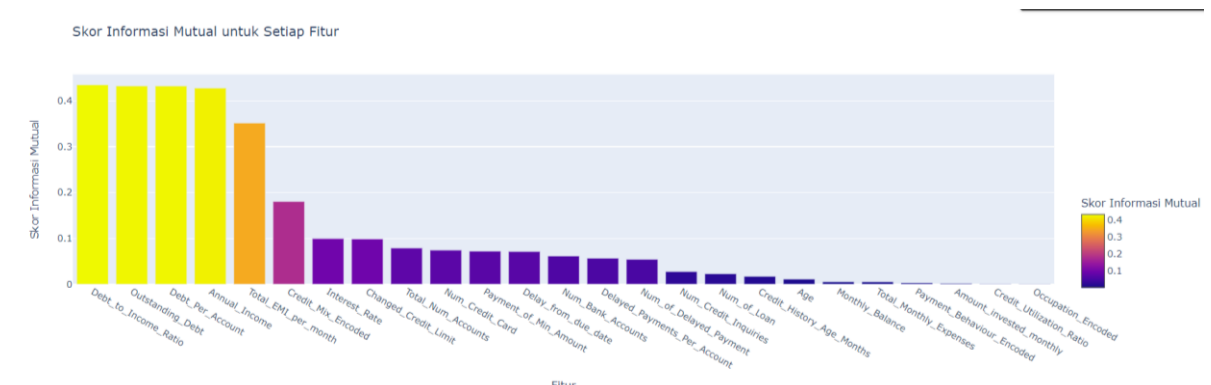
# Skala warna yang digunakan untuk plot
colorscale = 'Viridis'

# Membuat plot batang dengan menggunakan skor informasi mutual dan nama
fitur
fig = go.Figure(data=[go.Bar(x=sorted_columns, y=sorted_scores,
marker=dict(color=sorted_scores, colorbar=dict(title='Skor Informasi
Mutual', len=0.5, y=0.2)))]

# Menyesuaikan tata letak plot dan memberi judul pada sumbu-sumbunya
fig.update_layout(title='Skor Informasi Mutual untuk Setiap Fitur',
                    xaxis_title='Fitur',
                    yaxis_title='Skor Informasi Mutual')

# Menampilkan plot
fig.show()
```

Output :



Kode di atas menghasilkan sebuah plot batang yang menunjukkan skor informasi mutual untuk setiap fitur dalam dataset, yang telah diurutkan berdasarkan skor tersebut secara menurun. Hal ini membantu dalam visualisasi dan pemahaman tentang seberapa informatif setiap fitur dalam memprediksi target. Skala warna pada plot menunjukkan seberapa tinggi skor informasi mutual untuk setiap fitur.

- Machine Learning Model

```
# Menyiapkan dataframe untuk latih pada variabel y pada kolom
Credit_Score_Encoded
y = train['Credit_Score_Encoded']

# # Menyiapkan dataframe untuk latih pada variabel X pada kolom yang
determined dibawah
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 data 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)
```

Kode di atas membagi data menjadi data latih (X_train, y_train) dan data uji (X_test, y_test) dengan proporsi 80:20.

Fitur-fitur yang digunakan untuk membangun model adalah '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', dan 'Delayed_Payments_Per_Account'.

Data diacak menggunakan nilai seed 77 untuk memastikan hasil yang konsisten dalam pengujian berulang-ulang.

- Random Forest

```
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)
```

Kode di atas menciptakan model Regresi Random Forest dengan menggunakan kelas RandomForestRegressor dari modul sklearn.ensemble. Model ini akan dilatih dengan menggunakan data latih (X_train dan y_train).

Parameter yang digunakan untuk membuat model ini adalah sebagai berikut:

- n_estimators=500: Menentukan jumlah pohon keputusan yang akan digunakan dalam ensemble. Dalam hal ini, kami menggunakan 500 pohon.
- bootstrap=True: Menentukan apakah pengambilan sampel bootstrap dilakukan saat pembuatan setiap pohon. Jika diatur ke True, setiap pohon akan dilatih dengan sampel yang diambil secara acak dengan penggantian dari data latih. Ini membantu meningkatkan variasi antar pohon, yang sering kali menghasilkan model yang lebih kuat.
- random_state=77: Menentukan seed untuk pengacakan. Seed ini digunakan untuk membuat hasil yang dapat direproduksi, sehingga model akan memberikan hasil yang konsisten ketika dijalankan berulang kali dengan seed yang sama.

Setelah model dibuat, fungsi fit() digunakan untuk melatih model menggunakan data latih, yaitu fitur-fitur dalam X_train dan label y_train. Model ini akan belajar untuk memetakan fitur-fitur ini ke label y_train, sehingga dapat melakukan prediksi nilai Credit_Score_Encoded berdasarkan fitur-fitur yang diberikan.

- Model Evaluation
 - Mean Squared Error (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
```

Kode di atas digunakan untuk membuat prediksi menggunakan model Regresi Random Forest yang telah dilatih sebelumnya (model) pada data uji (X_test). Prediksi kemudian disimpan dalam variabel y_pred.

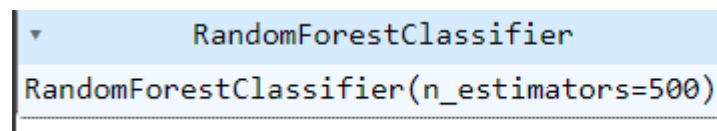
Selanjutnya, menggunakan nilai prediksi (y_pred) dan nilai sebenarnya dari data uji (y_test), dihitung Mean Squared Error (MSE) dengan memanggil fungsi mean_squared_error() dari modul sklearn.metrics. Nilai MSE digunakan sebagai metrik untuk mengevaluasi seberapa baik model melakukan prediksi terhadap data yang belum pernah dilihat sebelumnya.

Hasil MSE kemudian dicetak untuk memberikan informasi tentang seberapa baik model yang telah dilatih mampu melakukan prediksi terhadap data uji. Semakin rendah nilai MSE, semakin baik kinerja model dalam melakukan prediksi.

- Accuracy Test

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

Output :



```
RandomForestClassifier  
RandomForestClassifier(n_estimators=500)
```

Kode di atas digunakan untuk membuat dan melatih model klasifikasi Random Forest menggunakan RandomForestClassifier dari library scikit-learn. Model tersebut dilatih pada data latih (X_train dan y_train) menggunakan 500 pohon keputusan (estimators) dan penggunaan sampel bootstrap untuk pembangunan setiap pohon.

Setelah melatih model, model tersebut siap untuk digunakan untuk membuat prediksi kelas pada data uji atau digunakan untuk tujuan klasifikasi lainnya.

```
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))
```

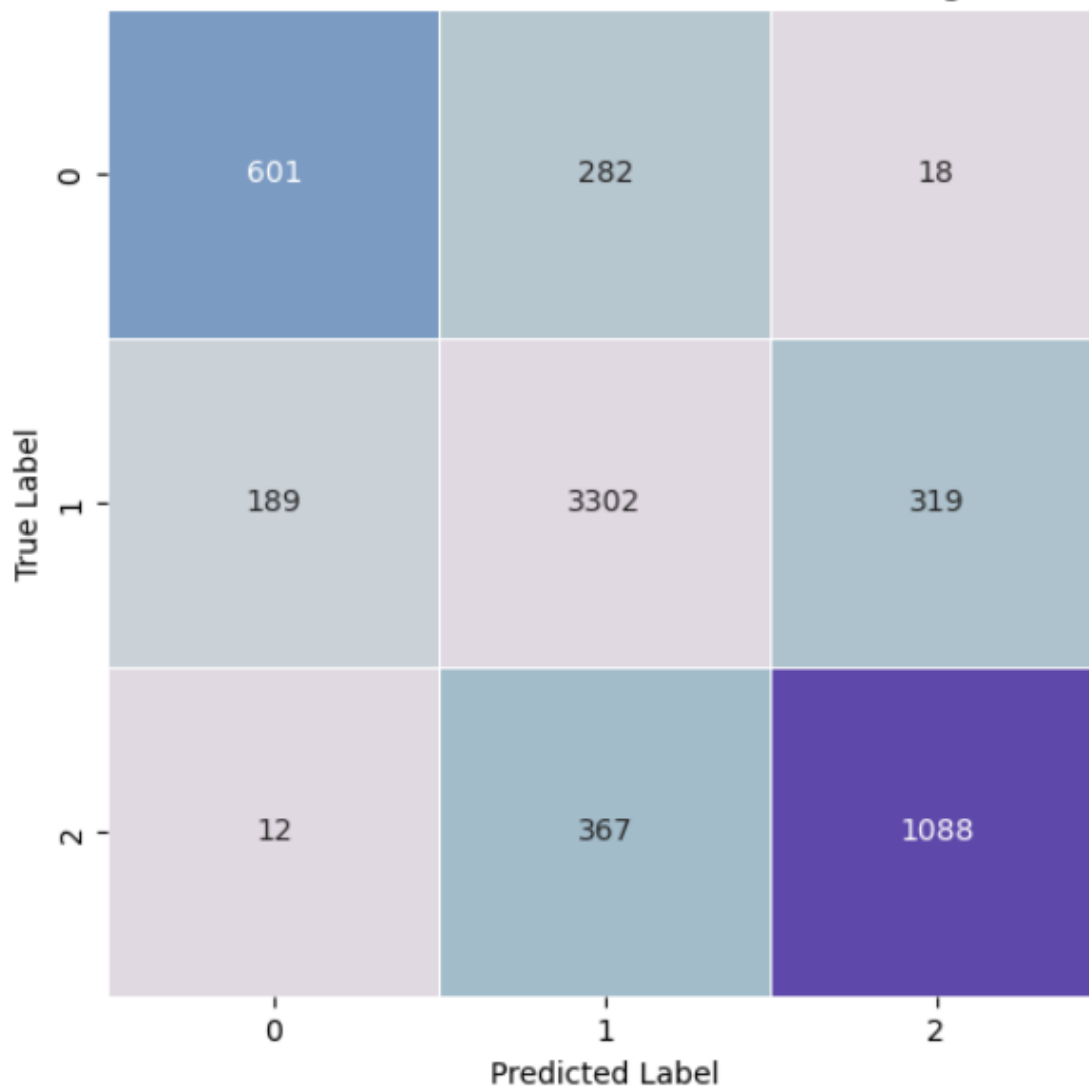
Output :

```
Accuracy on original test set: 0.8078666235027517
```

```
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	0.76	0.74	0.75	1467
accuracy			0.81	6178
macro avg	0.78	0.76	0.77	6178
weighted avg	0.81	0.81	0.81	6178

Confusion Matrix for RandomForestClassifier on original test set



Kode tersebut digunakan untuk melakukan evaluasi kinerja model klasifikasi Random Forest pada data uji. Pertama, model digunakan untuk membuat prediksi kelas pada data uji (`X_test`) menggunakan metode `predict()`.

Kemudian, akurasi model dihitung dengan membandingkan prediksi yang dihasilkan (`y_pred`) dengan label sebenarnya dari data uji (`y_test`) menggunakan fungsi `accuracy_score()` dari modul `sklearn.metrics`.

Selanjutnya, dilakukan pembuatan confusion matrix dengan memanfaatkan fungsi `confusion_matrix()` dari modul `sklearn.metrics` untuk menunjukkan seberapa baik model dalam memprediksi kelas tertentu. Confusion matrix ini ditampilkan menggunakan heatmap dari library Seaborn.

Terakhir, dilakukan pencetakan laporan klasifikasi yang mencakup beberapa metrik evaluasi seperti precision, recall, dan f1-score menggunakan fungsi `classification_report()` dari modul `sklearn.metrics`. Laporan ini memberikan informasi lebih detail tentang kinerja model dalam mengklasifikasikan setiap kelas.