In [116…
```
import pandas as pd
import numpy as np
import plotly.express as px
import pyodbc
import datasist as ds
import seaborn as sns
import matplotlib
import warnings
from matplotlib import pyplot as plt
%matplotlib inline
from pandas_profiling import ProfileReport
```

In [117…
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score
from sklearn.model_selection import cross_val_score, train_test_split
```

In [118…
```
data_risk = pd.read_csv("C:/Users/jeana/OneDrive/Desktop/Data Analytics-Capstone Proje
```

In [119…
```
data_risk.head()
```

Out[119]:

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	original_l
0	1000	6000.0	5	83.0	300.0	
1	1001	39000.0	5	82.0	7200.0	
2	1002	18000.0	5	78.0	2700.0	
3	1003	23250.0	3	76.0	3900.0	
4	1004	12000.0	3	74.0	2100.0	

In [120…
```
profile = ProfileReport(data_risk, title="Loan Delinquent Prediction")
profile
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]
Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]
Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
```

Overview

Dataset statistics

Number of variables	11
Number of observations	5783
Missing cells	1885
Missing cells (%)	3.0%
Duplicate rows	533
Duplicate rows (%)	9.2%
Total size in memory	497.1 KiB
Average record size in memory	88.0 B

Variable types

Numeric	9
Categorical	2

Alerts

Dataset has 533 (9.2%) duplicate rows	**Duplicates**
`monthly_income` is highly overall correlated with `closing_principal_balance` and 1 other fields (closing_principal_balance, original_loan_amount)	**High correlation**

Out[120]:

In [121…
```
data_risk["loan_id"].nunique()
```

Out[121]:
```
5250
```

In [122…
```
ds.structdata.describe(data_risk)
```

```
First five data points
```

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	original_l
0	1000	6000.0	5	83.0	300.0	
1	1001	39000.0	5	82.0	7200.0	
2	1002	18000.0	5	78.0	2700.0	
3	1003	23250.0	3	76.0	3900.0	
4	1004	12000.0	3	74.0	2100.0	

Random five data points

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	origin
3062	3779	NaN	2	NaN	NaN	
3320	4017	24750.0	4	23.0	13800.0	
4412	5006	11250.0	3	34.0	1350.0	
2144	2948	18750.0	2	24.0	14400.0	
1986	2805	15000.0	2	20.0	13500.0	

Last five data points

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	origin
5778	6245	8250.0	4	6.0	6000.0	
5779	6246	2250.0	5	6.0	1500.0	
5780	6247	3750.0	5	6.0	3000.0	
5781	6248	9750.0	7	6.0	7500.0	
5782	6249	2250.0	6	6.0	1500.0	

Shape of data set: (5783, 11)

Size of data set: 63613

Data Types
Note: All Non-numerical features are identified as objects in pandas

	Data Type
loan_id	int64
monthly_income	float64
origination_score_band	int64
TOB_months	float64
closing_principal_balance	float64
original_loan_amount	float64
product	object
original_loan_term	int64
remaining_loan_term	int64
delq_history	float64
target	int64

```
Numerical Features in Data set
['loan_id', 'monthly_income', 'origination_score_band', 'TOB_months', 'closing_princi
pal_balance', 'original_loan_amount', 'original_loan_term', 'remaining_loan_term', 'd
elq_history', 'target']


Categorical Features in Data set
['product']

Statistical Description of Columns
```

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance
count	5783.000000	5406.000000	5783.000000	5406.000000	5406.000000
mean	3625.042711	14314.372919	3.904029	28.385683	5714.391417
std	1516.292643	13799.514988	1.490877	16.357494	7837.483534
min	1000.000000	750.000000	1.000000	6.000000	100.000000
25%	2310.500000	6000.000000	3.000000	17.000000	1380.000000
50%	3623.000000	9750.000000	4.000000	24.000000	3000.000000
75%	4934.500000	17250.000000	5.000000	38.000000	6800.000000
max	6249.000000	99750.000000	8.000000	83.000000	88200.000000

```
Description of Categorical Features
```

	count	unique	top	freq
product	5783	4	A	2942

```
Unique class Count of Categorical features
```

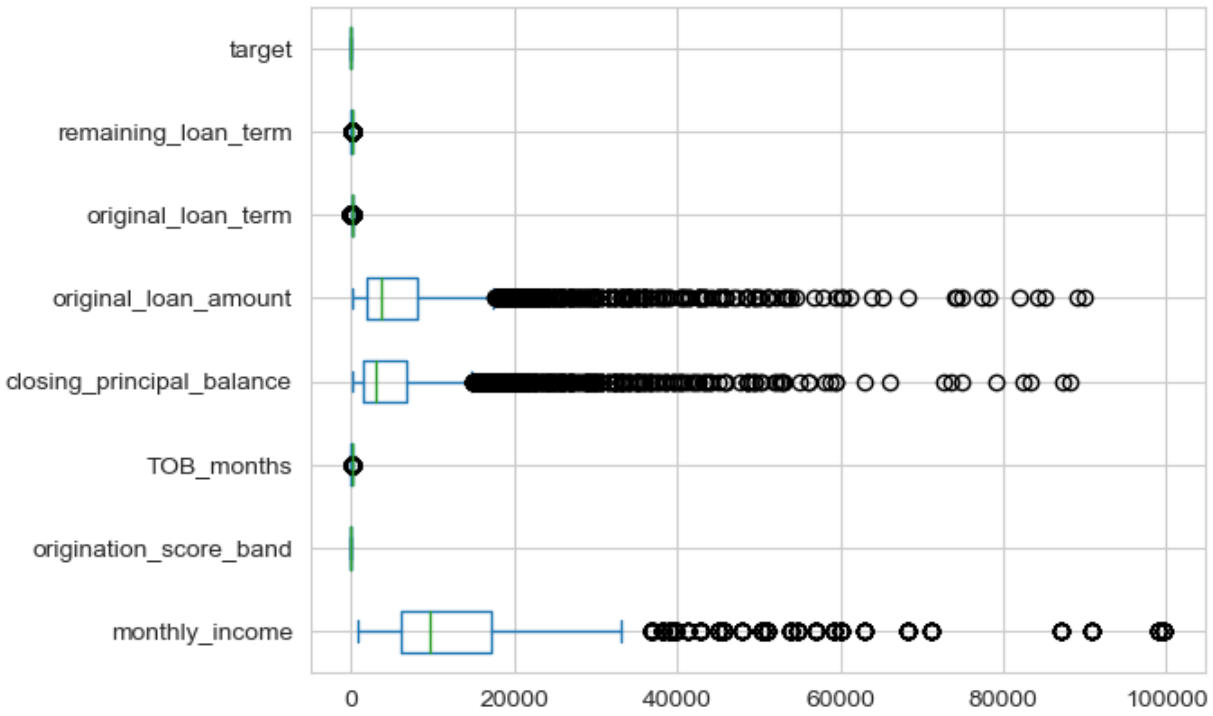
	Feature	Unique Count
0	product	4

Missing Values in Data

	features	missing_counts	missing_percent
0	loan_id	0	0.0
1	monthly_income	377	6.5
2	origination_score_band	0	0.0
3	TOB_months	377	6.5
4	closing_principal_balance	377	6.5
5	original_loan_amount	377	6.5
6	product	0	0.0
7	original_loan_term	0	0.0
8	remaining_loan_term	0	0.0
9	delq_history	377	6.5
10	target	0	0.0

In [123…
```
n_col = ['monthly_income', 'origination_score_band', 'TOB_months',
         'closing_principal_balance', 'original_loan_amount',
         'original_loan_term', 'remaining_loan_term','target']
```

In [124…
```
data_risk[n_col].plot(kind='box', vert=False);
```



outliers

In [125…
```
data_risk[data_risk.isnull().any(axis=1)]
```

Out[125]:

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	origir
17	1016	NaN	4	NaN	NaN	
22	1021	NaN	3	NaN	NaN	
53	1049	NaN	3	NaN	NaN	
85	1077	NaN	3	NaN	NaN	
88	1079	NaN	5	NaN	NaN	
...	...	...	...	...	...	
5704	6174	NaN	5	NaN	NaN	
5717	6186	NaN	2	NaN	NaN	
5723	6192	NaN	4	NaN	NaN	
5729	6198	NaN	5	NaN	NaN	
5760	6227	NaN	5	NaN	NaN	

377 rows × 11 columns

In [126…
```
data_risk.isnull().sum()
```

Out[126]:
```
loan_id                      0
monthly_income             377
origination_score_band       0
TOB_months                 377
closing_principal_balance  377
original_loan_amount       377
product                      0
original_loan_term           0
remaining_loan_term          0
delq_history               377
target                       0
dtype: int64
```

In [9]:
```
data_risk
```

Out[9]:

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	origin
0	1000	6000.0	5	83.0	300.0	
1	1001	39000.0	5	82.0	7200.0	
2	1002	18000.0	5	78.0	2700.0	
3	1003	23250.0	3	76.0	3900.0	
4	1004	12000.0	3	74.0	2100.0	
...	...	...	...	...	...	
5778	6245	8250.0	4	6.0	6000.0	
5779	6246	2250.0	5	6.0	1500.0	
5780	6247	3750.0	5	6.0	3000.0	
5781	6248	9750.0	7	6.0	7500.0	
5782	6249	2250.0	6	6.0	1500.0	

5783 rows × 11 columns

In [11]:
```
data_risk['monthly_income'] = data_risk['monthly_income'].fillna(data_risk['monthly_ir
```

In [12]:
```
data_risk['TOB_months'] = data_risk['TOB_months'].fillna(data_risk['TOB_months'].mean(
```

In [13]:
```
data_risk['closing_principal_balance'] = data_risk['closing_principal_balance'].fillna
```

In [14]:
```
data_risk['original_loan_amount'] = data_risk['original_loan_amount'].fillna(data_risk
```

In [15]:
```
data_risk['TOB_months'] = data_risk['TOB_months'].fillna(data_risk['TOB_months'].mean(
```

In [16]:
```
data_risk['delq_history'] = data_risk['delq_history'].fillna(data_risk['delq_history']
```

In [ ]:
```

```

In [17]:
```
data_risk.isnull().sum()
```

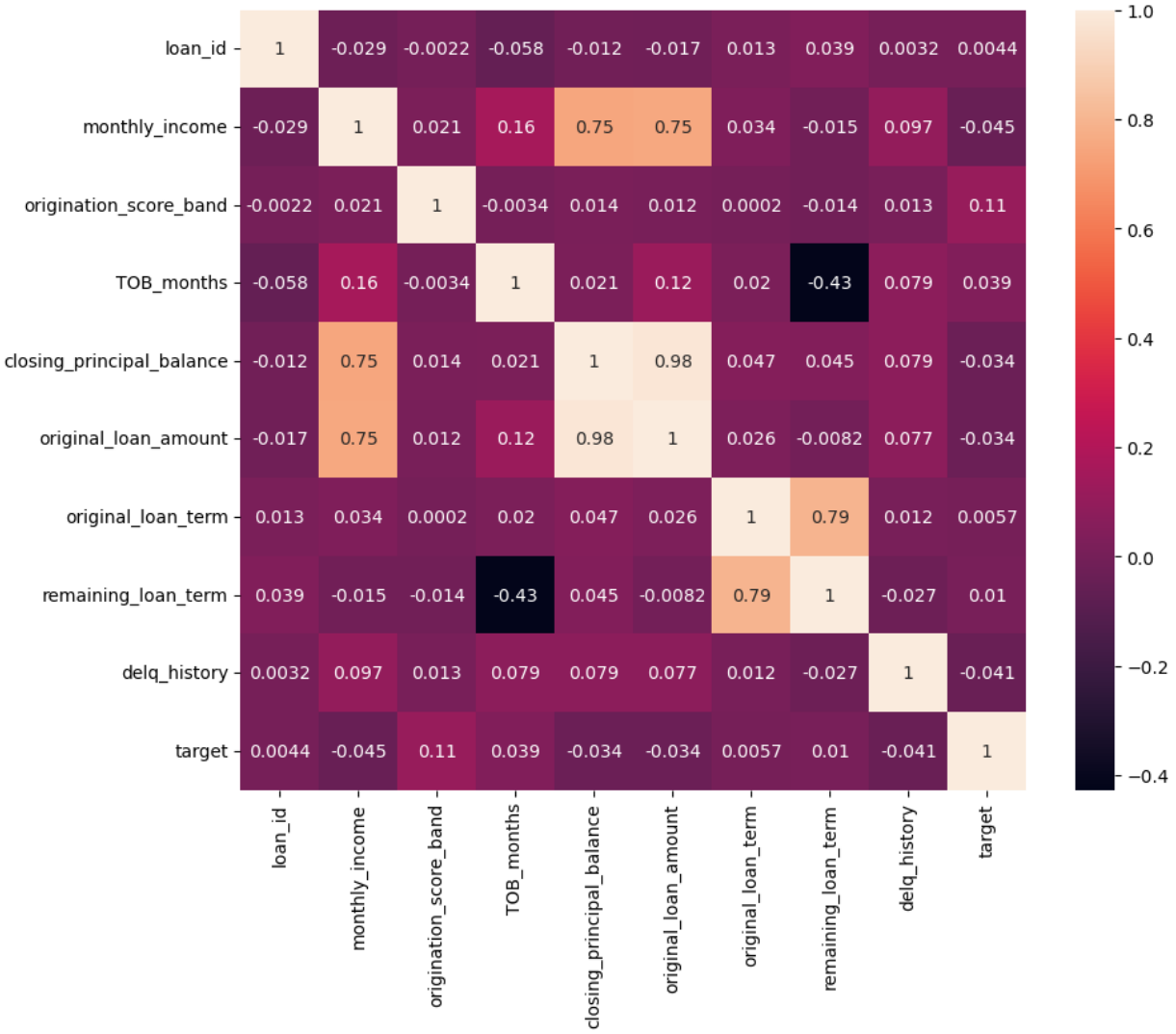
Out[17]:
```
loan_id                    0
monthly_income             0
origination_score_band     0
TOB_months                 0
closing_principal_balance  0
original_loan_amount       0
product                    0
original_loan_term         0
remaining_loan_term        0
delq_history               0
target                     0
dtype: int64
```

In [18]:
```
data_risk.info()
```

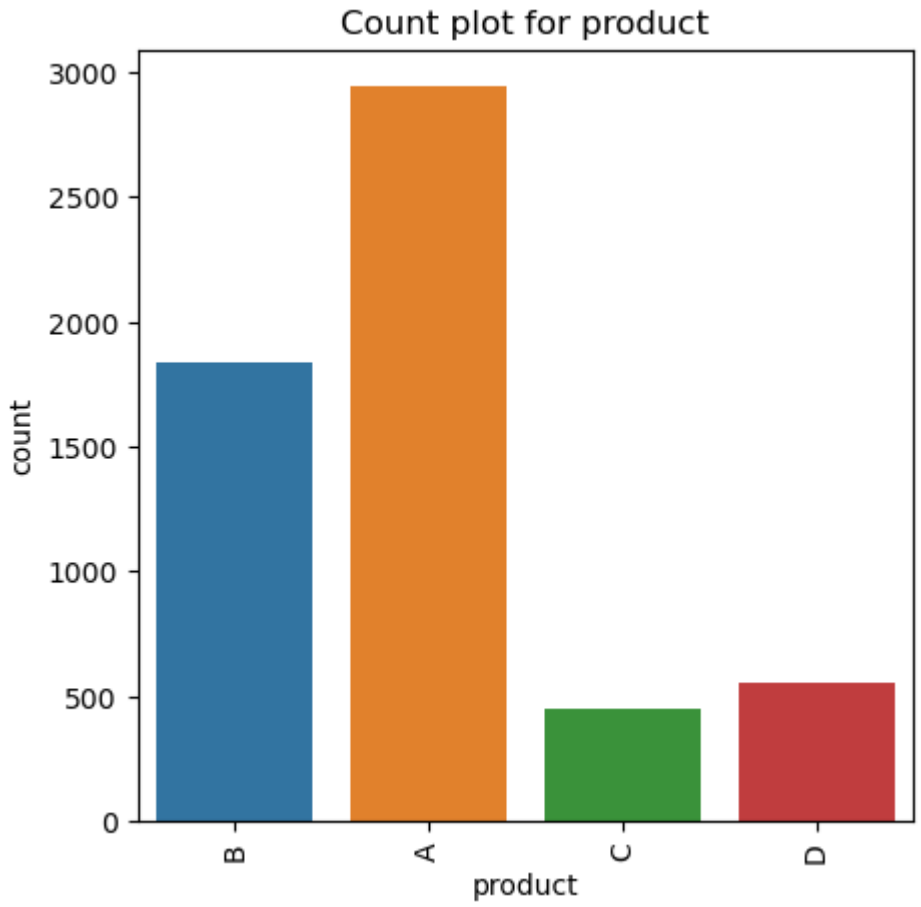
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5783 entries, 0 to 5782
Data columns (total 11 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   loan_id                    5783 non-null   int64
 1   monthly_income             5783 non-null   float64
 2   origination_score_band     5783 non-null   int64
 3   TOB_months                 5783 non-null   float64
 4   closing_principal_balance  5783 non-null   float64
 5   original_loan_amount       5783 non-null   float64
 6   product                    5783 non-null   object
 7   original_loan_term         5783 non-null   int64
 8   remaining_loan_term        5783 non-null   int64
 9   delq_history               5783 non-null   float64
 10  target                     5783 non-null   int64
dtypes: float64(5), int64(5), object(1)
memory usage: 497.1+ KB
```

In [19]:
```
corr = data_risk.corr()
plt.figure(figsize=(10,8))
sns.heatmap(corr, annot = True)
```

Out[19]:
```
<AxesSubplot:>
```



In [21]:
```
ds.visualizations.countplot(data_risk)
```



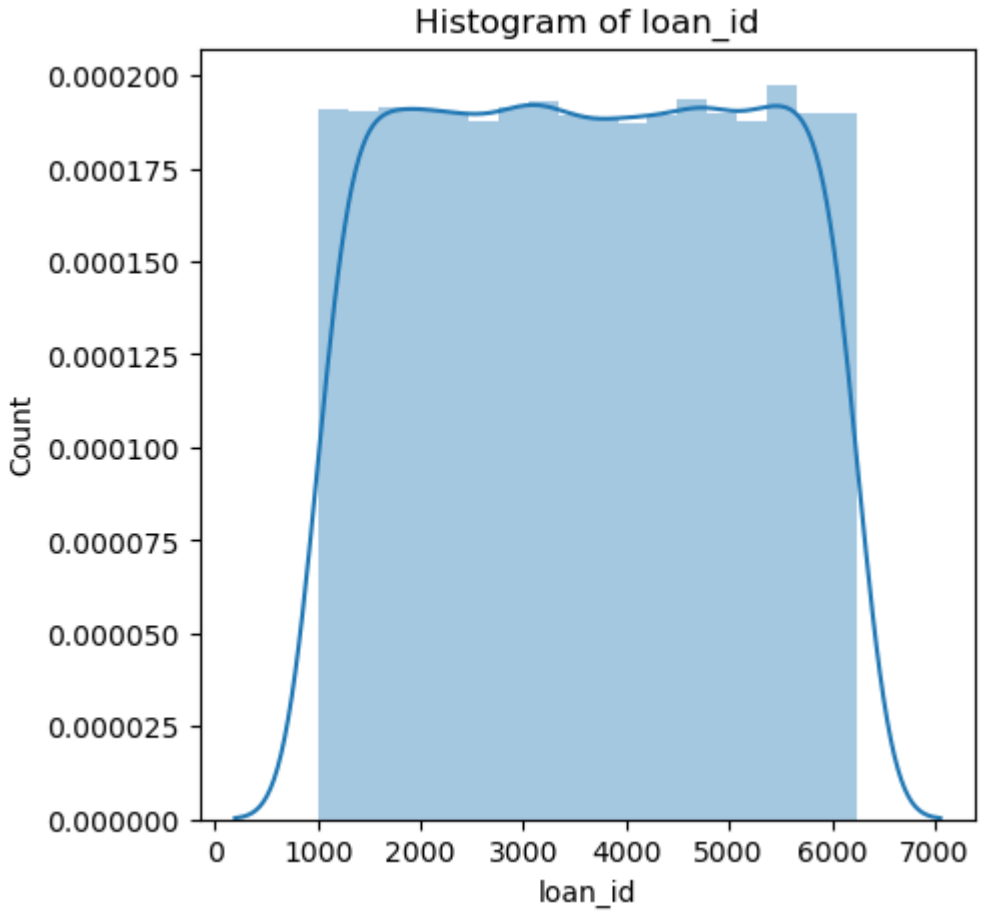
In [22]:
```
ds.visualizations.class_count(data_risk)
```

Class Count for product

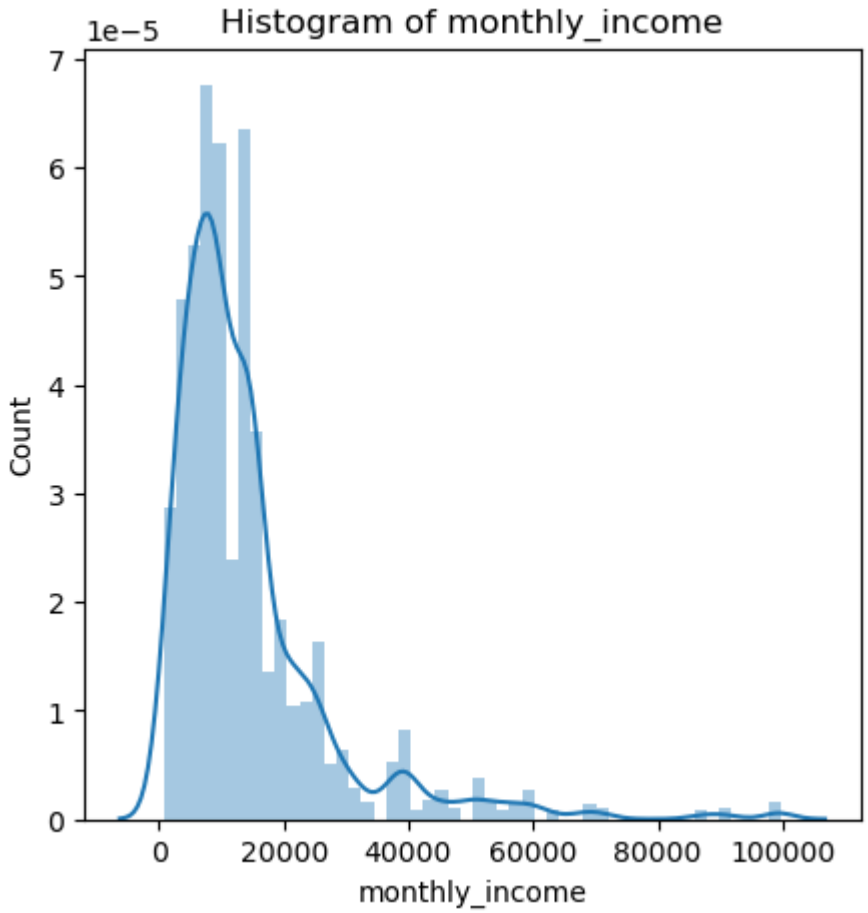
	product
A	2942
B	1837
D	554
C	450

In [23]:
```
ds.visualizations.histogram(data_risk)
```

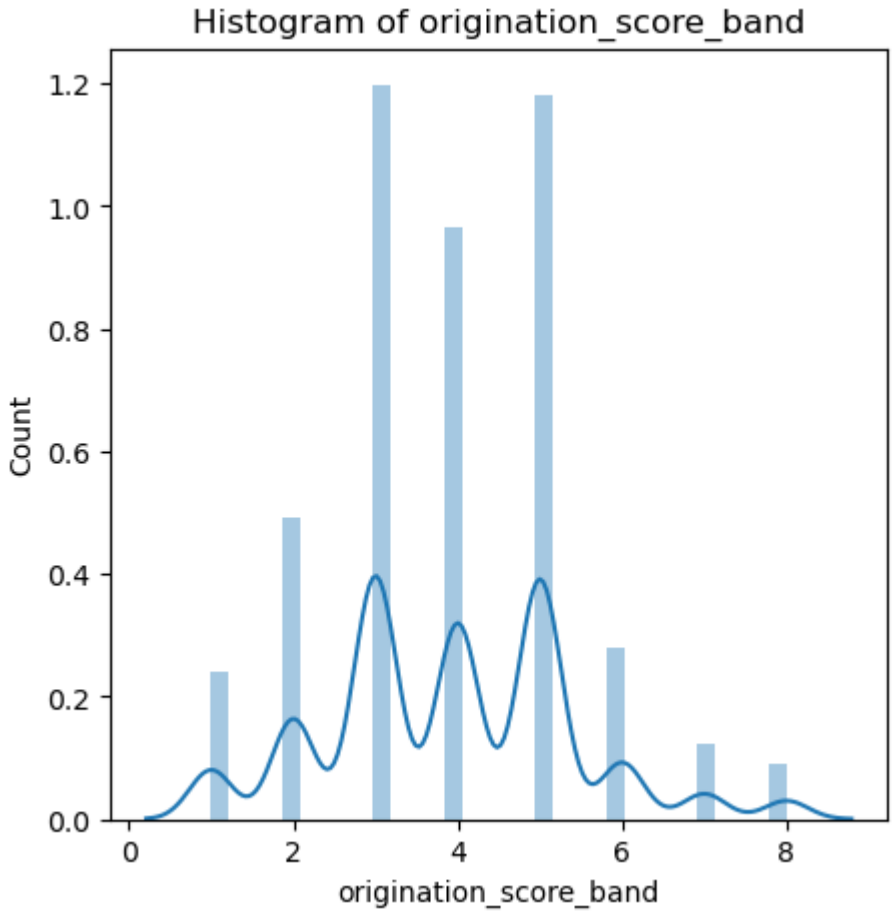
```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



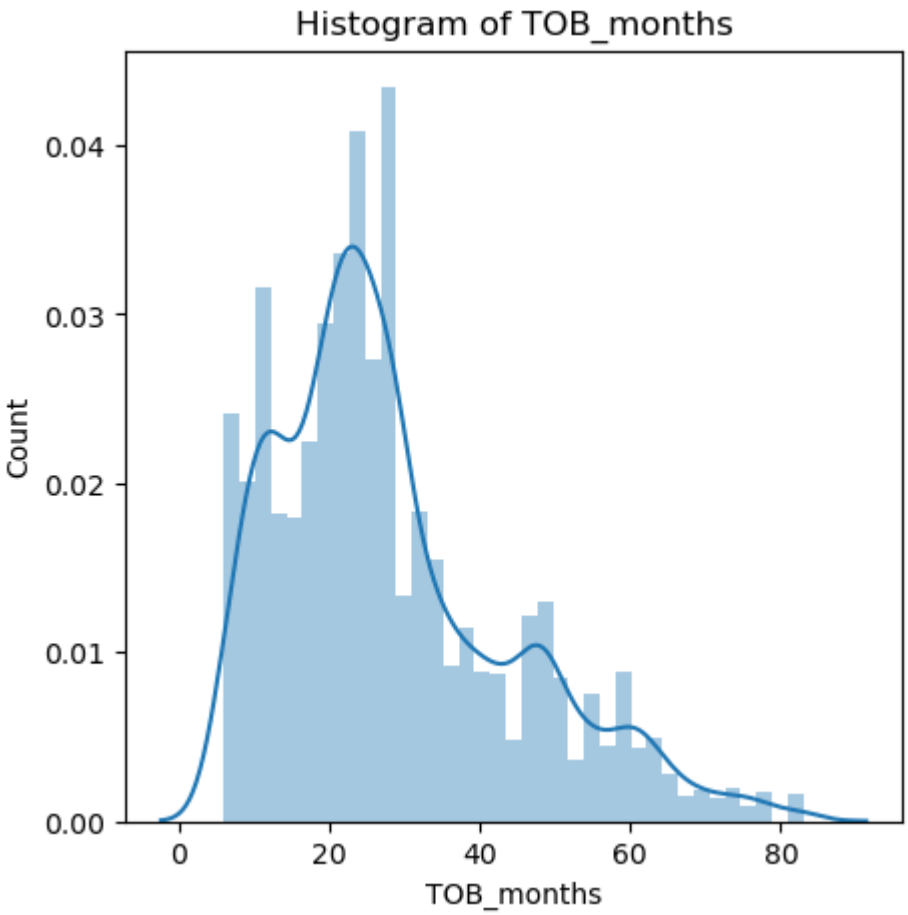

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



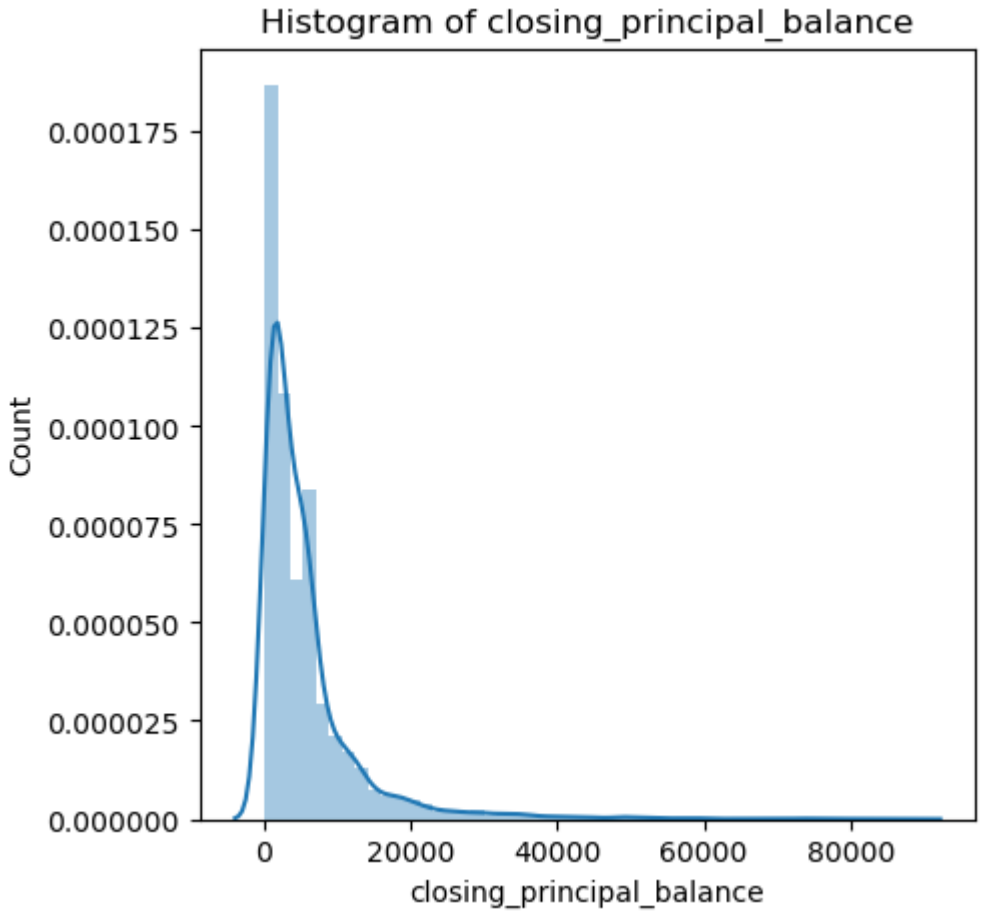

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



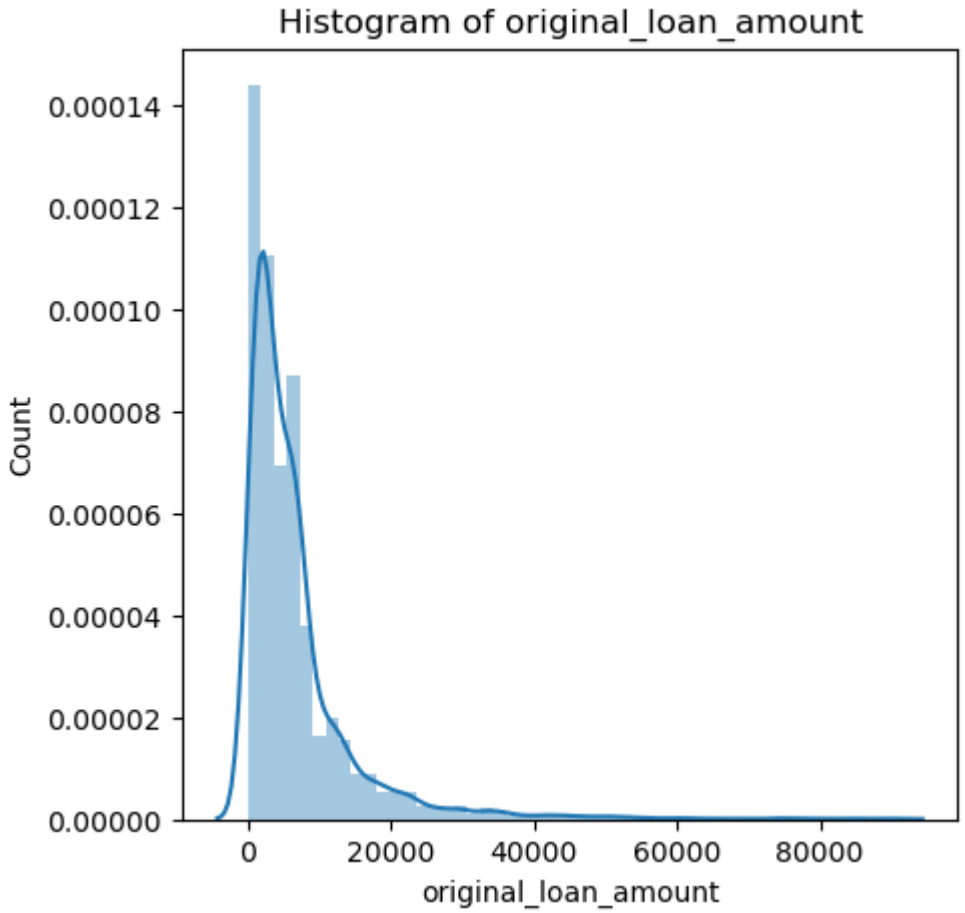

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



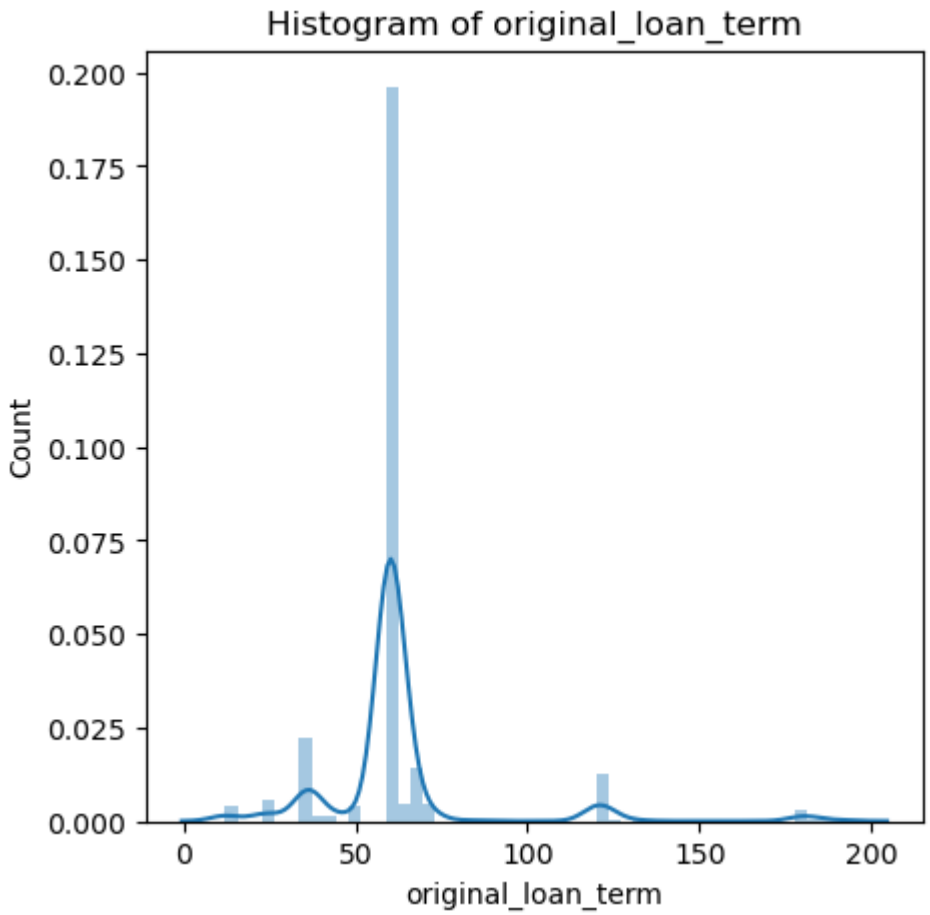

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



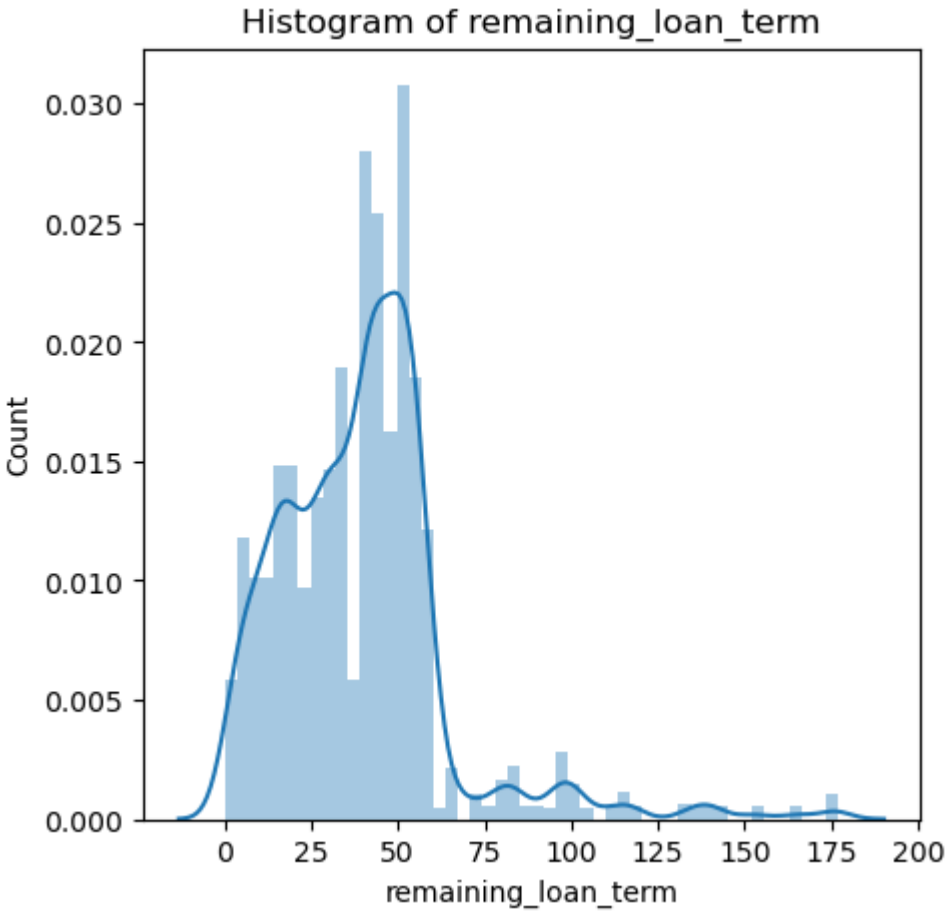

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



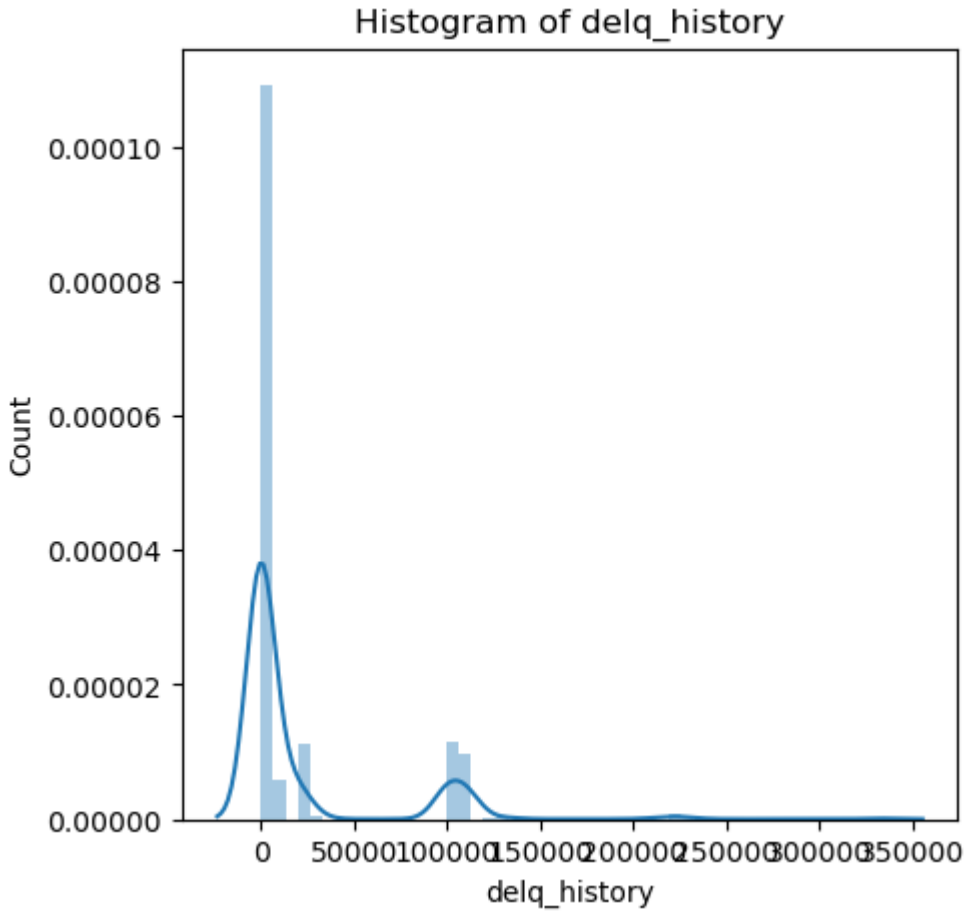

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



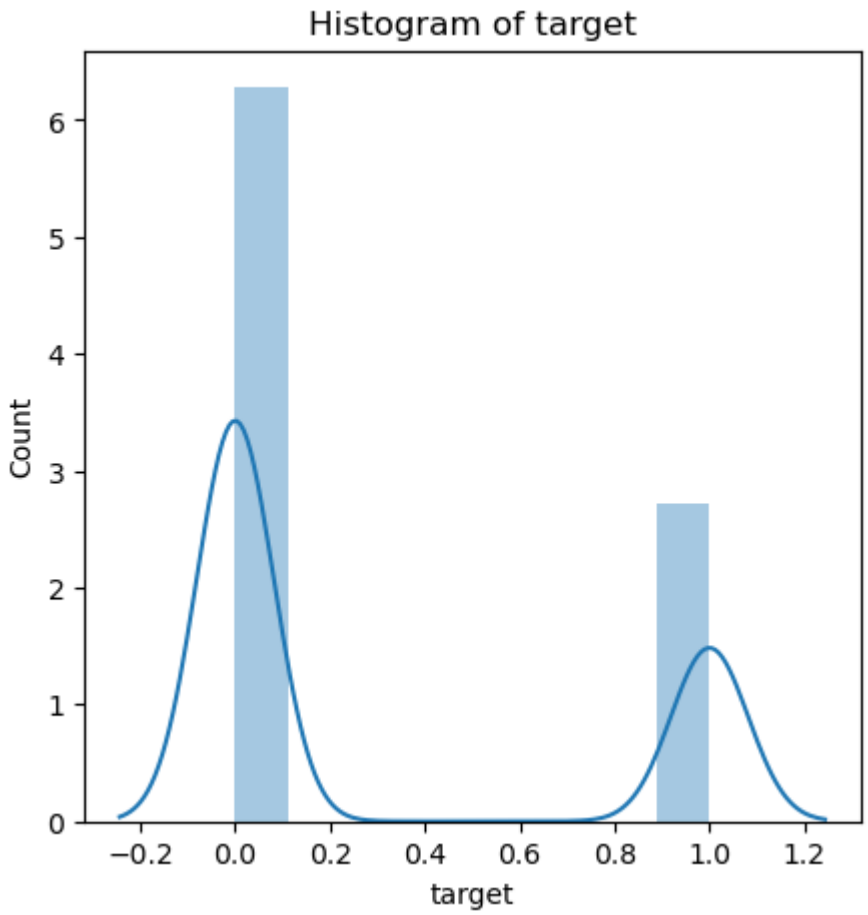

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```




```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

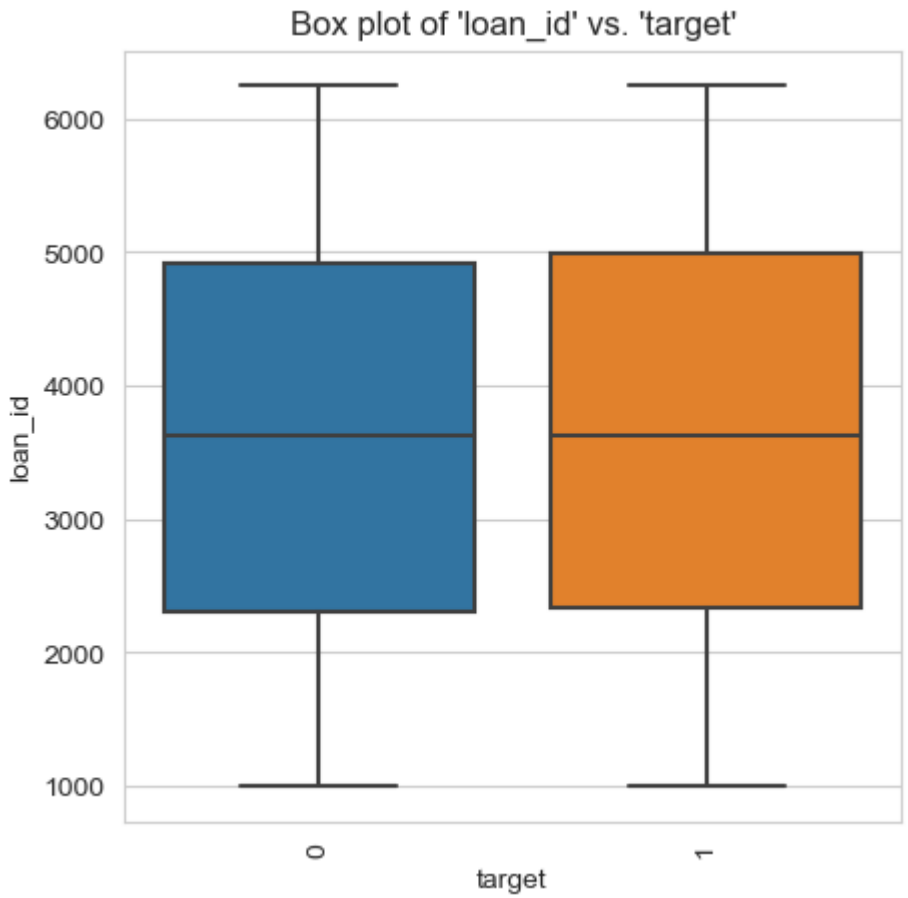



```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

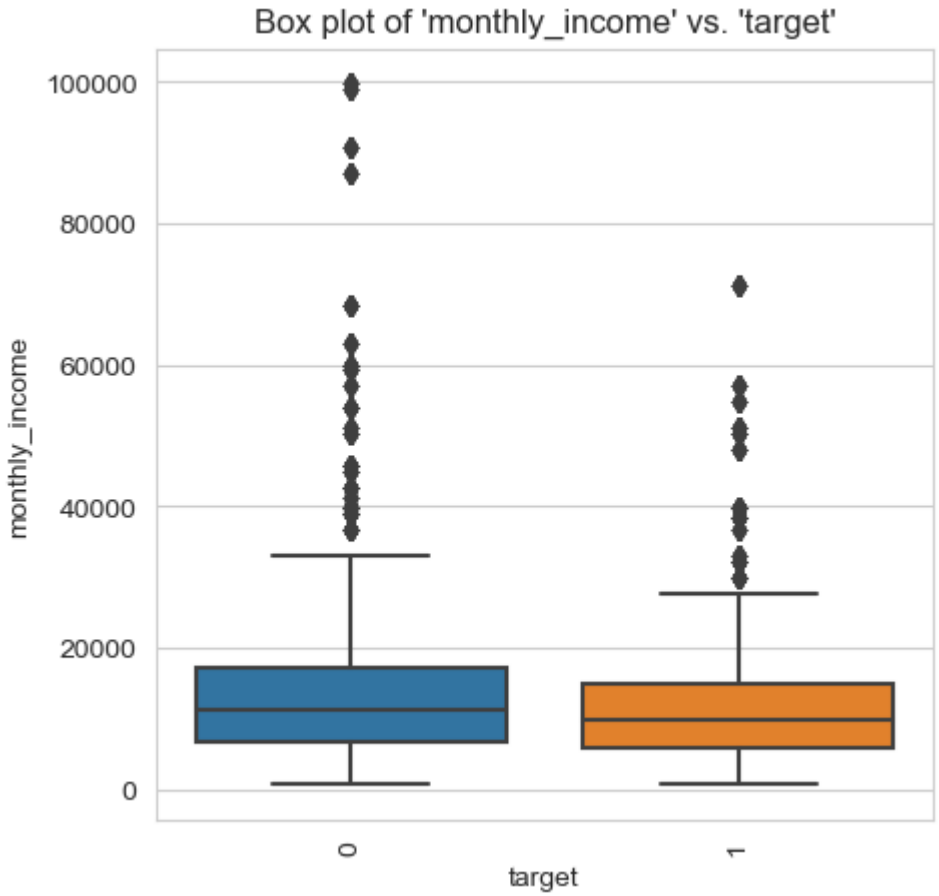


In [25]:
```
ds.visualizations.boxplot(data=data_risk, target='target', fig_size=(5,5))
```

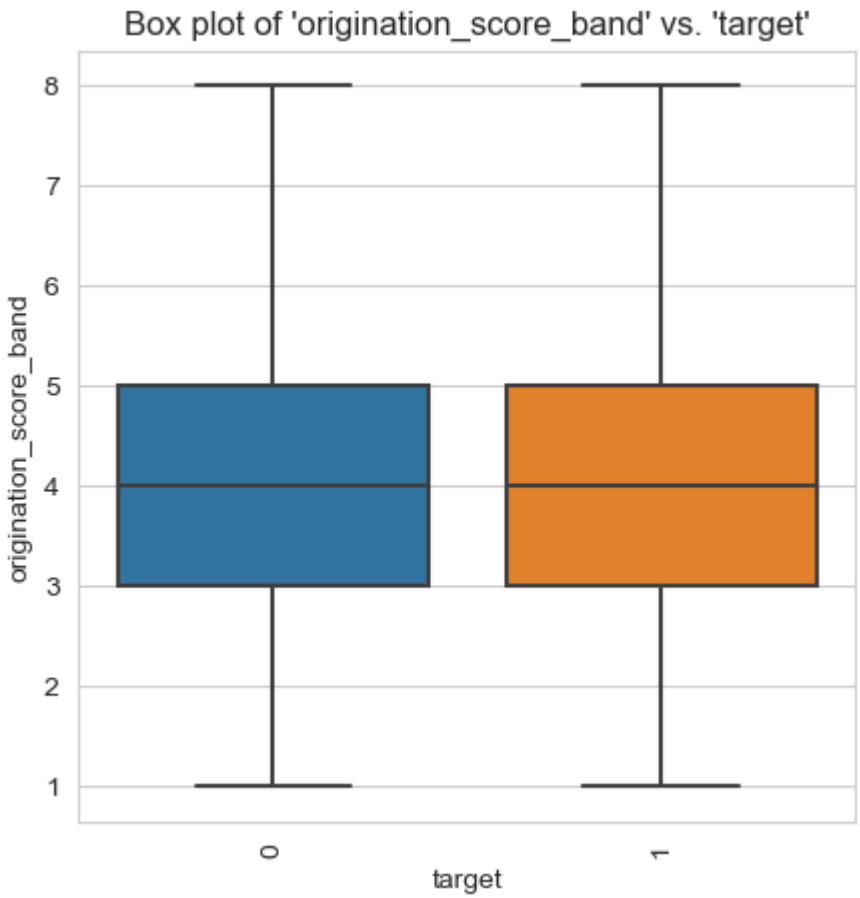
```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



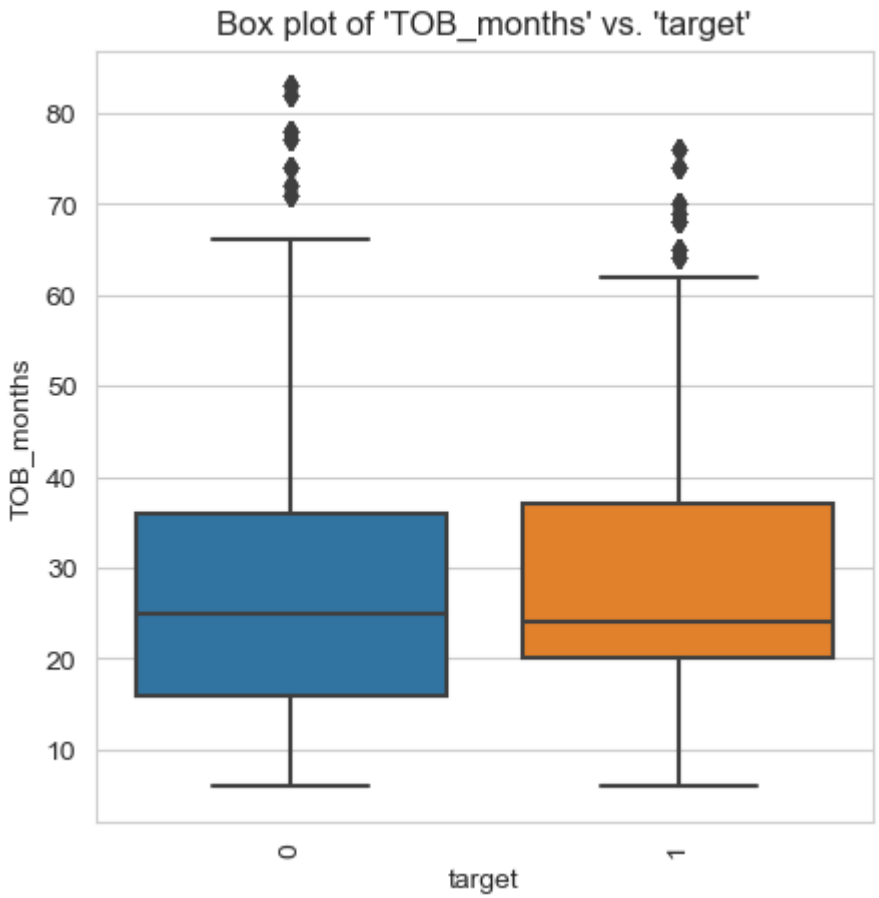

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



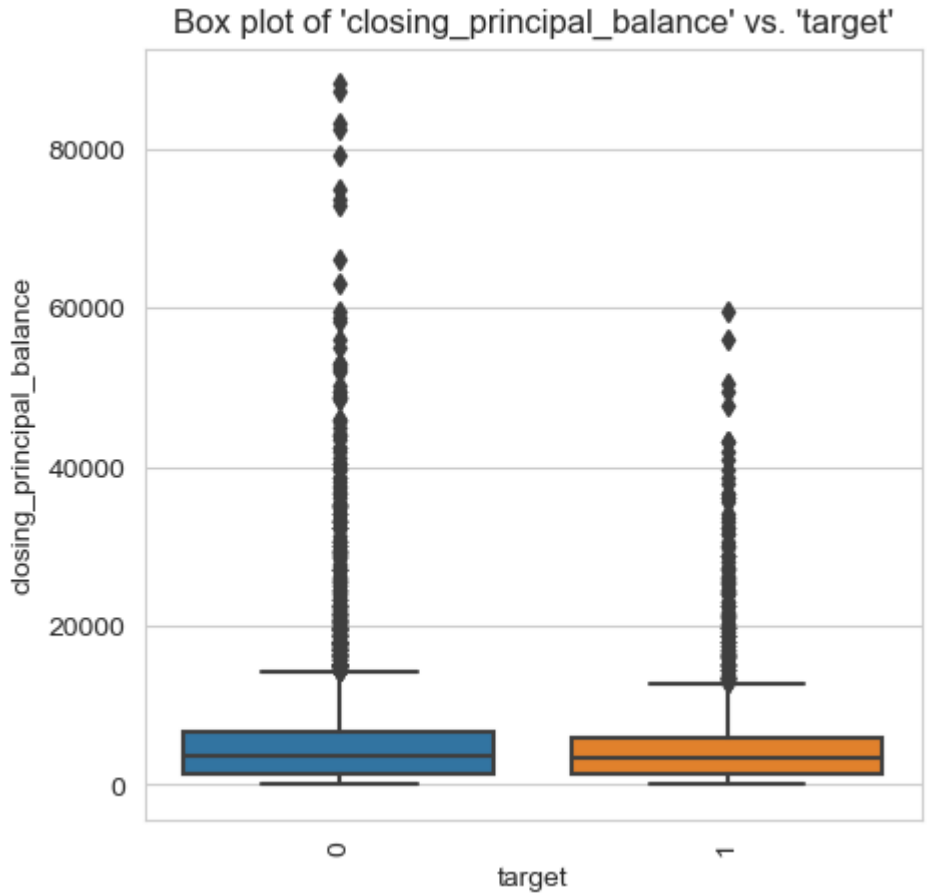

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



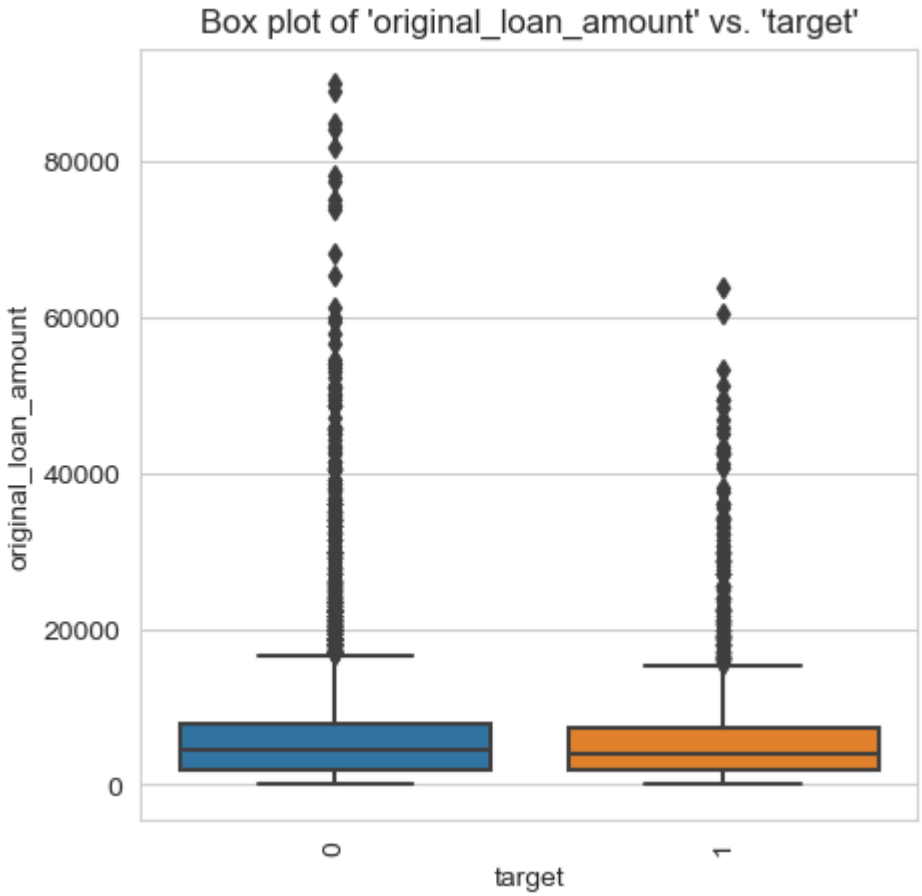

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



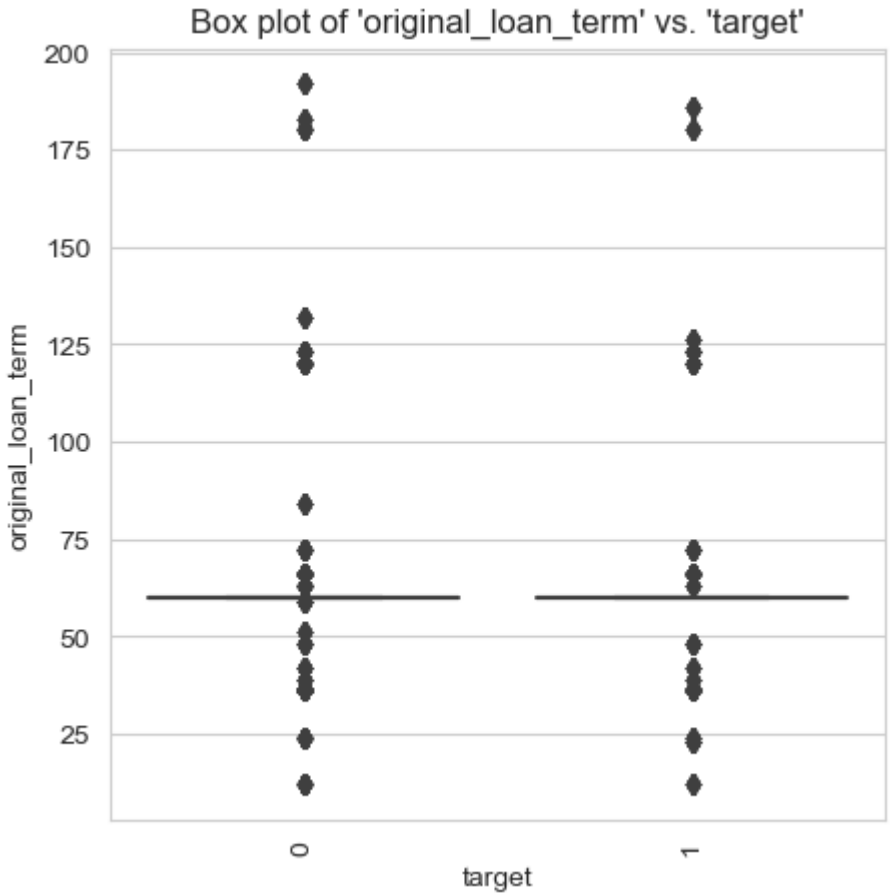

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



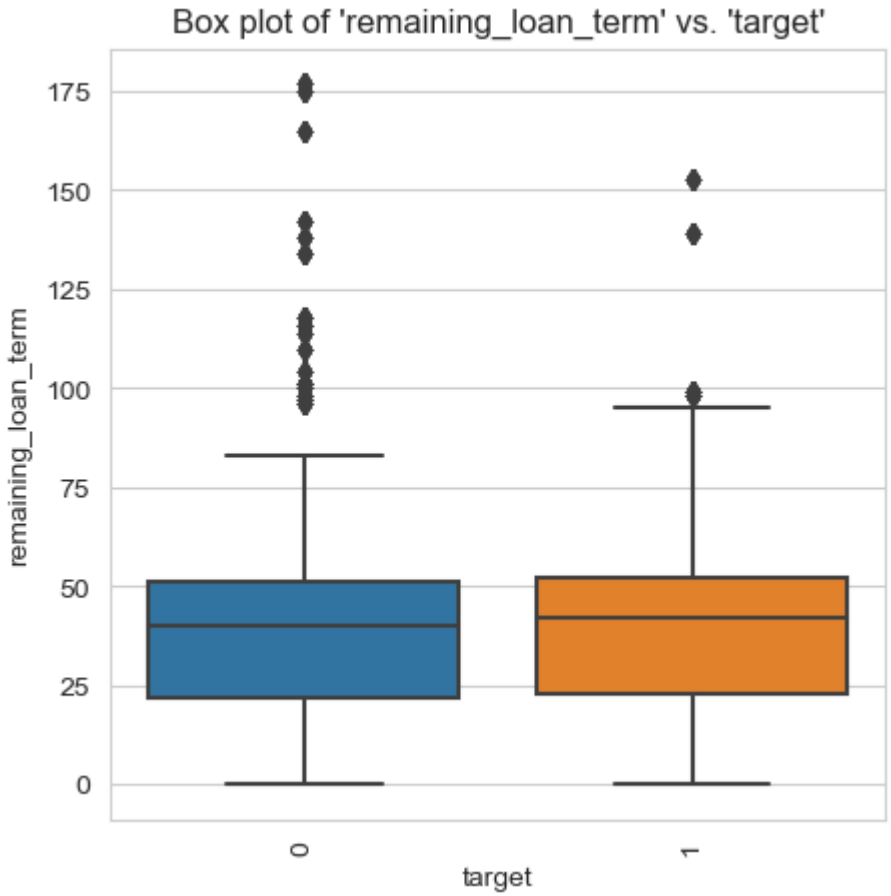

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



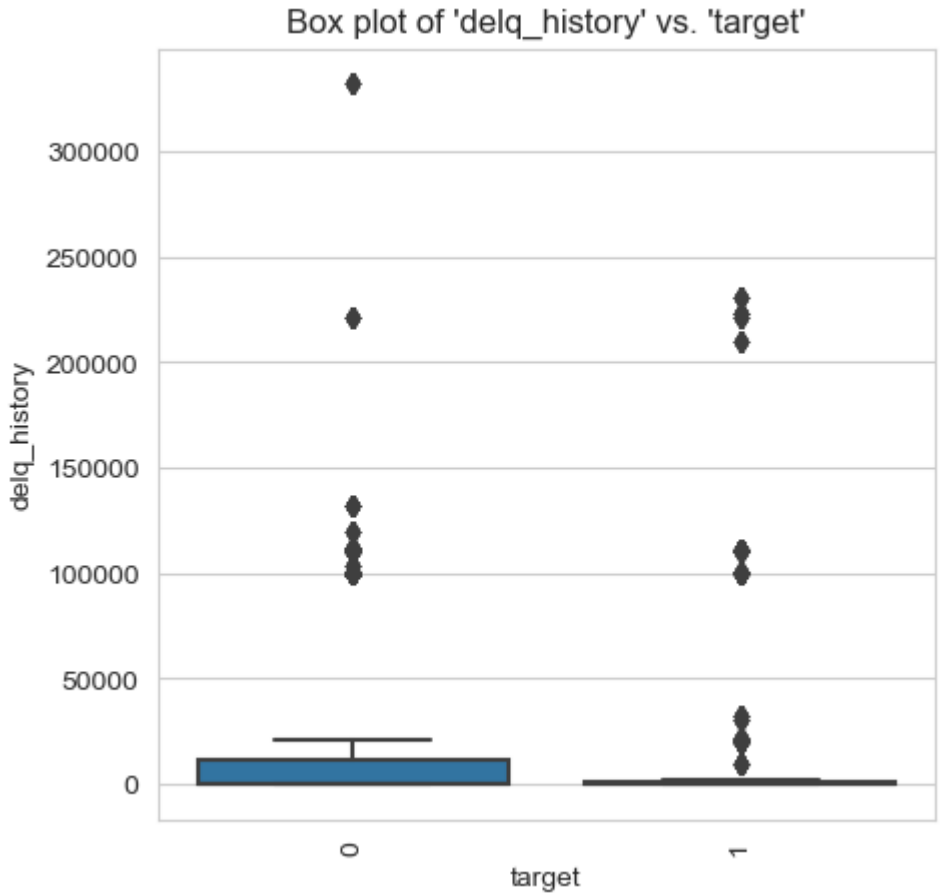

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



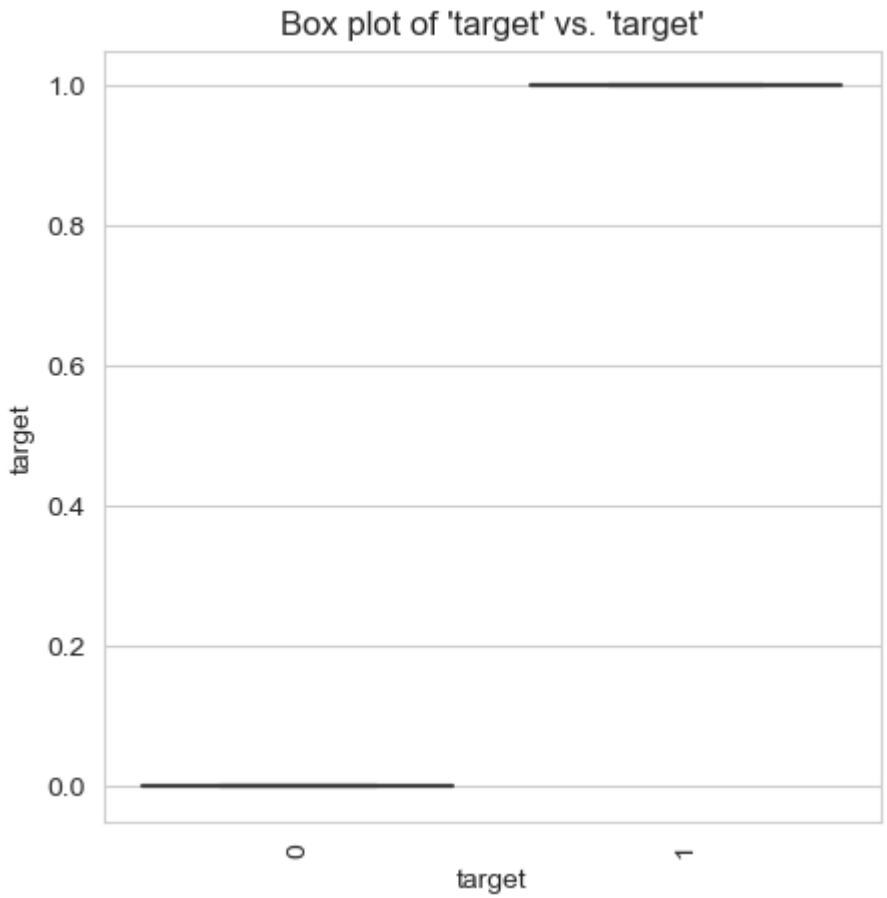

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```




```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```




```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



Handling categorical data - Method 2 (Using the label encoder)

In [47]:
```
## Using the label encoder
from sklearn.preprocessing import LabelEncoder
```

In [48]:
```
data_risk.head(2)
```

Out[48]:

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	original_l
0	1000	6000.0	5	83.0	300.0	
1	1001	39000.0	5	82.0	7200.0	

In [49]:
```
df = data_risk
```

In [50]:
```
df.head(2)
```

Out[50]:

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	original_l
0	1000	6000.0	5	83.0	300.0	
1	1001	39000.0	5	82.0	7200.0	

In [51]:
```
le = LabelEncoder()
```

In [52]:
```
label = le.fit_transform(df["product"])
```

In [53]:
```
le.classes_
```

Out[53]:
```
array(['A', 'B', 'C', 'D'], dtype=object)
```

In [54]:
```
df = df.drop("product", axis ='columns')
```

In [55]:
```
df.head(5)
```

Out[55]:

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	original_l
0	1000	6000.0	5	83.0	300.0	
1	1001	39000.0	5	82.0	7200.0	
2	1002	18000.0	5	78.0	2700.0	
3	1003	23250.0	3	76.0	3900.0	
4	1004	12000.0	3	74.0	2100.0	

Appending/joining the table

In [56]:
```
df["product"] = label
```

In [57]:
```
df.head(5)
```

Out[57]:

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	original_l
0	1000	6000.0	5	83.0	300.0	
1	1001	39000.0	5	82.0	7200.0	
2	1002	18000.0	5	78.0	2700.0	
3	1003	23250.0	3	76.0	3900.0	
4	1004	12000.0	3	74.0	2100.0	

In [103…
```
df['product'].unique
```

Out[103]:
```
<bound method Series.unique of 0       1
1       1
2       1
3       1
4       1
       ..
5778    0
5779    2
5780    3
5781    3
5782    2
Name: product, Length: 5783, dtype: int32>
```

Normalizing Dataset

In [58]:
```
## df["acb"] = df["acb"]/df["acb"].max() -- Simple scaling
```

In [59]:
```
df.columns
```

Out[59]:
```
Index(['loan_id', 'monthly_income', 'origination_score_band', 'TOB_months',
       'closing_principal_balance', 'original_loan_amount',
       'original_loan_term', 'remaining_loan_term', 'delq_history', 'target',
       'product'],
      dtype='object')
```

Creating x_train and y_train

In [60]:
```
x_Data = df[['loan_id', 'monthly_income', 'origination_score_band', 'TOB_months',
       'closing_principal_balance', 'original_loan_amount',
       'original_loan_term', 'remaining_loan_term', 'product','target']]
```

In [61]:
```
## Method 1 Normalizing
```

In [62]:
```
x_Data.head(2)
```

Out[62]:

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	original_l
0	1000	6000.0	5	83.0	300.0	
1	1001	39000.0	5	82.0	7200.0	

Using MinMaxScaler

In [63]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [64]:
```
scaler = MinMaxScaler()
scaler.fit(x_Data)
N_df = scaler.transform(x_Data)
new_df = pd.DataFrame(N_df, columns=(['loan_id', 'monthly_income', 'origination_score_
       'closing_principal_balance', 'original_loan_amount',
       'original_loan_term', 'remaining_loan_term', 'product','target']))
new_df.head(5)
```

Out[64]:

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	original
0	0.000000	0.053030	0.571429	1.000000	0.002270	
1	0.000191	0.386364	0.571429	0.987013	0.080590	
2	0.000381	0.174242	0.571429	0.935065	0.029512	
3	0.000572	0.227273	0.285714	0.909091	0.043133	
4	0.000762	0.113636	0.285714	0.883117	0.022701	

In [65]:
```
new_df.info()
new_df.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5783 entries, 0 to 5782
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   loan_id                   5783 non-null   float64
 1   monthly_income            5783 non-null   float64
 2   origination_score_band    5783 non-null   float64
 3   TOB_months                5783 non-null   float64
 4   closing_principal_balance  5783 non-null   float64
 5   original_loan_amount      5783 non-null   float64
 6   original_loan_term        5783 non-null   float64
 7   remaining_loan_term       5783 non-null   float64
 8   product                   5783 non-null   float64
 9   target                    5783 non-null   float64
dtypes: float64(10)
memory usage: 451.9 KB
```

Out[65]:
```
(5783, 10)
```

In [66]:
```
ds.structdata.describe(new_df)
```

```
First five data points
```

Out[66]:

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	original
0	0.000000	0.053030	0.571429	1.000000	0.002270	
1	0.000191	0.386364	0.571429	0.987013	0.080590	
2	0.000381	0.174242	0.571429	0.935065	0.029512	
3	0.000572	0.227273	0.285714	0.909091	0.043133	
4	0.000762	0.113636	0.285714	0.883117	0.022701	

```
Random five data points
```

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	origi
5137	0.887788	0.053030	0.714286	0.129870	0.014529	
4798	0.831206	0.196970	0.428571	0.428571	0.040636	
886	0.153172	0.212121	0.142857	0.090909	0.106697	
3617	0.626977	0.234848	0.285714	0.519481	0.055619	
1651	0.285197	0.378788	0.571429	0.155844	0.038593	

Last five data points

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	origi
5778	0.999238	0.075758	0.428571	0.0	0.066969	
5779	0.999428	0.015152	0.571429	0.0	0.015891	
5780	0.999619	0.030303	0.571429	0.0	0.032917	
5781	0.999809	0.090909	0.857143	0.0	0.083995	
5782	1.000000	0.015152	0.714286	0.0	0.015891	

```
Shape of  data set: (5783, 10)
```

```
Size of  data set: 57830
```

```
Data Types
Note: All Non-numerical features are identified as objects in pandas
```

	Data Type
loan_id	float64
monthly_income	float64
origination_score_band	float64
TOB_months	float64
closing_principal_balance	float64
original_loan_amount	float64
original_loan_term	float64
remaining_loan_term	float64
product	float64
target	float64

```
Numerical Features in Data set
['loan_id', 'monthly_income', 'origination_score_band', 'TOB_months', 'closing_princi
pal_balance', 'original_loan_amount', 'original_loan_term', 'remaining_loan_term', 'p
roduct', 'target']


Categorical Features in Data set
[]
```

Statistical Description of Columns

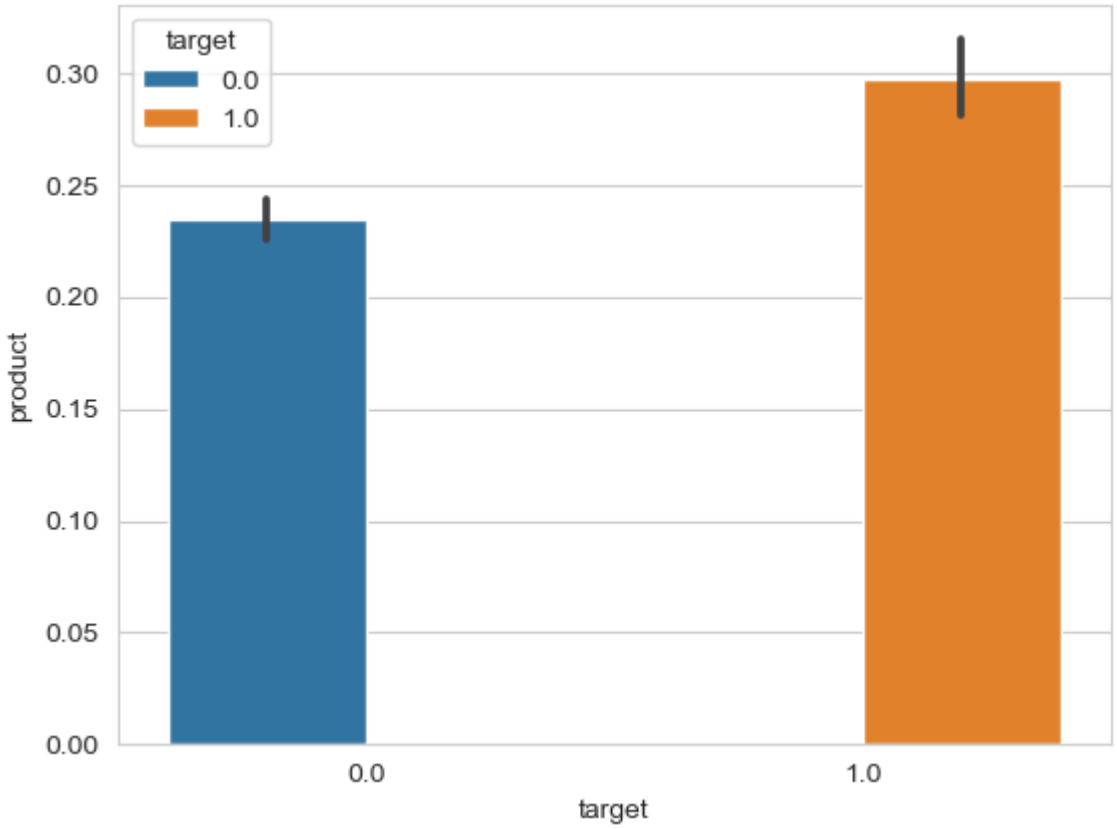
	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance
count	5783.000000	5783.000000	5783.000000	5783.000000	5783.000000
mean	0.500103	0.137014	0.414861	0.290723	0.063727
std	0.288873	0.134768	0.212982	0.205393	0.086012
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.249667	0.053030	0.285714	0.149351	0.015891
50%	0.499714	0.090909	0.428571	0.246753	0.038593
75%	0.749571	0.151515	0.571429	0.396104	0.072361
max	1.000000	1.000000	1.000000	1.000000	1.000000

Description of Categorical Features

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_20520\473831064.py in <module>
----> 1 ds.structdata.describe(new_df)

~\anaconda3\lib\site-packages\datasist\structdata.py in describe(data, name, date_col
s, show_categories, plot_missing)
     97     print('Description of Categorical Features')
     98     if cat_features != None:
---> 99         display(data.describe(include=[np.object, pd.Categorical]).T)
    100         _space()
    101

~\anaconda3\lib\site-packages\pandas\core\generic.py in describe(self, percentiles, i
nclude, exclude, datetime_is_numeric)
  10230         max            NaN      3.0
  10231         """
> 10232         return describe_ndframe(
  10233             obj=self,
  10234             include=include,

~\anaconda3\lib\site-packages\pandas\core\describe.py in describe_ndframe(obj, includ
e, exclude, datetime_is_numeric, percentiles)
     92         )
     93
---> 94     result = describer.describe(percentiles=percentiles)
     95     return cast(NDFrameT, result)
     96

~\anaconda3\lib\site-packages\pandas\core\describe.py in describe(self, percentiles)
    175
    176         col_names = reorder_columns(ldesc)
--> 177         d = concat(
    178             [x.reindex(col_names, copy=False) for x in ldesc],
    179             axis=1,

~\anaconda3\lib\site-packages\pandas\util\_decorators.py in wrapper(*args, **kwargs)
    309                     stacklevel=stacklevel,
    310                 )
--> 311             return func(*args, **kwargs)
    312
    313         return wrapper

~\anaconda3\lib\site-packages\pandas\core\reshape\concat.py in concat(objs, axis, joi
n, ignore_index, keys, levels, names, verify_integrity, sort, copy)
    345     ValueError: Indexes have overlapping values: ['a']
    346     """
--> 347     op = _Concatenator(
    348         objs,
    349         axis=axis,

~\anaconda3\lib\site-packages\pandas\core\reshape\concat.py in __init__(self, objs, a
xis, join, keys, levels, names, ignore_index, verify_integrity, copy, sort)
    402
    403         if len(objs) == 0:
--> 404             raise ValueError("No objects to concatenate")
    405
    406         if keys is None:

ValueError: No objects to concatenate
```

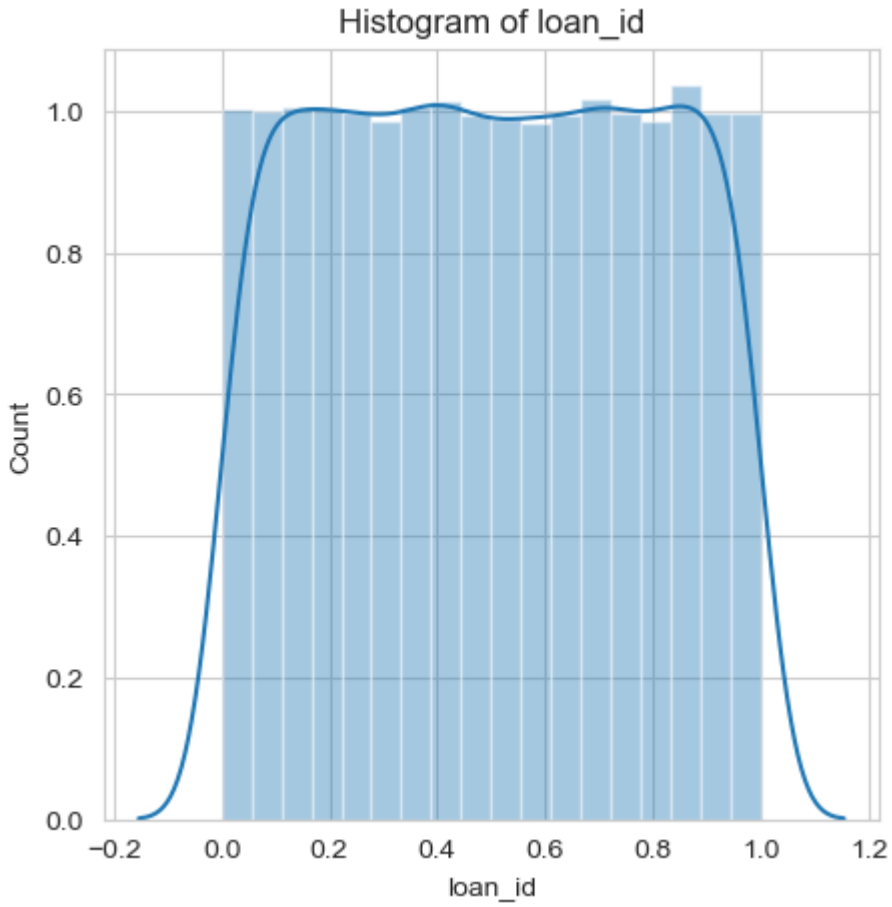
In [67]:
```
sns.barplot(data=new_df, x="target", y="product", hue="target")
```

Out[67]:
```
<AxesSubplot:xlabel='target', ylabel='product'>
```

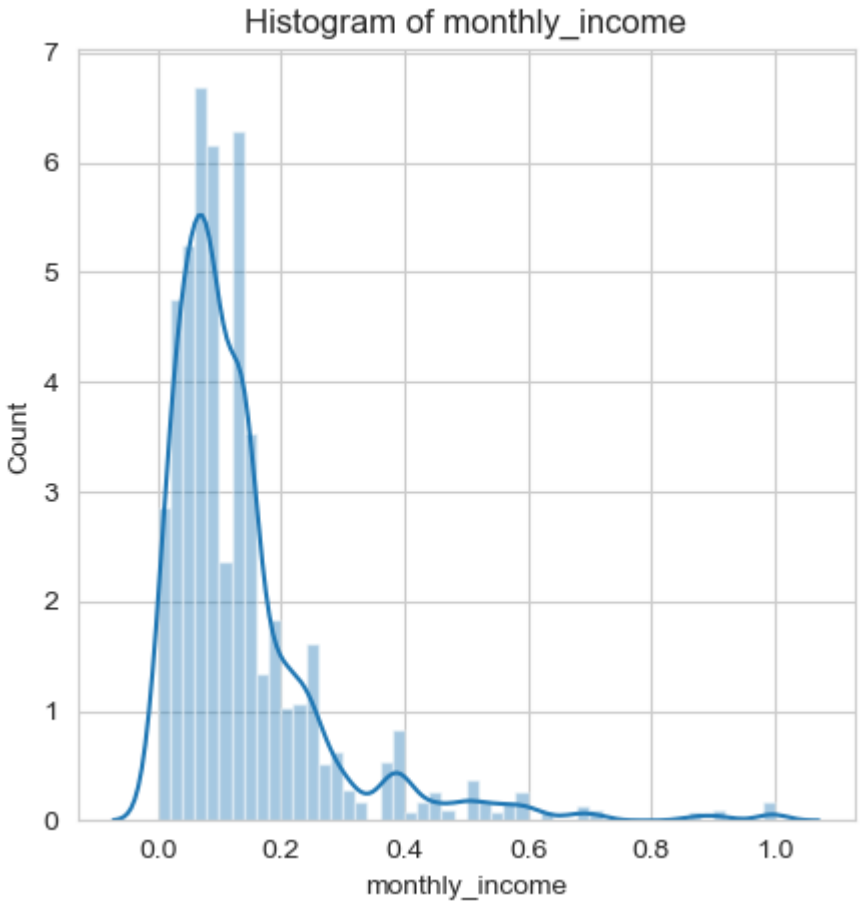


In [68]:
```
ds.visualizations.histogram(new_df)
```

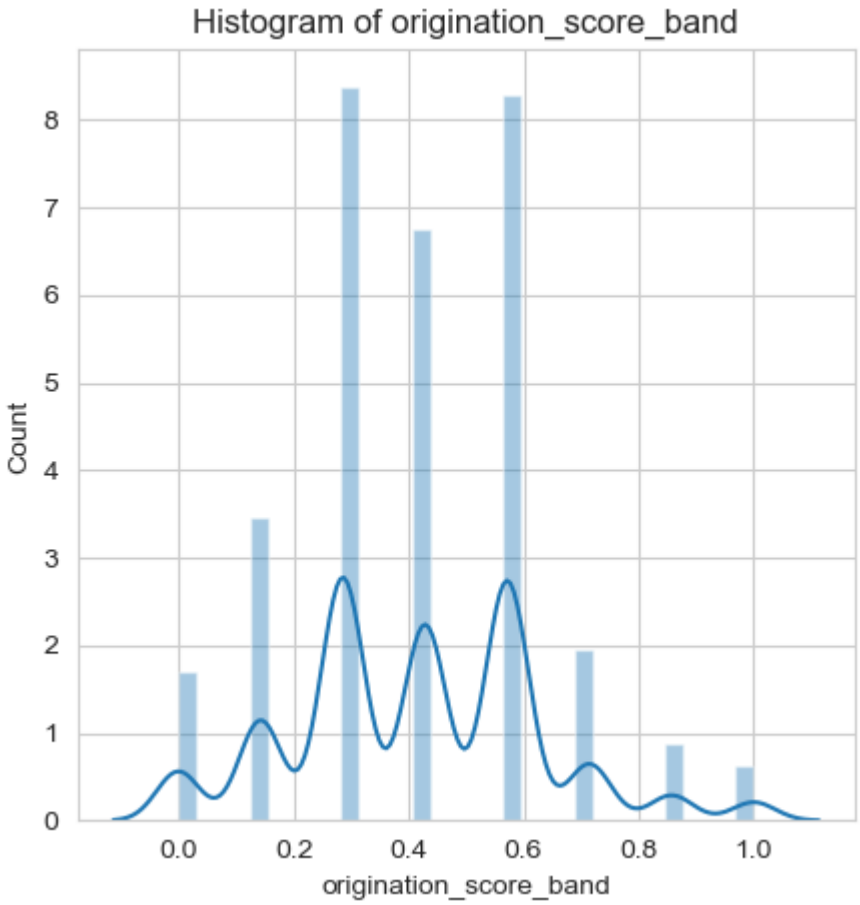
```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng:

`distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
```



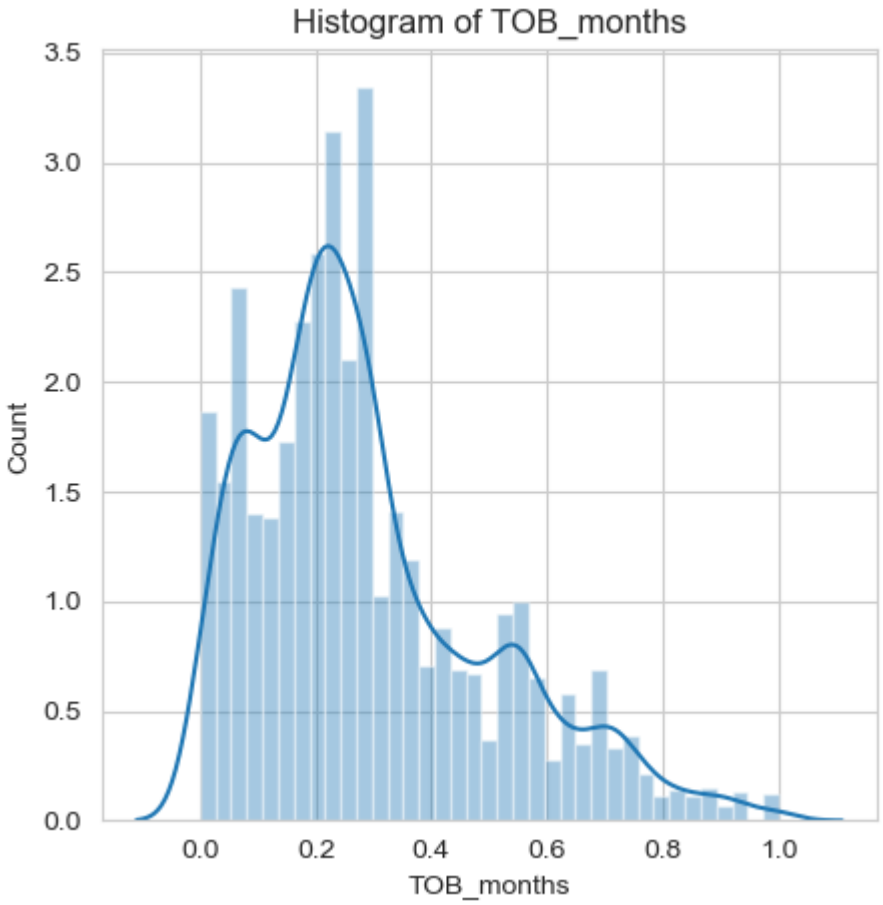

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng:

`distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
```



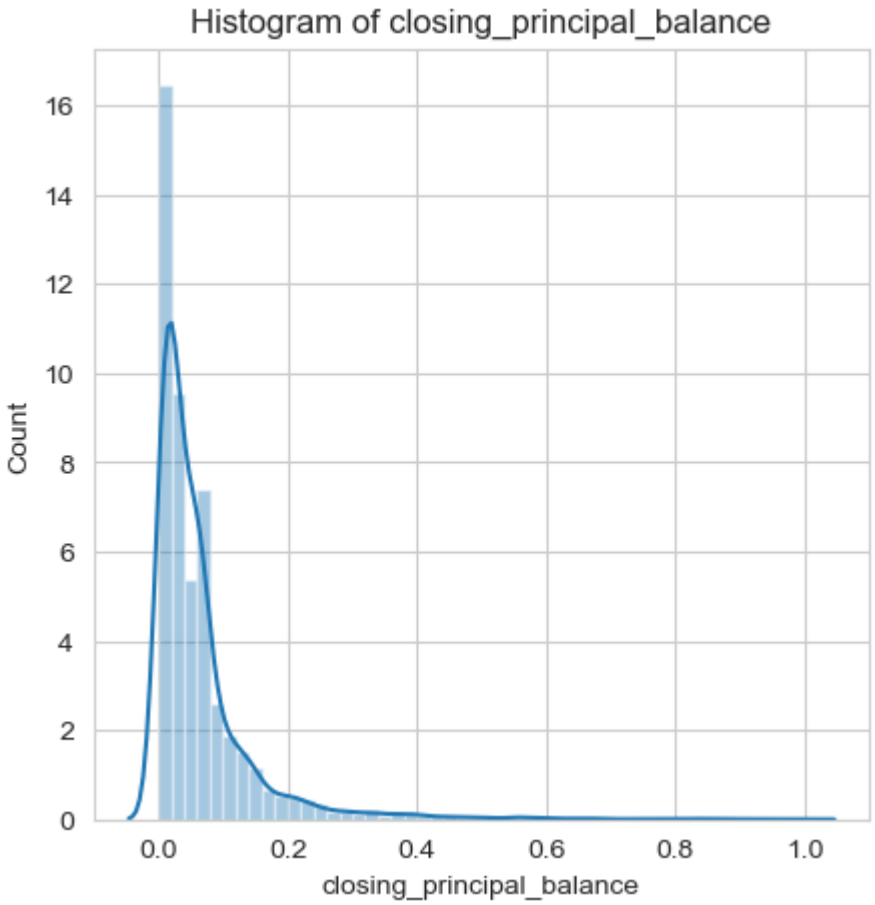

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng:

`distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
```



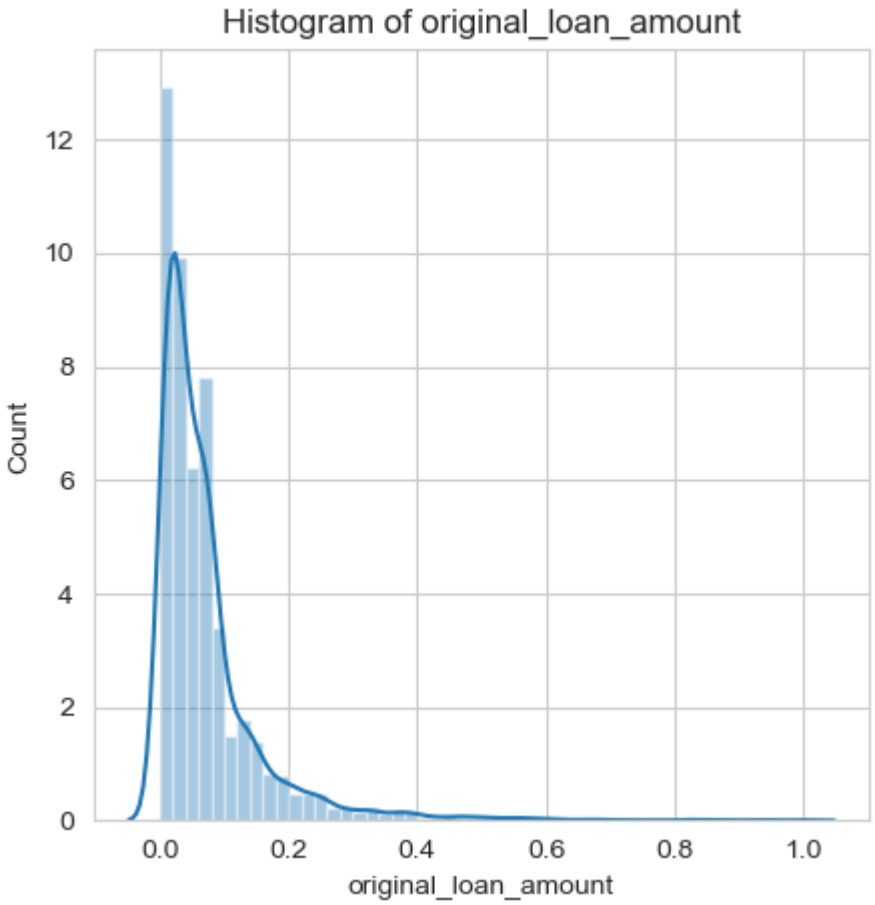

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng:

`distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
```



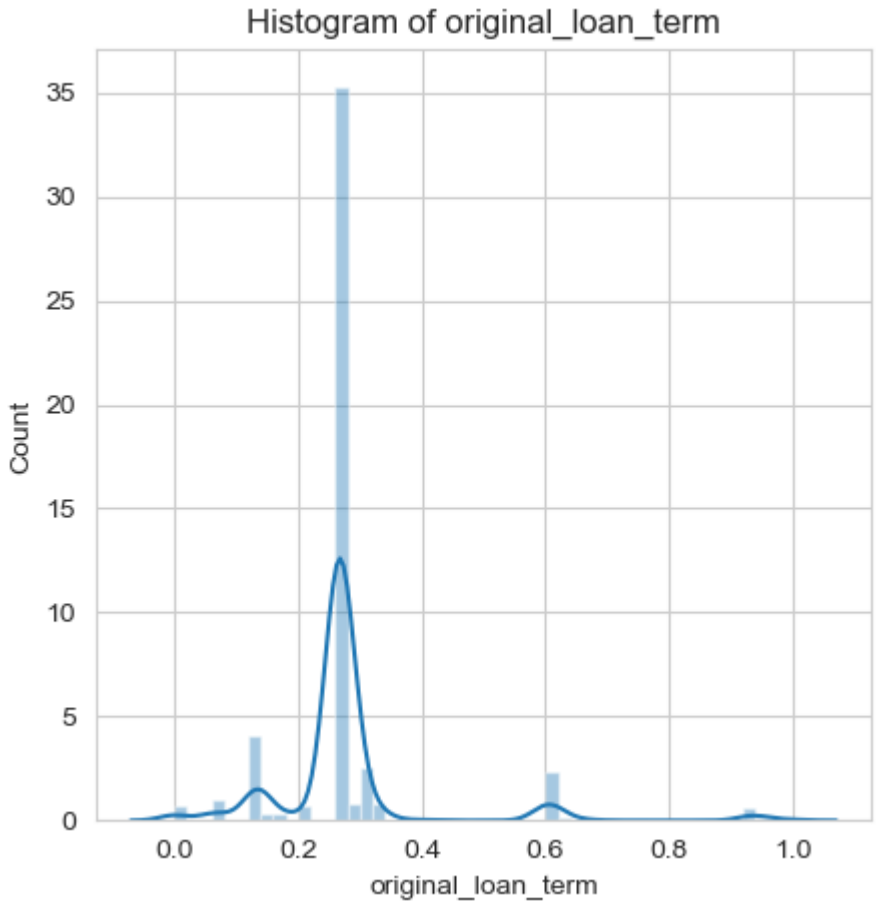

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng:

`distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
```



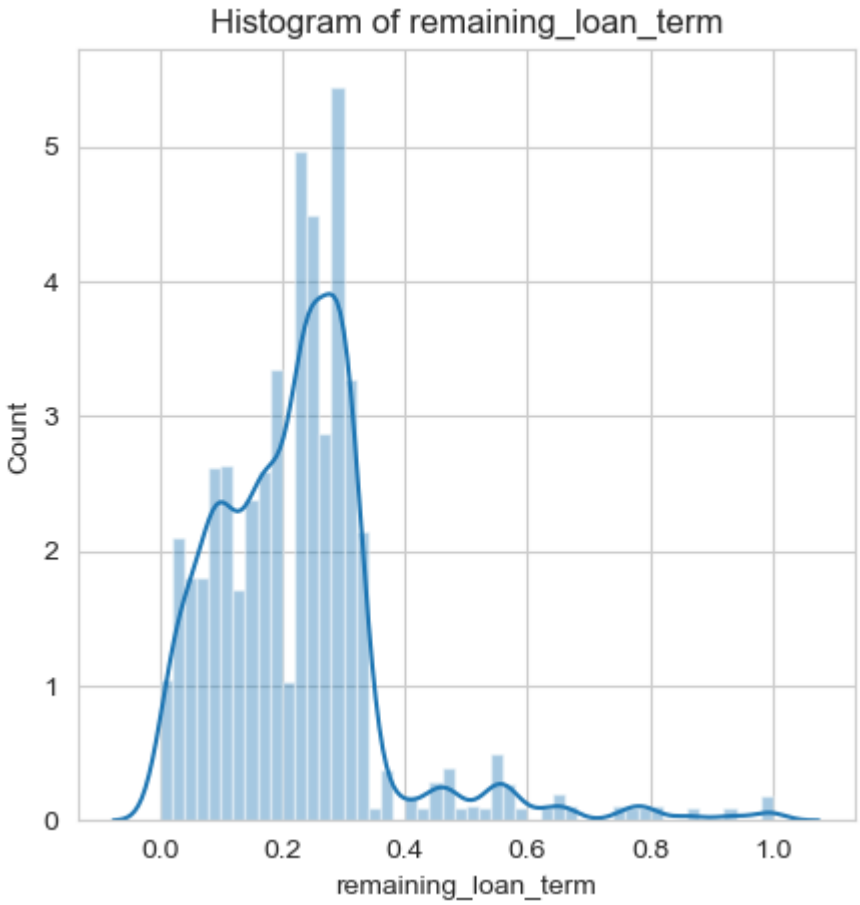

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng:

`distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
```



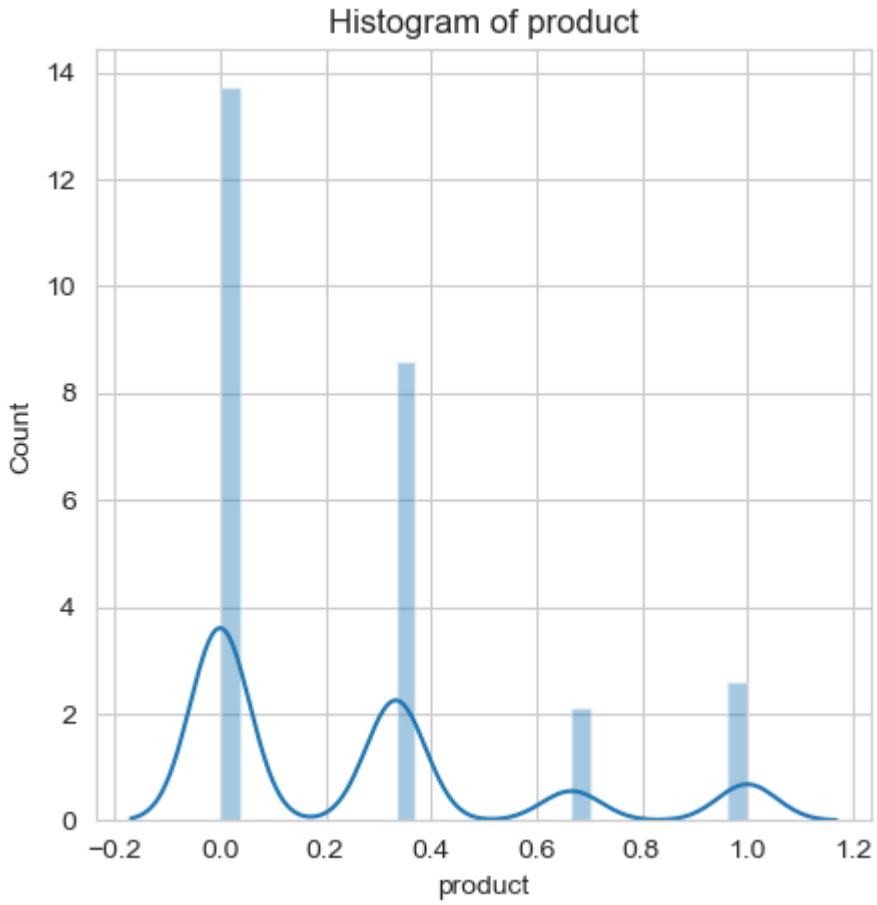

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng:

`distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
```



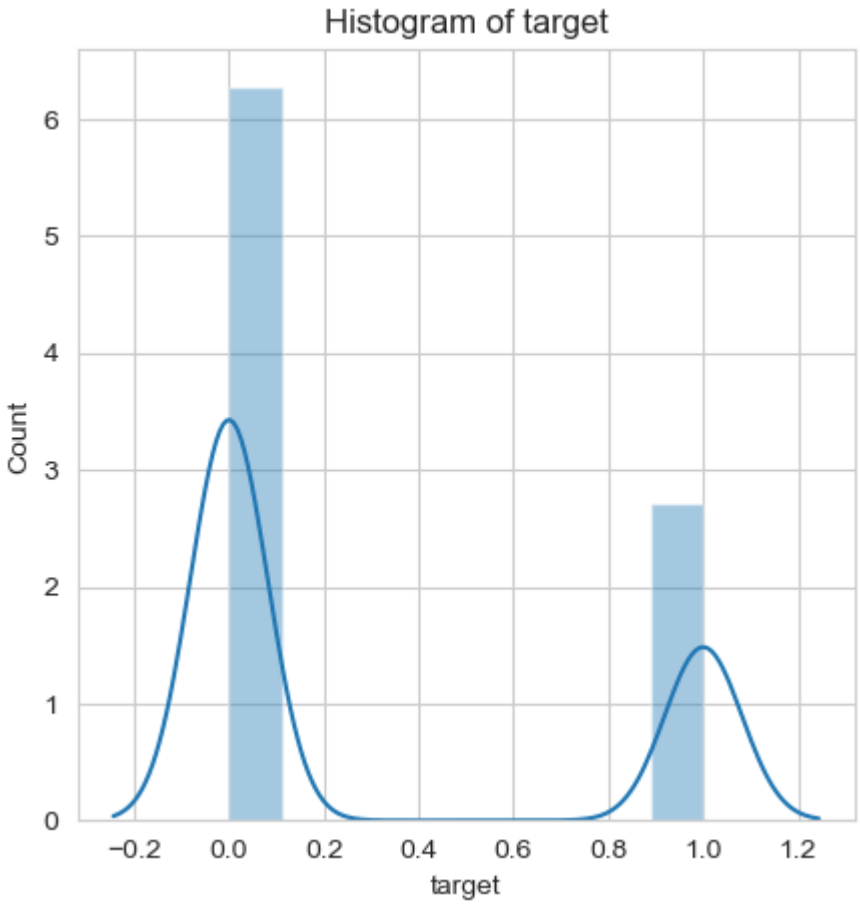

```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng:

`distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
```




```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng:

`distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
```

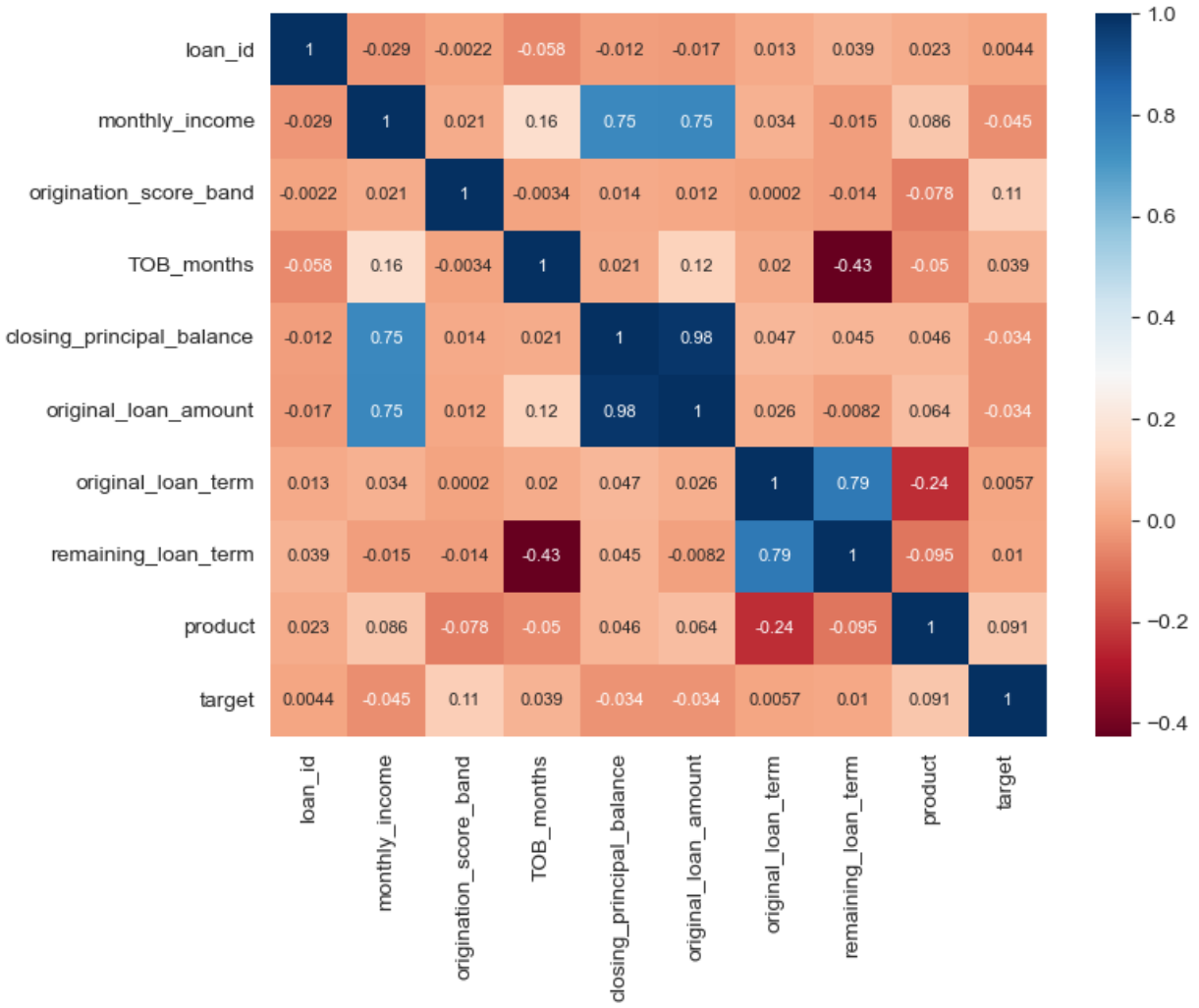



```
C:\Users\jeana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng:

`distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
```



In [69]:
```
corr = new_df.corr()
plt.figure(figsize=(8,6))
sns.heatmap(corr, annot = True, cmap='RdBu', annot_kws={'fontsize':8})
```

Out[69]:
```
<AxesSubplot:>
```



Spliting Dataset into train and test

In [70]:
```
## Determing x and y ( input and ouput) and using he coulumn ; target as the predictor
```

In [71]:
```
new_df.columns
```

Out[71]:
```
Index(['loan_id', 'monthly_income', 'origination_score_band', 'TOB_months',
       'closing_principal_balance', 'original_loan_amount',
       'original_loan_term', 'remaining_loan_term', 'product', 'target'],
      dtype='object')
```

In [72]:
```
cols = new_df.columns
```

In [73]:
```
x = new_df.drop("target", axis ='columns')
x.head(2)
```

Out[73]:

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	original_
0	0.000000	0.053030	0.571429	1.000000	0.00227	
1	0.000191	0.386364	0.571429	0.987013	0.08059	

In [74]:
```
y = new_df['target']

y
```

Out[74]:
```
0       0.0
1       0.0
2       0.0
3       1.0
4       0.0
       ...
5778    0.0
5779    0.0
5780    0.0
5781    1.0
5782    0.0
Name: target, Length: 5783, dtype: float64
```

In [75]:
```
from sklearn.model_selection import train_test_split
```

In [76]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state
```

In [77]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(4337, 9)
(1446, 9)
(4337,)
(1446,)
```

In [78]:
```
y_train.head(5)
```

Out[78]:
```
5731    0.0
2481    0.0
2902    1.0
5452    0.0
1211    0.0
Name: target, dtype: float64
```

In [79]:
```
x_train.head(5)
```

Out[79]:

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	origi
5731	0.990665	0.393939	0.428571	0.064935	0.339387	
2481	0.427891	0.136364	0.285714	0.493506	0.124291	
2902	0.501429	0.507576	0.285714	0.831169	0.284904	
5452	0.942656	0.137014	0.428571	0.290723	0.063727	
1211	0.209754	0.075758	0.571429	0.532468	0.006810	

In [80]:
```
y_test.head(5)
```

Out[80]:
```
1886    0.0
2067    1.0
3334    1.0
4065    0.0
156     0.0
Name: target, dtype: float64
```

In [81]:
```
x_test.head(5)
```

Out[81]:

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	origi
1886	0.326348	0.136364	0.428571	0.532468	0.111237	
2067	0.357782	0.136364	0.571429	0.532468	0.121453	
3334	0.577443	0.137014	0.714286	0.290723	0.063727	
4065	0.704706	0.121212	0.285714	0.701299	0.009081	
156	0.027243	0.196970	0.285714	0.493506	0.063564	

In [82]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,ExtraTreesClassifier
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
```

In [83]:
```
## Using RandomForestClassifier and LogisticRegression
rf_model = RandomForestClassifier(n_estimators=100, random_state= 0)
lg_model = LogisticRegression(max_iter=100, random_state=2, solver='lbfgs')
```

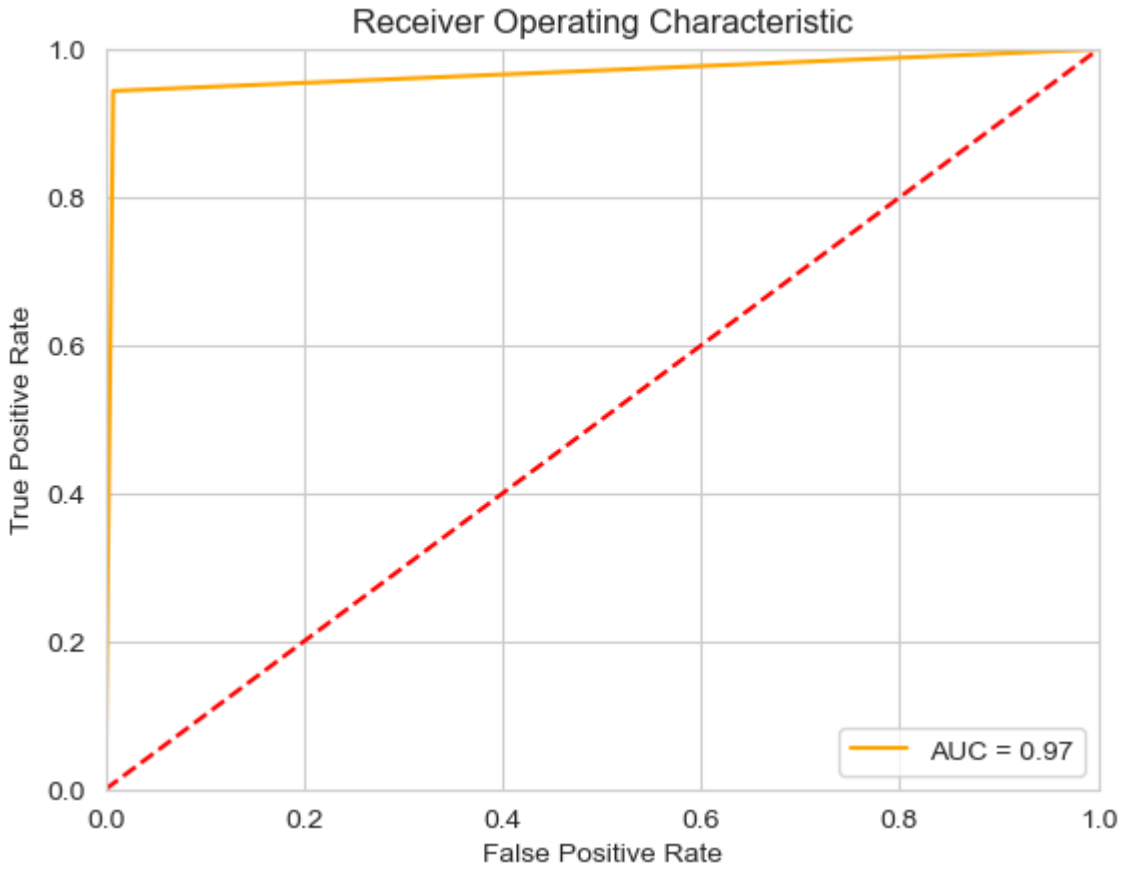
In [84]:
```
##Using RandomForestClassifier
rf_model.fit(x_train, y_train)
y_pred = rf_model.predict(x_test)
```

In [85]:
```
y_pred
```

Out[85]:
```
array([0., 1., 0., ..., 0., 1., 1.])
```

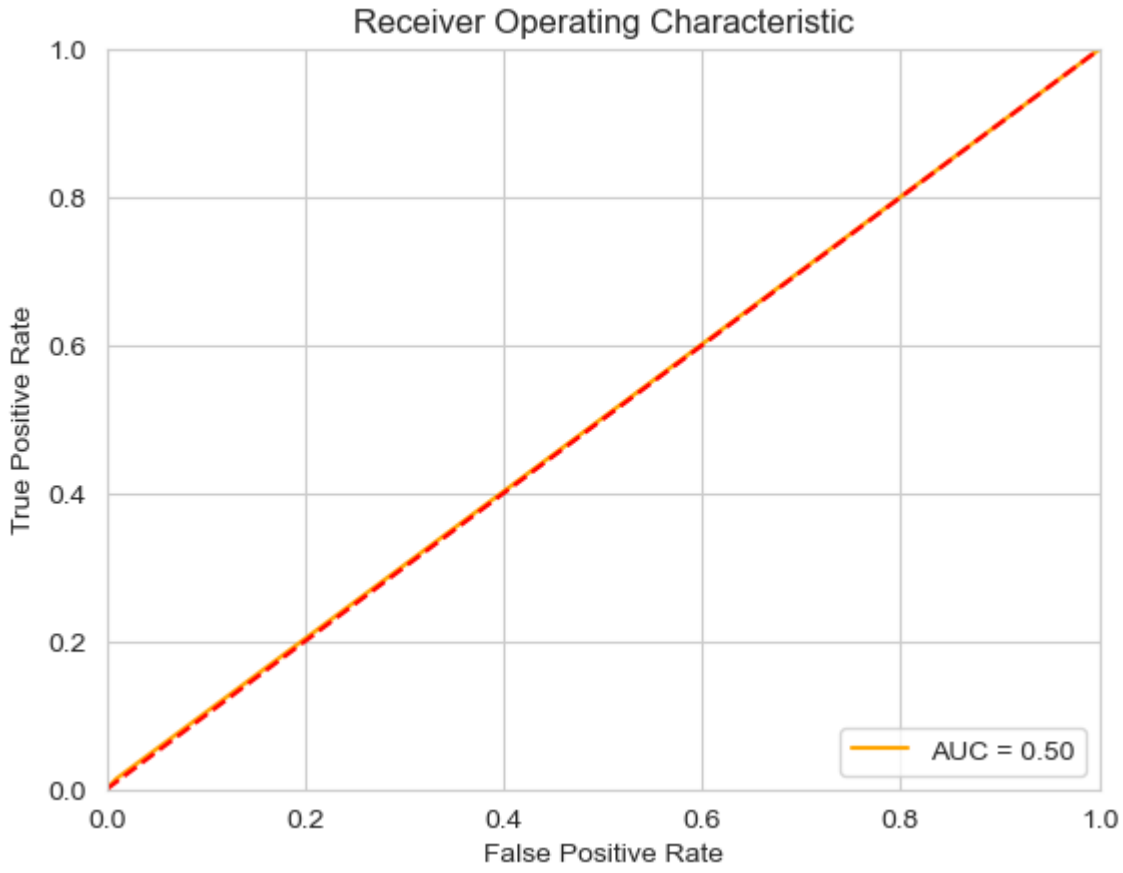
In [86]:
```
ds.model.get_classification_report(y_test, y_pred)
```

```
Accuracy is  98
F1 score is  96
Precision is  98
Recall is  94
********************************************************************************
***************
confusion Matrix
                 Score positive    Score negative
Actual positive      1011                 8
Actual negative        24               403
```



In [87]:
```
## Using LogisticRegression
lg_model.fit(x_train, y_train)
y_pred = lg_model.predict(x_test)
ds.model.get_classification_report(y_test, y_pred)
```

```
Accuracy is  70
F1 score is  3
Precision is  38
Recall is  1
********************************************************************************
***************
confusion Matrix
                 Score positive    Score negative
Actual positive      1009                10
Actual negative       421                 6
```


Receiver Operating Characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
AUC = 0.50

In [ ]: