Python初級數據分析員證書

（六）數據分析 及可視化專案

13. 數據分析專案

Demo 14 – Loan Data

Review

- Statistics
- Hypothesis testing
- Algebra
- Linear regression
- Propositional logic
- Python
- R
- SQL

- Pandas, NumPy, SciPy
- Data Visualization, Matplotlib, Seaborn, Plotly
- Dashboard Visualization, Business Intelligence
- Storytelling

Chapter Summary

- Scenario
- Data Import
- Missingno
- Pd.crosstab
- Numeric data heatpmap
- Plot with hue for Target analysis

Scenario

Loans may be for a specific, one-time amount, or they may be available as an open-ended line of credit up to a specified limit. Loans come in many different forms including secured, unsecured, commercial, and personal loans.

There are several key factors that lenders consider:

- Income
- Credit Score
- Debt-to-income

Data import

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 import missingno as mso
6 import scipy
7 from scipy import stats
8 from scipy.stats import pearsonr
9 from scipy.stats import ttest_ind
```

```
1 train = pd.read_csv("loan_train.csv")
2 test = pd.read_csv("loan_test.csv")
```

The original dataset has been already split into train data and test data. Both of their columns are the same, just different in rows.

Overview all columns

The Status is the Target we need to analyse how the loan get approval by the particulars variables.

```
1 train.sample(5)
```

	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Coapplicant_Income	Loan_Amount	Term	Credit_History	Area	Status
425	Male	Yes	0	Graduate	No	266600	430000.0	12100000	360.0	1.0	Rural	Y
436	Male	No	0	Graduate	No	192600	185100.0	5000000	360.0	1.0	Semiurban	Y
479	Male	Yes	2	Graduate	No	294700	160300.0	0	360.0	1.0	Urban	N
434	Male	No	0	Graduate	No	375000	0.0	10000000	360.0	1.0	Urban	Y
300	Male	Yes	0	Not Graduate	No	180000	293400.0	9300000	360.0	0.0	Urban	N

Overview all columns

```
1 train.describe(include='all').style.background_gradient(cmap="tab10")
```

	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Coapplicant_Income	Loan_Amount	Term	Credit_History	A
count	601	611	599	614	582	614.000000	614.000000	614.000000	600.000000	564.000000	
unique	2	2	4	2	2	nan	nan	nan	nan	nan	
top	Male	Yes	0	Graduate	No	nan	nan	nan	nan	nan	Semiur
freq	489	398	345	480	500	nan	nan	nan	nan	nan	
mean	nan	nan	nan	nan	nan	540345.928339	162124.579803	14141042.345277	342.000000	0.842199	
std	nan	nan	nan	nan	nan	610904.167339	292624.836922	8815682.464395	65.120410	0.364878	
min	nan	nan	nan	nan	nan	15000.000000	0.000000	0.000000	12.000000	0.000000	
25%	nan	nan	nan	nan	nan	287750.000000	0.000000	9800000.000000	360.000000	1.000000	
50%	nan	nan	nan	nan	nan	381250.000000	118850.000000	12500000.000000	360.000000	1.000000	
75%	nan	nan	nan	nan	nan	579500.000000	229725.000000	16475000.000000	360.000000	1.000000	
max	nan	nan	nan	nan	nan	8100000.000000	4166700.000000	70000000.000000	480.000000	1.000000	

Overview all columns

```
1 print(train.shape, test.shape)
```

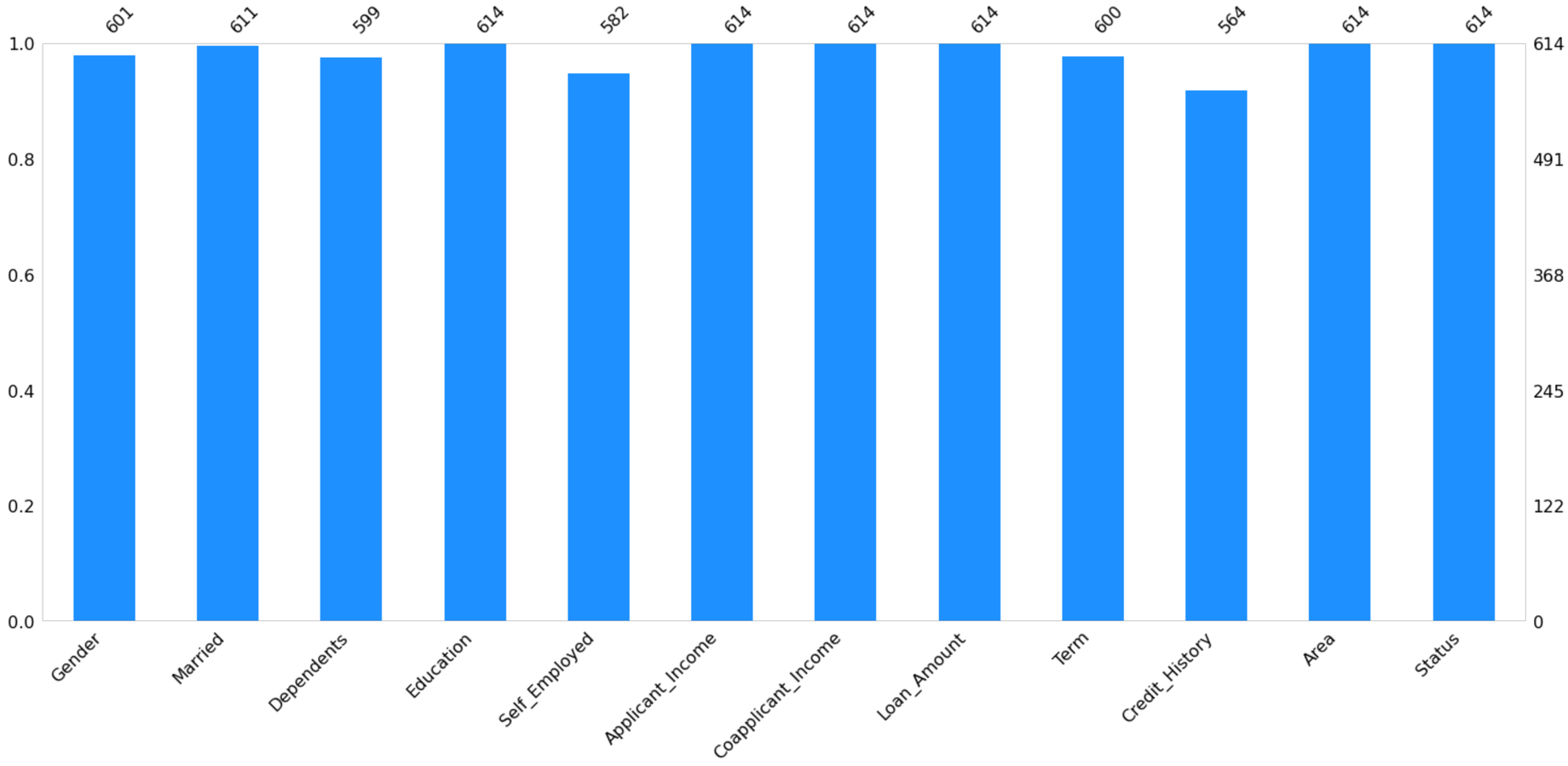
```
(614, 12) (367, 11)
```

```
1 test.describe(include='all').style.background_gradient(cmap="Pastel2")
```

	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Coapplicant_Income	Loan_Amount	Term	Credit_History	Area
count	356	367	357	367	344	367.000000	367.000000	367.000000	361.000000	338.000000	367
unique	2	2	4	2	2	nan	nan	nan	nan	nan	3
top	Male	Yes	0	Graduate	No	nan	nan	nan	nan	nan	Urban
freq	286	233	200	283	307	nan	nan	nan	nan	nan	140
mean	nan	nan	nan	nan	nan	480559.945504	156957.765668	13427792.915531	342.537396	0.825444	nan
std	nan	nan	nan	nan	nan	491068.539898	233423.209869	6296142.567929	65.156643	0.380150	nan
min	nan	nan	nan	nan	nan	0.000000	0.000000	0.000000	6.000000	0.000000	nan
25%	nan	nan	nan	nan	nan	286400.000000	0.000000	10000000.000000	360.000000	1.000000	nan
50%	nan	nan	nan	nan	nan	378600.000000	102500.000000	12500000.000000	360.000000	1.000000	nan
75%	nan	nan	nan	nan	nan	506000.000000	243050.000000	15750000.000000	360.000000	1.000000	nan
max	nan	nan	nan	nan	nan	7252900.000000	2400000.000000	55000000.000000	480.000000	1.000000	nan

Missing data check

```
1 msno.bar(train, color="dodgerblue")
```



Missing data check

```
1 train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Gender             601 non-null    object 
 1   Married            611 non-null    object 
 2   Dependents         599 non-null    object 
 3   Education          614 non-null    object 
 4   Self_Employed      582 non-null    object 
 5   Applicant_Income   614 non-null    int64  
 6   Coapplicant_Income 614 non-null    float64
 7   Loan_Amount        614 non-null    int64  
 8   Term               600 non-null    float64
 9   Credit_History     564 non-null    float64
 10  Area               614 non-null    object 
 11  Status             614 non-null    object 
dtypes: float64(3), int64(2), object(7)
memory usage: 57.7+ KB
```

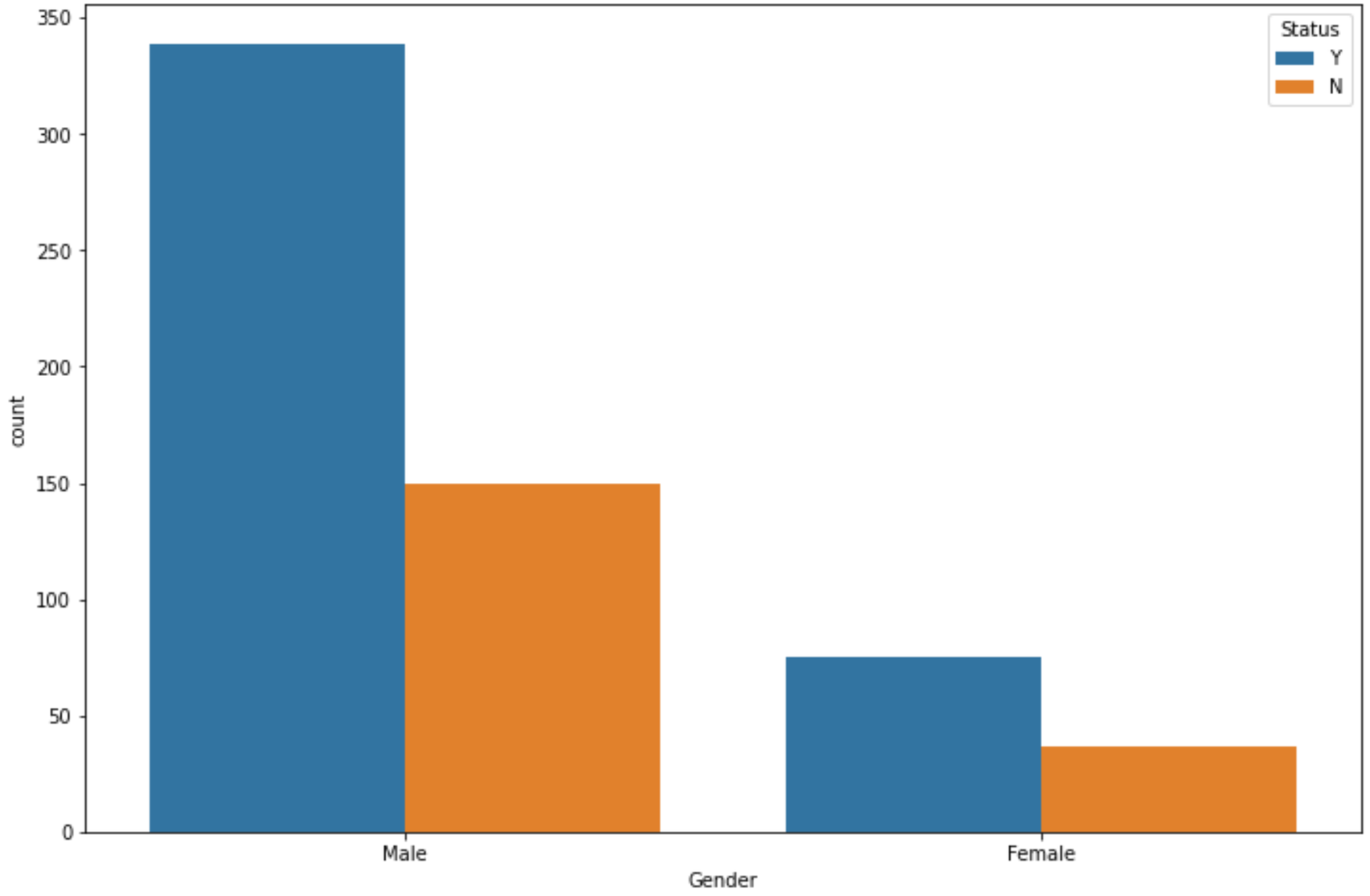
Gender vs Status

```
1 train.Gender.value_counts(dropna = False)
```

```
Male      489
Female    112
NaN        13
Name: Gender, dtype: int64
```

```
1 sns.countplot(x="Gender", data = train, hue='Status')
2 plt.show()
```

Gender vs Status

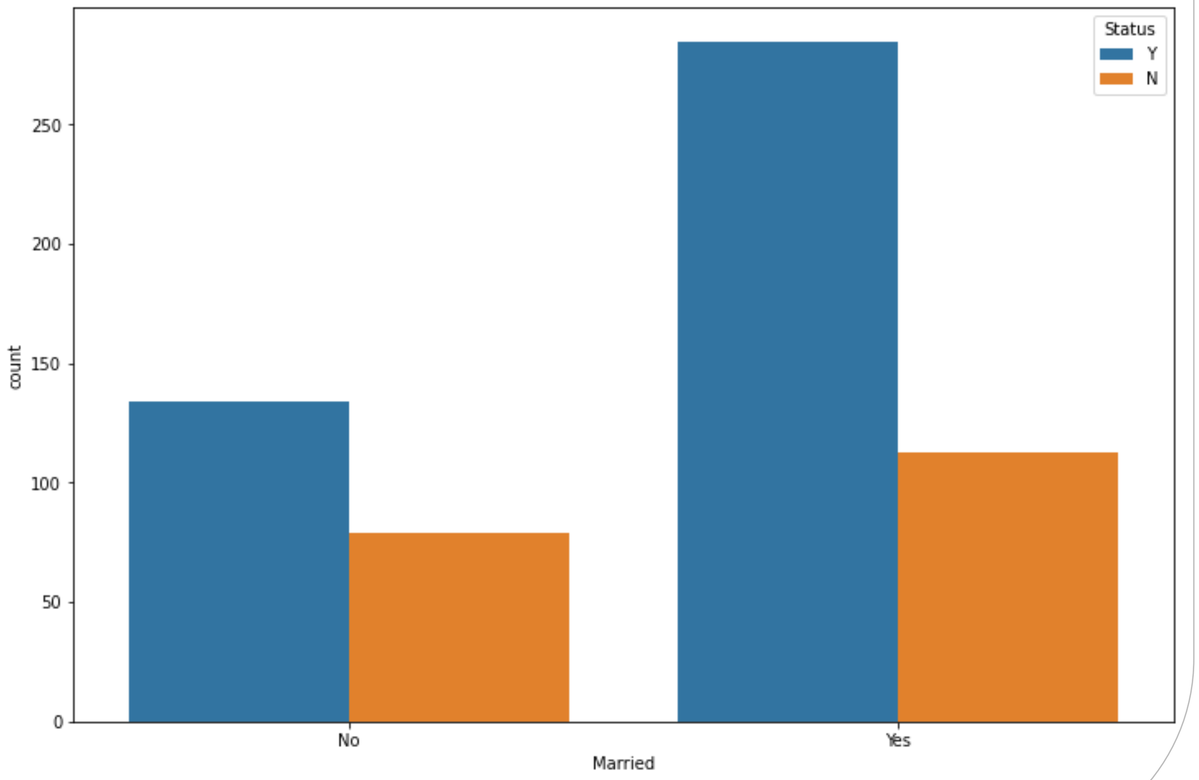


Married vs Status

```
1 train.Married.value_counts(dropna = False)
```

```
Yes    398
No     213
NaN      3
Name: Married, dtype: int64
```

```
1 sns.countplot(x = "Married", data = train, hue='Status')
2 plt.show()
```

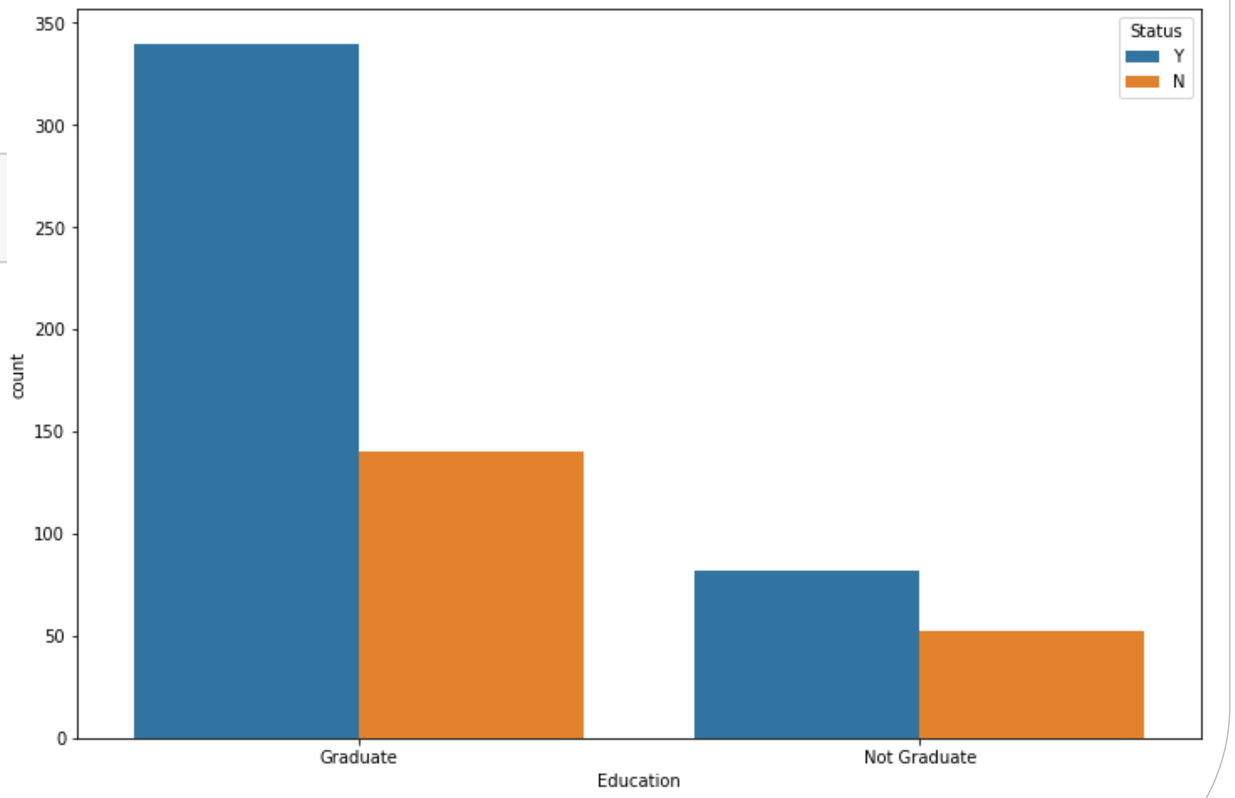


Education vs Status

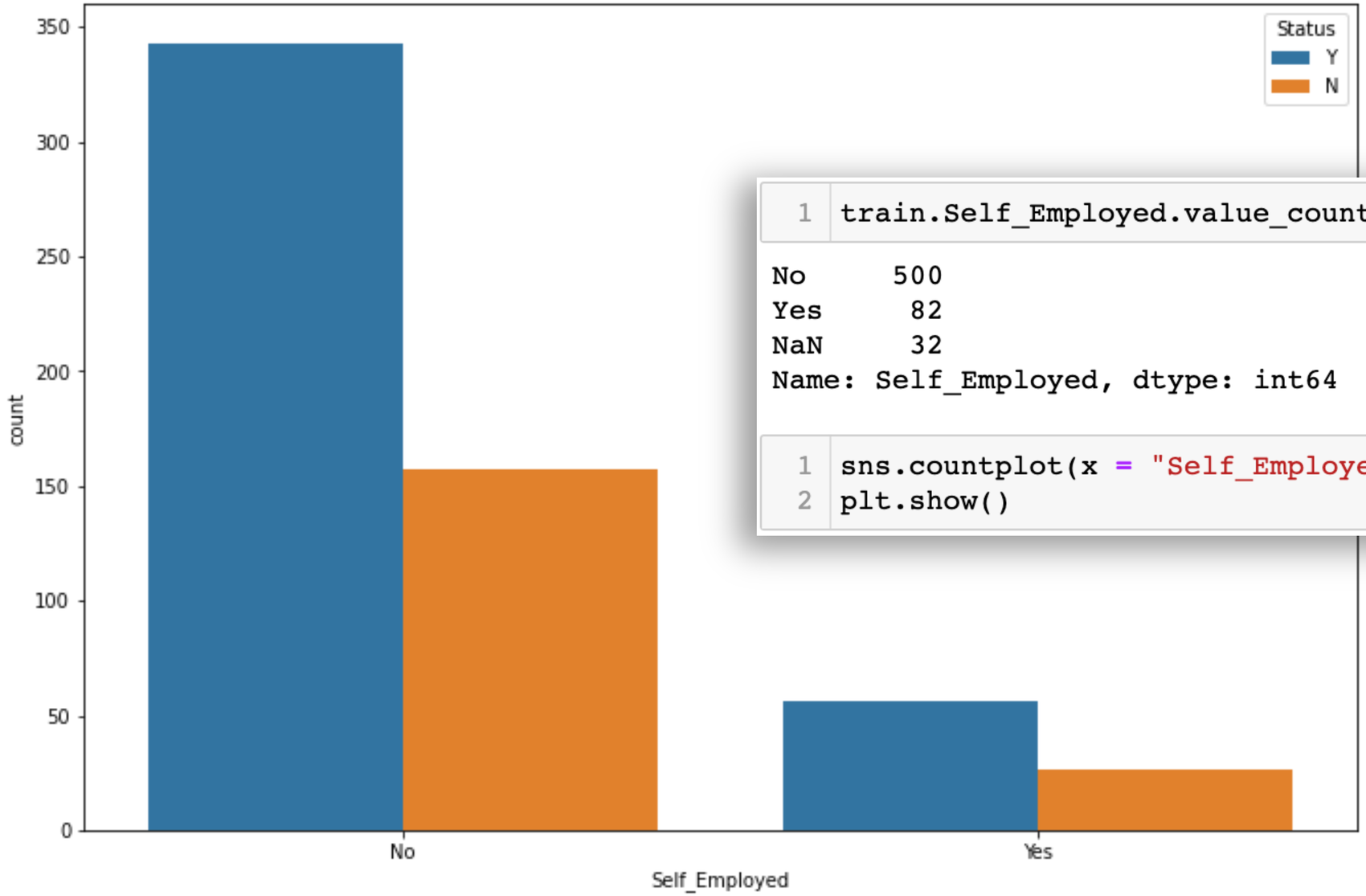
```
1 train.Education.value_counts(dropna = False)
```

```
Graduate        480
Not Graduate    134
Name: Education, dtype: int64
```

```
1 sns.countplot(x = "Education", data = train, hue='Status')
2 plt.show()
```



Self_employed vs Status




```
1  train.Self_Employed.value_counts(dropna=False)
```

```
No      500
Yes      82
NaN      32
Name: Self_Employed, dtype: int64
```

```
1  sns.countplot(x = "Self_Employed", data = train, hue='Status')
2  plt.show()
```

Credit_history vs Status

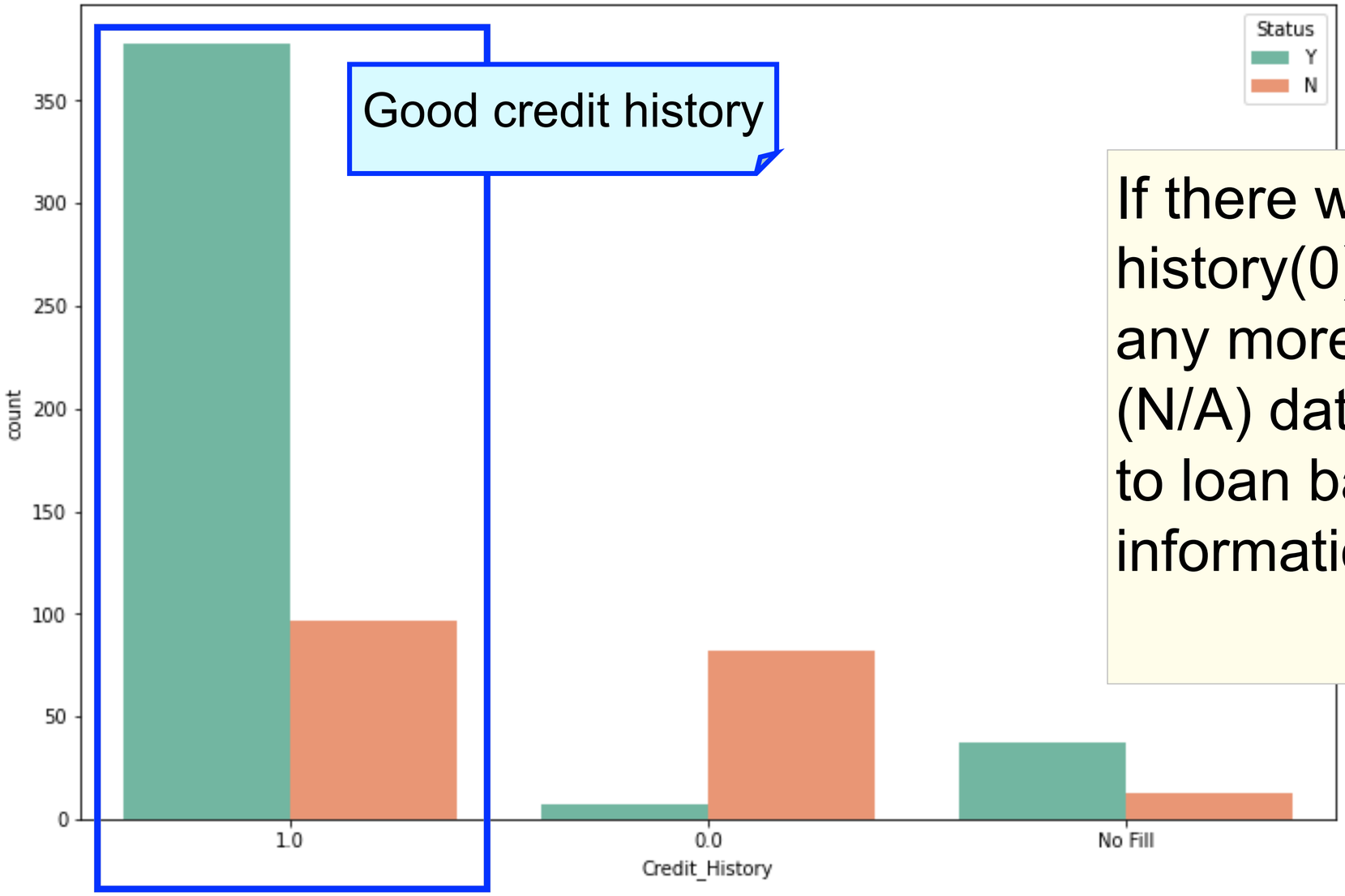
```
1 train.Credit_History.value_counts(dropna=False)
```

```
1.0    475
0.0     89
NaN     50
Name: Credit_History, dtype: int64
```

```
1 credit_h = train[['Credit_History','Status']]
2 credit_h = credit_h.fillna('No Fill')
3 sns.countplot(x = "Credit_History", data = credit_h,
4                hue = "Status", palette="Set2")
5 plt.show()
```

Credit_history 0 means there is a previous bad credit record.
Since there is great portion in NaN, we could not skip it. Instead we fill it with No Fill, and see how its Status vary to other.

Credit_history vs Status



If there was bad credit history(0), probably can’t loan any more. If there are unknow (N/A) data, it still have chance to loan based on other information.

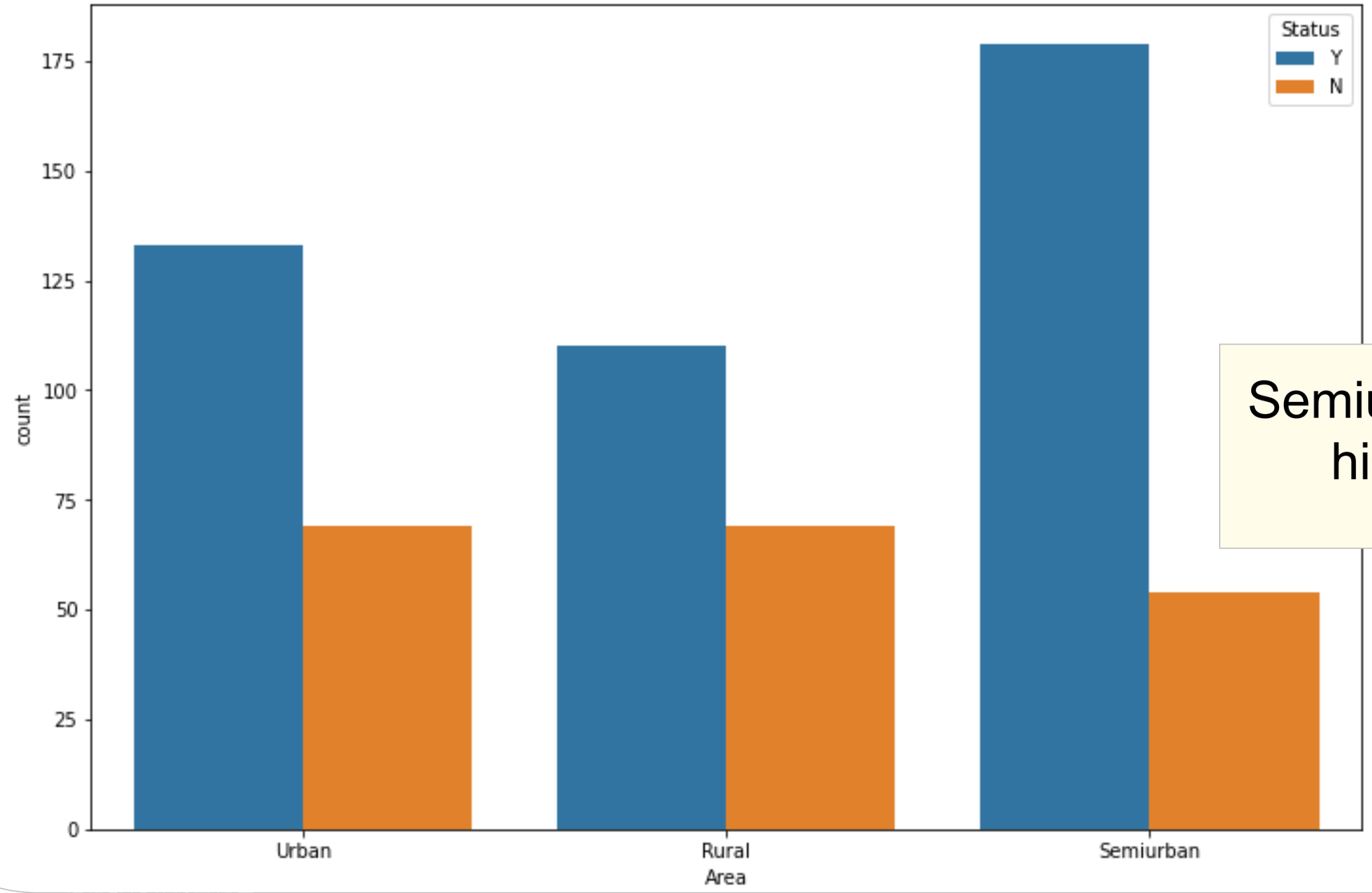
Property Area vs Status

```
1  train.Area.value_counts(dropna=False)
```

```
Semiurban     233
Urban         202
Rural         179
Name: Area, dtype: int64
```

```
1  sns.countplot(x = "Area", data = train,
2                hue="Status", palette="tab10")
3  plt.show()
```

Property Area vs Status



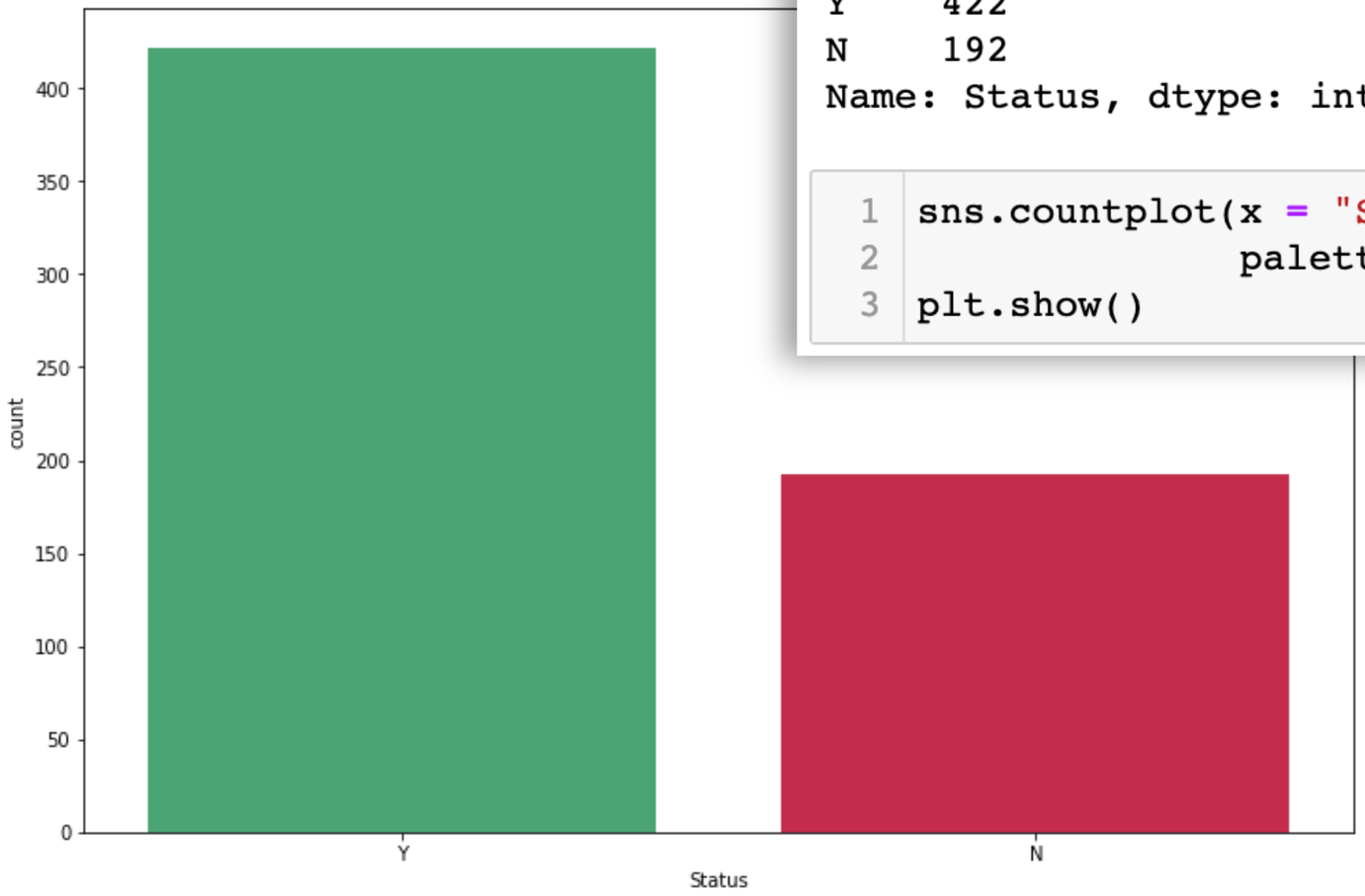
Semiurban area has quite high approval ratio

Status

```
1  train.Status.value_counts(dropna=False)
```

```
Y    422
N    192
Name: Status, dtype: int64
```

```
1  sns.countplot(x = "Status", data = train,
2                palette = {"Y": "#3CB372", "N": "#DC143C"})
3  plt.show()
```



Loan Amount Term

```
1 train.Term.value_counts(dropna=False)
```

```
360.0    512
180.0     44
480.0     15
NaN       14
300.0     13
240.0      4
84.0       4
120.0      3
60.0       2
36.0       2
12.0       1
Name: Term, dtype: int64
```

```
1 sns.countplot(x = "Term", data = train, palette = "Set1")
2 plt.show()
```

Income vs Loan amount

```
1 train[['Applicant_Income','Coapplicant_Income','Loan_Amount']].describe()\
2 .style.background_gradient(cmap="tab10")
```

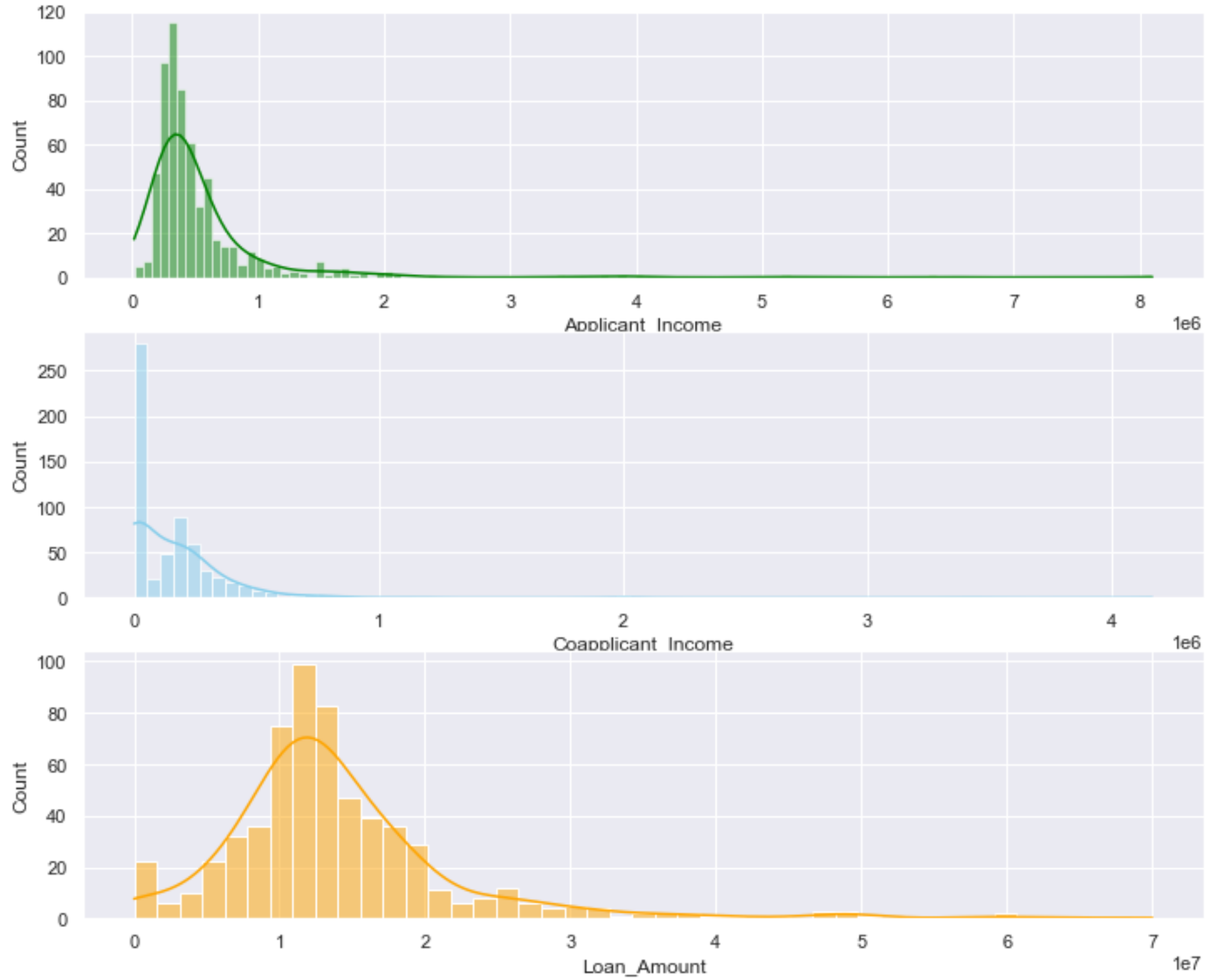
	Applicant_Income	Coapplicant_Income	Loan_Amount
count	614.000000	614.000000	614.000000
mean	540345.928339	162124.579803	14141042.345277
std	610904.167339	292624.836922	8815682.464395
min	15000.000000	0.000000	0.000000
25%	287750.000000	0.000000	9800000.000000
50%	381250.000000	118850.000000	12500000.000000
75%	579500.000000	229725.000000	16475000.000000
max	8100000.000000	4166700.000000	70000000.000000

Income vs Loan amount Distribution

Plot histogram to view the distribution vertically, and check if there is some correlation.

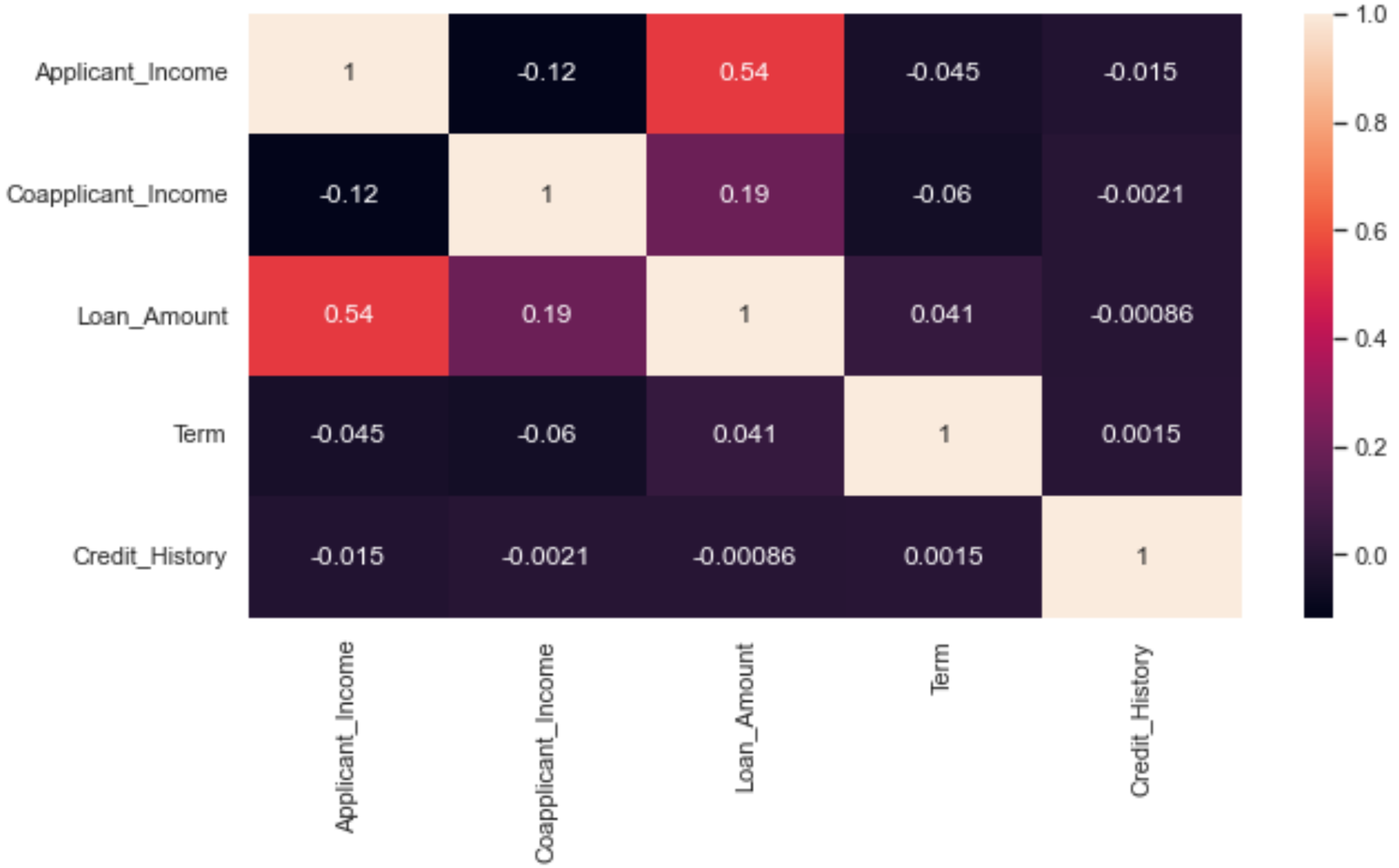
```
sns.set(style = "darkgrid")
fig, axs = plt.subplots(3, 1, figsize=(12, 10))
plt.subplot(311)
sns.histplot(data = train, x = "Applicant_Income", kde = True, color = 'green')
plt.subplot(312)
sns.histplot(data = train, x = "Coapplicant_Income", kde = True, color = 'skyblue')
plt.subplot(313)
sns.histplot(data = train, x = "Loan_Amount", kde = True, color = 'orange');
```

Income vs Loan amount Distribution



Heatmap

```
1 plt.figure(figsize = (10,5))
2 sns.heatmap(train.select_dtypes(include='number').corr(), annot = True)
```



Gender vs Married

```
1 pd.crosstab(train.Gender, train.Married).plot(kind = "bar", stacked = True)
2 plt.title('Gender vs Married')
3 plt.xlabel('Gender')
4 plt.ylabel('Frequency')
5 plt.xticks(rotation = 0)
6 plt.show()
```

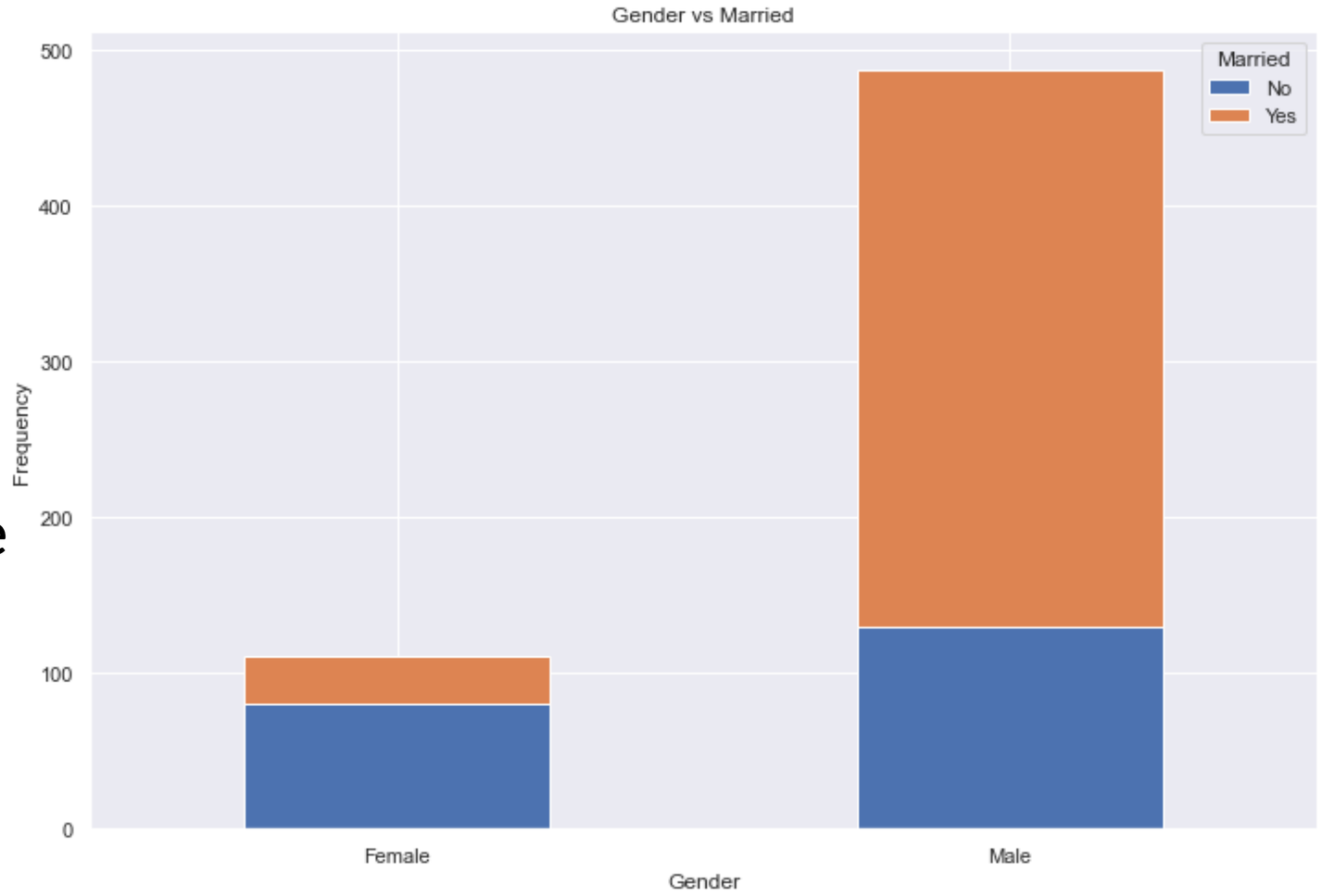
```
1 pd.crosstab(train.Gender, train.Married)
```

Married	No	Yes
Gender		
Female	80	31
Male	130	357

pd.crosstab() computes a frequency table of the factors unless an array of values and an aggregation function are passed.

Gender vs Married

Most male applicants are already married compared to female applicants. Also, the number of not married male applicants are higher compare to female applicants that had not married.



Self employed vs Credit history

```
1 pd.crosstab(train.Self_Employed, train.Credit_History)
```

Credit_History	0.0	1.0
Self_Employed		
No	76	387
Yes	12	63

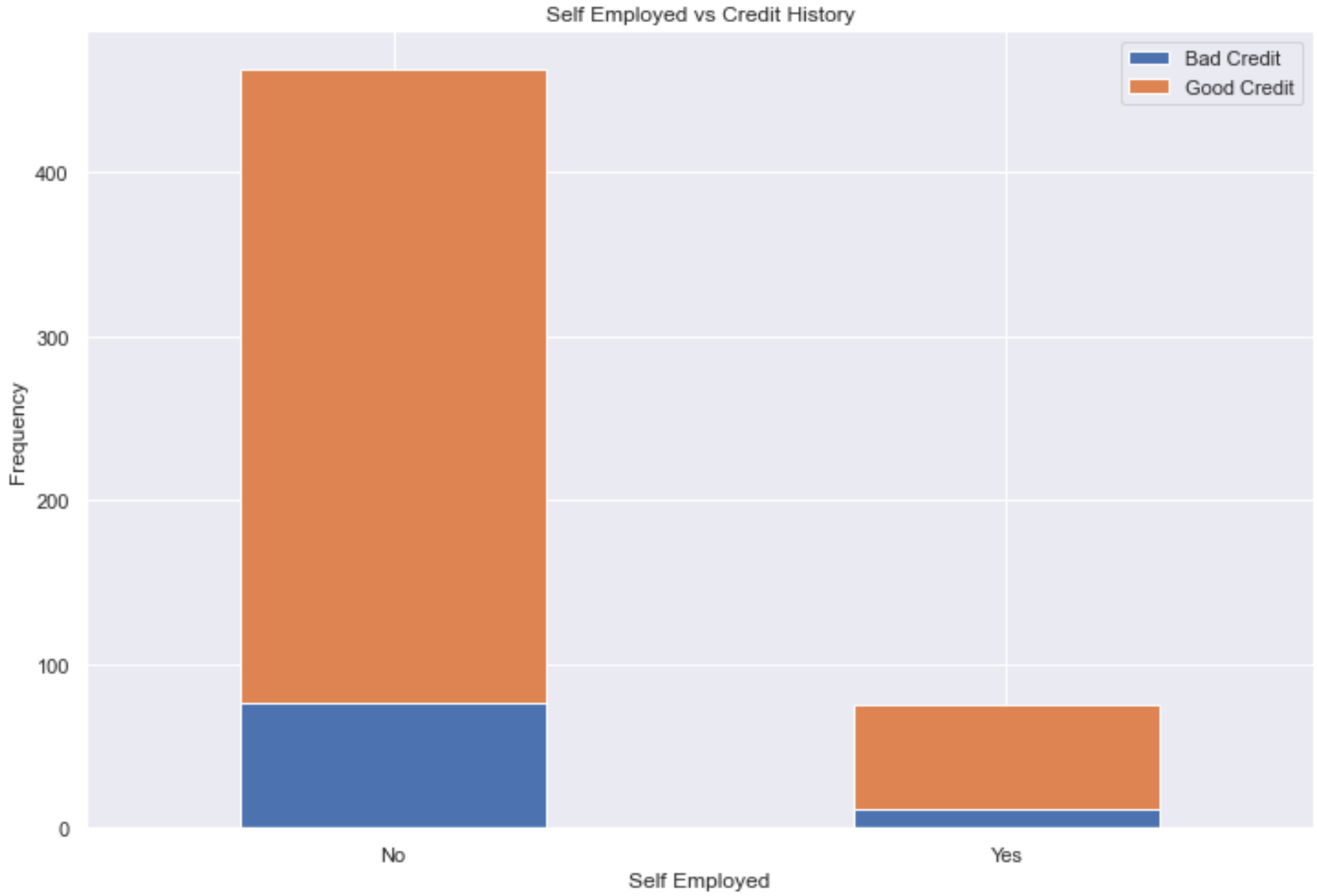
```
1 # show the ratio instead
2 pd.crosstab(train.Self_Employed, train.Credit_History, normalize='index')
```

Credit_History	0.0	1.0
Self_Employed		
No	0.164147	0.835853
Yes	0.160000	0.840000

Self employed vs Credit history

```
pd.crosstab(train.Self_Employed,
    train.Credit_History).plot(kind = "bar", stacked = True)
plt.title('Self Employed vs Credit History')
plt.xlabel('Self Employed')
plt.ylabel('Frequency')
plt.legend(["Bad Credit", "Good Credit"])
plt.xticks(rotation = 0)
plt.show()
```

Self employed vs Credit history



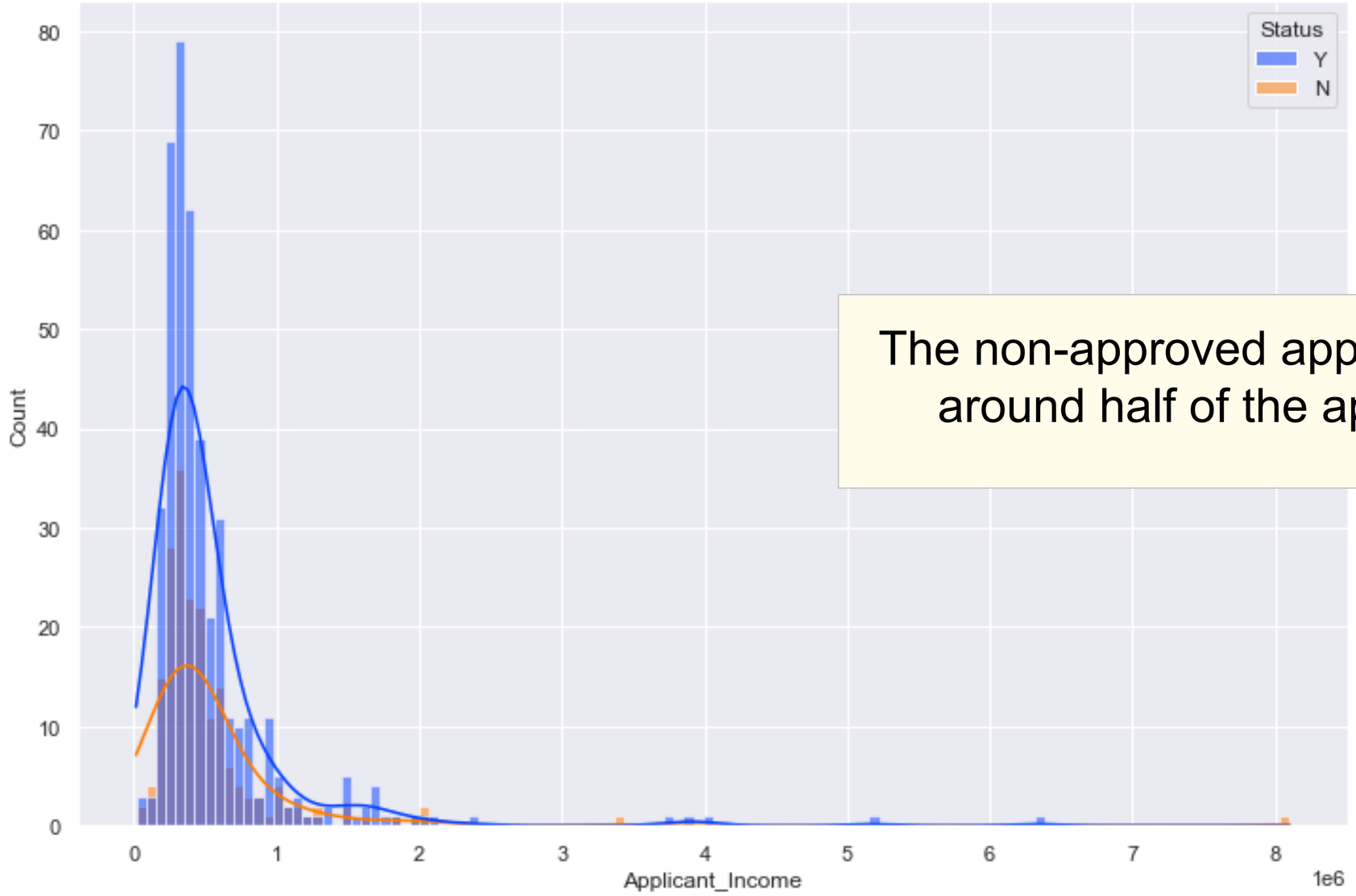
Applicant income vs Status

```
1  train['Applicant_Income'].describe()
```

```
count     6.140000e+02
mean      5.403459e+05
std       6.109042e+05
min       1.500000e+04
25%       2.877500e+05
50%       3.812500e+05
75%       5.795000e+05
max       8.100000e+06
Name: Applicant_Income, dtype: float64
```

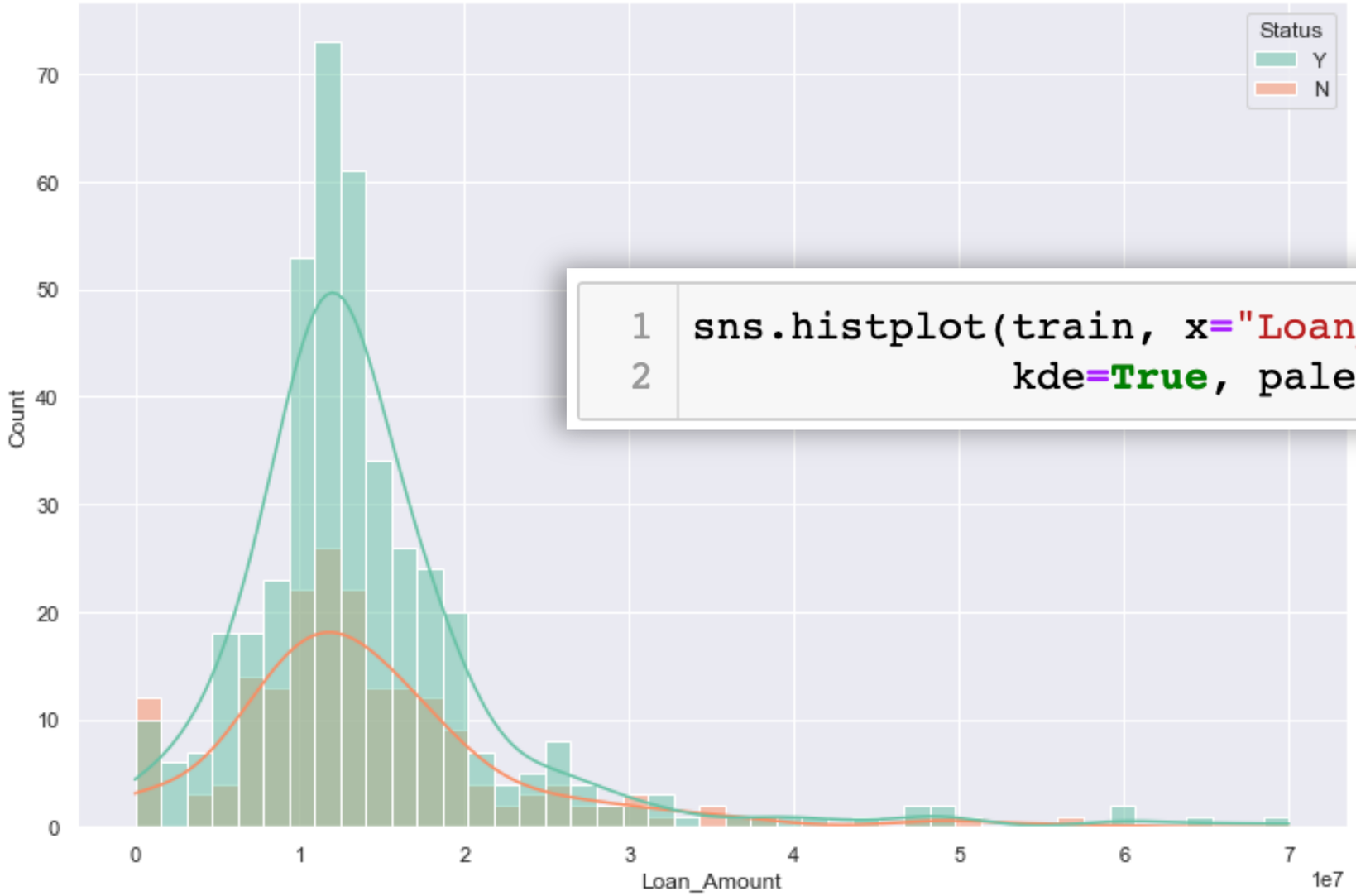
```
1  sns.histplot(train, x="Applicant_Income", hue="Status",
2               kde=True, palette="bright")
```

Applicant income vs Status



The non-approved applicant income is around half of the approved one.

Loan amount vs Status




```
sns.histplot(train, x="Loan_Amount", hue="Status",
             kde=True, palette="Set2")
```

Chapter Wrap Up

We had visualise the most of the columns and its relation to the Status.

To process the modelling of credit loan approval, we still need to clean the data, and decide what machine learning tool we shall use. You may refer to the “Heart Disease” diagnose model in Demo 11.

Reference & Resources

Official Website:

https://plotly.com/python/

Plotly Graph Objects:

https://plotly.com/python/graph-objects/

Seaborn:

https://seaborn.pydata.org/examples/index.html

Matplotlib:

https://matplotlib.org/