In [2]:
```
import numpy as np
import pandas as pd
import datetime as dt
import matplotlib.pyplot as plt

import seaborn as sns
sns.set()

%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
loan_df = pd.read_csv('Data/Loan_Data.csv', date_parser=True)
```

In [4]:
```
loan_df.head()
```

Out[4]:

	customer_id	credit_lines_outstanding	loan_amt_outstanding	total_debt_outstanding	income	years_employed	fico_score
0	8153374	0	5221.545193	3915.471226	78039.38546	5	605
1	7442532	5	1958.928726	8228.752520	26648.43525	2	572
2	2256073	0	3363.009259	2027.830850	65866.71246	4	602
3	4885975	0	4766.648001	2501.730397	74356.88347	5	612
4	4700614	1	1345.827718	1768.826187	23448.32631	6	631

In [55]:
```
len(loan_df)
```

Out[55]:
```
10000
```

In [5]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

In [203…
```
# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(loan_df.drop(['customer_id','default'], axis=1), loan_df['defau
model = LogisticRegression()

model.fit(x_train, y_train)

# Evaluate the model
y_pred = model.predict_proba(x_test)[:, 1]

for i in range(len(y_pred)):
    y_pred[i] = 1 if y_pred[i] > 0.95 else 0

accuracy = len(y_pred[y_pred == y_test])/len(y_test)
print('Accuracy:', accuracy)
```

```
Accuracy: 0.9815
```

In [213…

```
def cal_expected_loss(new_loan_amount, outstanding_loan_amount, probability_of_default, recovery_rate=0.1):

    likelihood = True if probability_of_default > 0.75 else False

    if likelihood == True:
        print("Likely to default!")
    else:
        print("Unlikely to default!")

    loss_given_default = new_loan_amount*(1-recovery_rate) + outstanding_loan_amount

    expected_loss = loss_given_default * probability_of_default

    return expected_loss
```

In [191…

```
loan_df.columns
```

Out[191]:

```
Index(['customer_id', 'credit_lines_outstanding', 'loan_amt_outstanding',
       'total_debt_outstanding', 'income', 'years_employed', 'fico_score',
       'default'],
      dtype='object')
```

In [217…

```
x_test = pd.DataFrame([[0, 5000000 ,3000, 60000, 3, 510]])

y_pred = model.predict_proba(x_test)[:, 1]
```

```
expected_loss = cal_expected_loss(5000000, 3000, y_pred[0])

print('Expected loss:', expected_loss)
```

```
Likely to default!
Expected loss: 4503000.0
```