The data set consists of 2000 samples from each of two categories. Five variables are

- 1. Income
- 2. Age
- 3. Loan
- 4. Loan to Income (engineered feature)
- 5. Default

import pandas as pd

default = pd.read_csv('https://github.com/ybifoundation/Dataset/raw/main/Credit%20Default.csv
default.head()

t	Defaul ⁻	Loan to Income	Loan	Age	Income	
0	(0.122537	8106.532131	59.017015	66155.92510	0
0	(0.190752	6564.745018	48.117153	34415.15397	1
0	(0.139940	8020.953296	63.108049	57317.17006	2
0	(0.142911	6103.642260	45.751972	42709.53420	3
1		0.130990	8770.099235	18.584336	66952.68885	4

default.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 5 columns):

#	Column	Non-Null Count	Dtype
0	Income	2000 non-null	float64
1	Age	2000 non-null	float64
2	Loan	2000 non-null	float64
3	Loan to Income	2000 non-null	float64
4	Default	2000 non-null	int64

dtypes: float64(4), int64(1)

memory usage: 78.2 KB

default.describe()

		Income	Age	Loan	Loan to Income	Default	7
	count	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	
	mean	45331.600018	40.927143	4444.369695	0.098403	0.141500	
	std	14326.327119	13.262450	3045.410024	0.057620	0.348624	
	min	20014.489470	18.055189	1.377630	0.000049	0.000000	
	25%	32796.459720	29.062492	1939.708847	0.047903	0.000000	
	50%	45789.117310	41.382673	3974.719418	0.099437	0.000000	
defau	lt['Def	ault'].value_	counts()				
0 1717 1 283 Name: Default, dtype: int64							
defau	lt.colu	ımns					
	Index(['Income', 'Ag	e', 'Loan', '	Loan to Incom	e', 'Default'],	dtype='object	')
defau	lt.colu	ımns					
<pre>Index(['Income', 'Age', 'Loan', 'Loan to Income', 'Default'], dtype='object')</pre>							
<pre>y = default['Default']</pre>							
X = d	efault.	drop(['Defaul	t'],axis=1)				
<pre>from sklearn.model_selection import train_test_split X_train, X_test, y_train, y_test = train_test_split(X,y, train_size=0.7, random_state=2529)</pre>							
<pre>X_train.shape, X_test.shape, y_train.shape, y_test.shape</pre>							
((1400, 4), (600, 4), (1400,), (600,))							
<pre>from sklearn.linear_model import LogisticRegression model = LogisticRegression()</pre>							
<pre>model.fit(X_train,y_train)</pre>							
▼ LogisticRegression							
LogisticRegression()							



```
model.intercept
   array([9.39569095])
model.coef
   array([[-2.31410016e-04, -3.43062682e-01, 1.67863323e-03,
          1.51188530e+00]])
y pred = model.predict(X test)
y_pred
   0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
         1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
         0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
         1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1,
         0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
         0, 0, 0, 0, 0, 01)
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
confusion_matrix(y_test,y_pred)
   array([[506, 13],
         [ 17, 64]])
accuracy_score(y_test,y_pred)
```

0.95

print(classification_report(y_test,y_pred))

₽		precision	recall	f1-score	support
	0	0.97	0.97	0.97	519
	1	0.83	0.79	0.81	81
	accuracy			0.95	600
	macro avg	0.90	0.88	0.89	600
	weighted avg	0.95	0.95	0.95	600

Colab paid products - Cancel contracts here

✓ 0s completed at 6:33 PM

×