

# Projeto de Regressão Logística

August 28, 2025

## 1 Projeto de Regressão Logística

Neste projeto estaremos trabalhando com um conjunto de dados falso de publicidade, indicando se um usuário de internet específico clicou ou não em uma propaganda. Vamos tentar criar um modelo que preveja se clicará ou não em um anúncio baseado nos recursos desse usuário.

Este conjunto de dados contém os seguintes recursos:

- ‘Daily Time Spent on Site’: tempo no site em minutos.
- ‘Age’: idade do consumidor.
- ‘Area Income’: Média da renda do consumidor na região.
- ‘Daily Internet Usage’: Média em minutos por di que o consumidor está na internet.
- ‘Linha do tópico do anúncio’: Título do anúncio.
- ‘City’: Cidade do consumidor.
- ‘Male’: Se o consumidor era ou não masculino.
- ‘Country’: País do consumidor.
- ‘Timestamp’: hora em que o consumidor clicou no anúncio ou janela fechada.
- ‘Clicked on Ad’: 0 ou 1 indicam se clicou ou não no anúncio.

### 1.1 Importar bibliotecas

Importe algumas bibliotecas que você acha que você precisará

```
[10]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
```

### 1.2 Obter dados

Leia o arquivo advertising.csv e grave-o em um DataFrame chamado data.

```
[2]: data = pd.read_csv('advertising.csv')
```

Verifique o cabeçalho do ad\_data

```
[3]: data.head()
```

```
[3]:      Daily Time Spent on Site  Age  Area Income  Daily Internet Usage  \
0          68.95    35      61833.90          256.09
1          80.23    31      68441.85          193.77
2          69.47    26      59785.94          236.50
3          74.15    29      54806.18          245.89
4          68.37    35      73889.99          225.58

      Ad Topic Line      City  Male  Country  \
0  Cloned 5thgeneration orchestration  Wrightburgh  0  Tunisia
1  Monitored national standardization  West Jodi  1  Nauru
2  Organic bottom-line service-desk  Davidton  0  San Marino
3  Triple-buffered reciprocal time-frame  West Terrifurt  1  Italy
4  Robust logistical utilization  South Manuel  0  Iceland

      Timestamp  Clicked on Ad
0  2016-03-27 00:53:11      0
1  2016-04-04 01:39:02      0
2  2016-03-13 20:35:42      0
3  2016-01-10 02:31:19      0
4  2016-06-03 03:36:18      0
```

**\*\* Use info() e describe() em ad\_data \*\***

```
[4]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
#   Column              Non-Null Count  Dtype
---  -
0   Daily Time Spent on Site  1000 non-null  float64
1   Age                   1000 non-null  int64
2   Area Income           1000 non-null  float64
3   Daily Internet Usage   1000 non-null  float64
4   Ad Topic Line          1000 non-null  object
5   City                   1000 non-null  object
6   Male                   1000 non-null  int64
7   Country                1000 non-null  object
8   Timestamp              1000 non-null  object
9   Clicked on Ad          1000 non-null  int64
dtypes: float64(3), int64(3), object(4)
memory usage: 78.3+ KB

[5]: data.describe()
```

```
[5]:      Daily Time Spent on Site      Age  Area Income  \
count      1000.000000  1000.000000  1000.000000
mean         65.000200   36.009000  55000.000080
```

std	15.853615	8.785562	13414.634022
min	32.600000	19.000000	13996.500000
25%	51.360000	29.000000	47031.802500
50%	68.215000	35.000000	57012.300000
75%	78.547500	42.000000	65470.635000
max	91.430000	61.000000	79484.800000

	Daily Internet Usage	Male	Clicked on Ad
count	1000.000000	1000.000000	1000.000000
mean	180.000100	0.481000	0.500000
std	43.902339	0.499889	0.500250
min	104.780000	0.000000	0.000000
25%	138.830000	0.000000	0.000000
50%	183.130000	0.000000	0.500000
75%	218.792500	1.000000	1.000000
max	269.960000	1.000000	1.000000

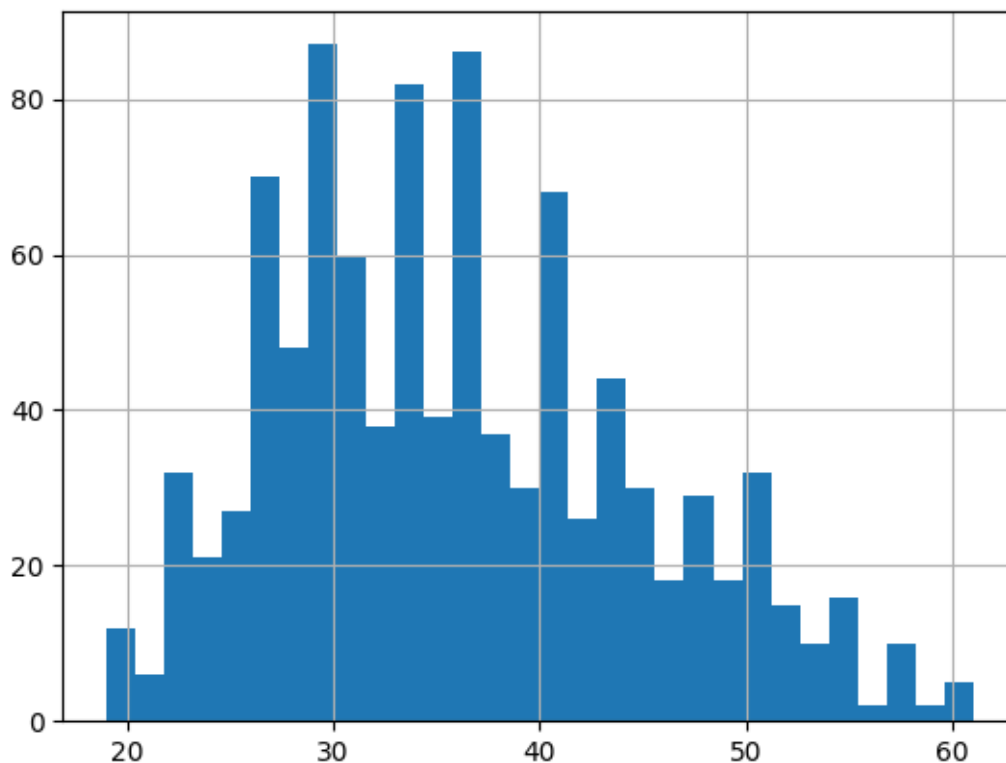
### 1.3 Análise de dados exploratória

Vamos usar Seaborn para explorar os dados!

Tente recriar os gráficos abaixo.

**Crie um histograma de “Age”**

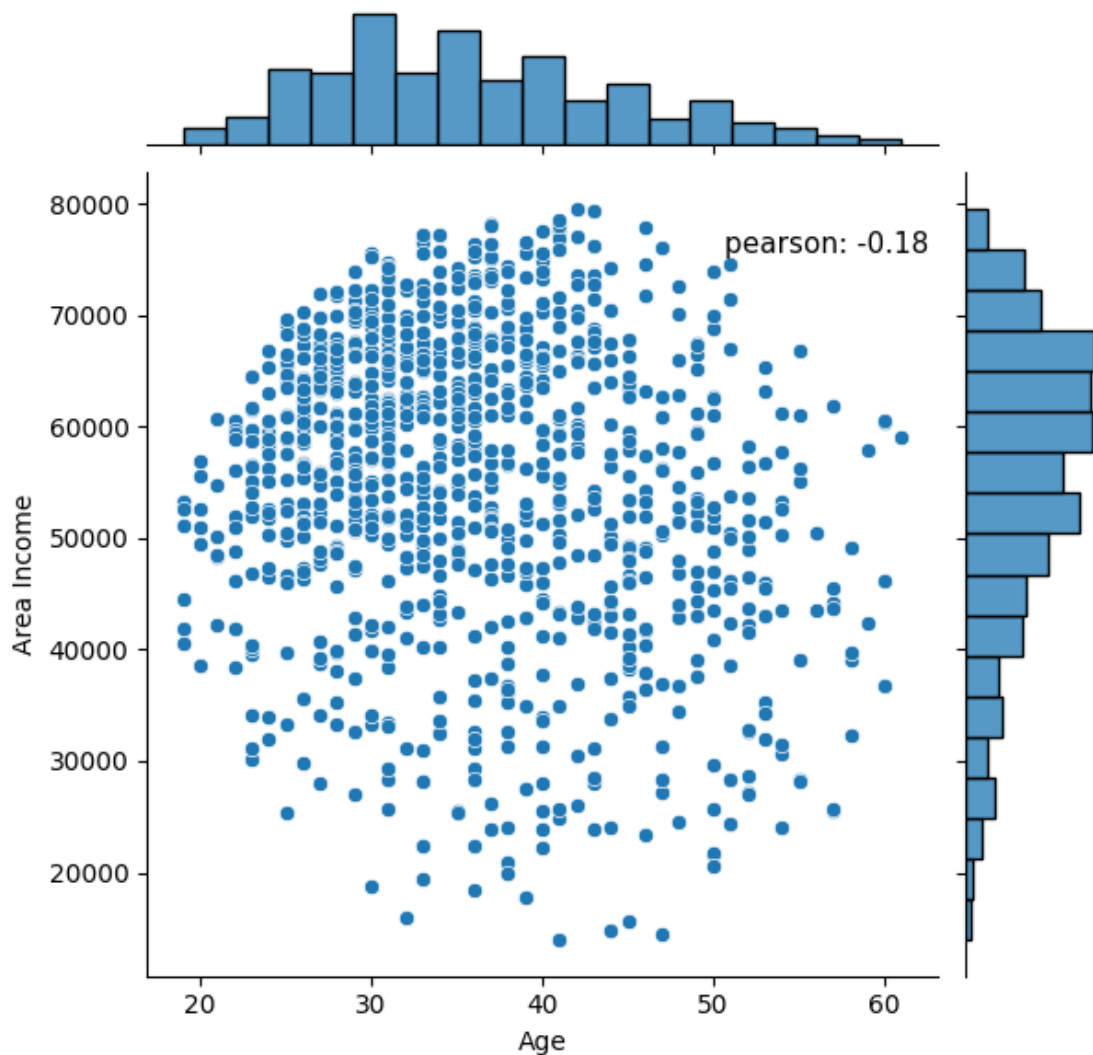
```
[7]: histogram=data['Age'].hist(bins=30)
```



Crie um jointplot mostrando “Area Income” versus “Age”

```
[17]: jointplot1 = sns.jointplot(x='Age', y='Area Income', data=data)
age_income_corr = data[['Age', 'Area Income']].corr().loc['Age', 'Area Income']
plt.annotate(f'pearson: {age_income_corr:.2f}', xy=(0.86, 0.9), xycoords='axes_
↪fraction', ha='center', fontsize=11)
```

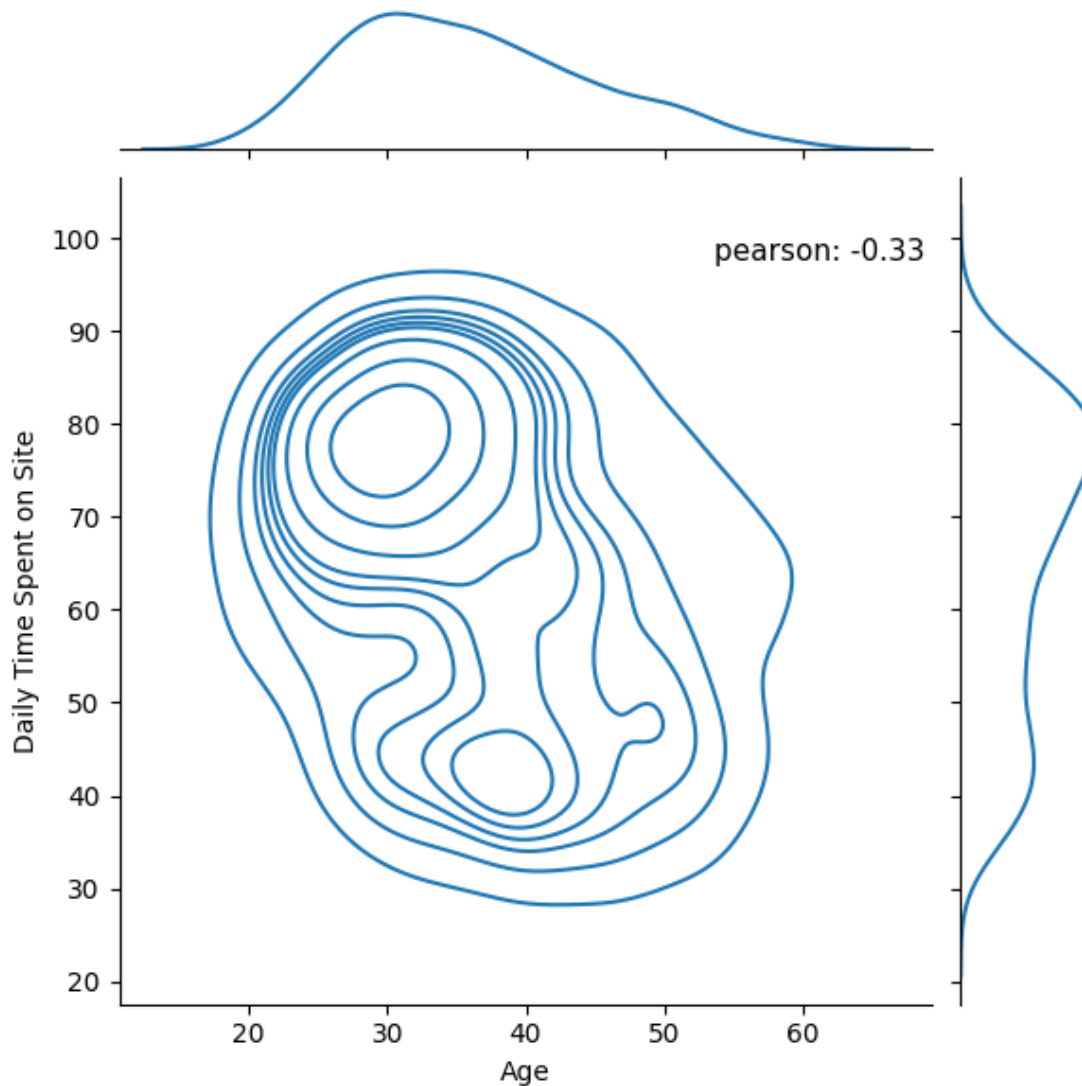
```
[17]: Text(0.86, 0.9, 'pearson: -0.18')
```



Crie um jointplot que mostre as distribuições KDE do “Daily Time spent” no site vs “Age”.

```
[18]: jointplot2=sns.jointplot(x='Age', y='Daily Time Spent on Site', data=data,
    ↪kind='kde')
daily_age_corr= data[['Age', 'Daily Time Spent on Site']].corr().loc['Age',
    ↪'Daily Time Spent on Site']
plt.annotate(f'pearson: {daily_age_corr:.2f}', xy=(0.86, 0.9), xycoords='axes_
    ↪fraction', ha='center', fontsize=11)
```

```
[18]: Text(0.86, 0.9, 'pearson: -0.33')
```

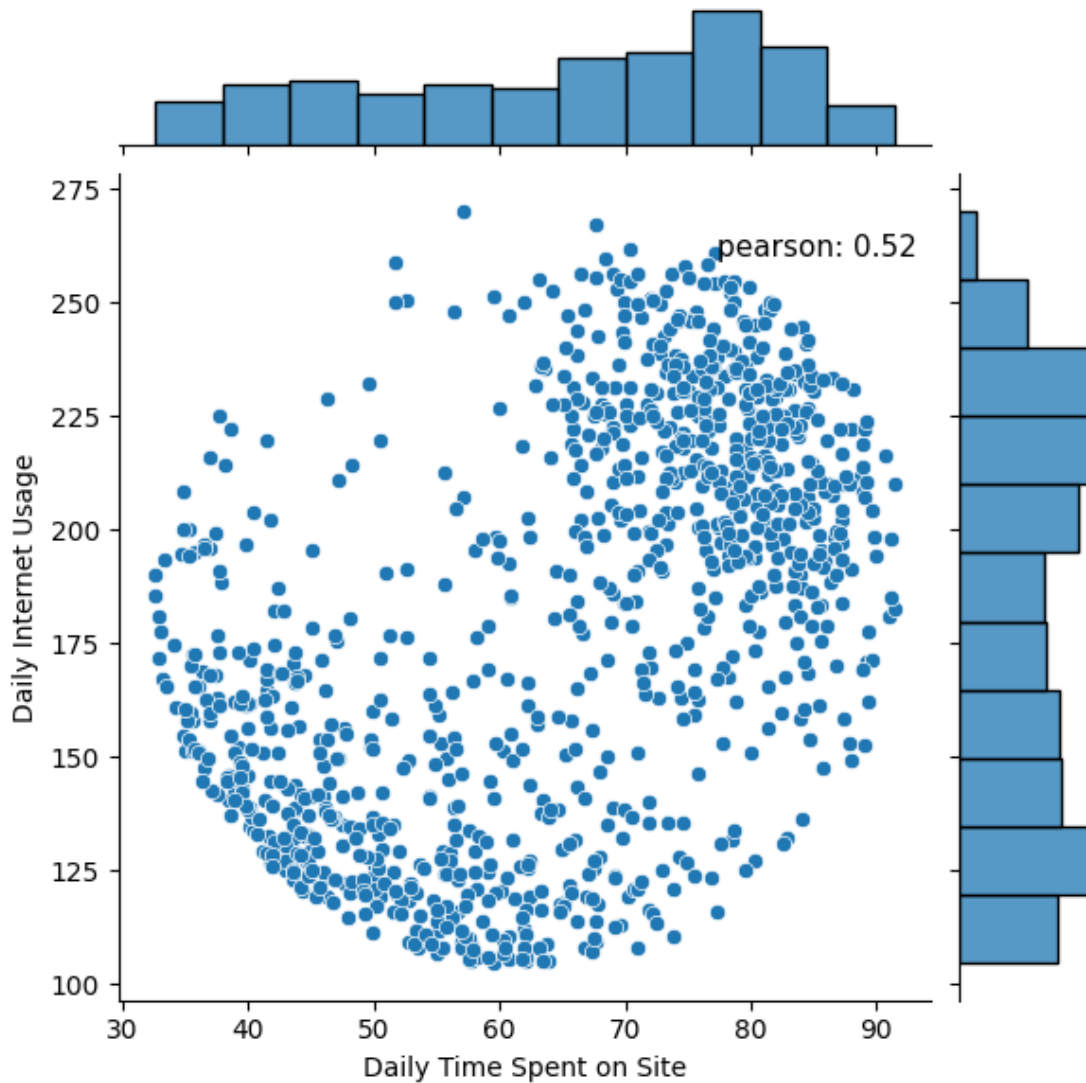


Crie um jointplot do ‘Daily Time Spent on Site’ vs. ‘Daily Internet Usage’

```
[20]: jointplot3 = sns.jointplot(x='Daily Time Spent on Site', y='Daily Internet
    ↪Usage', data=data,)
```

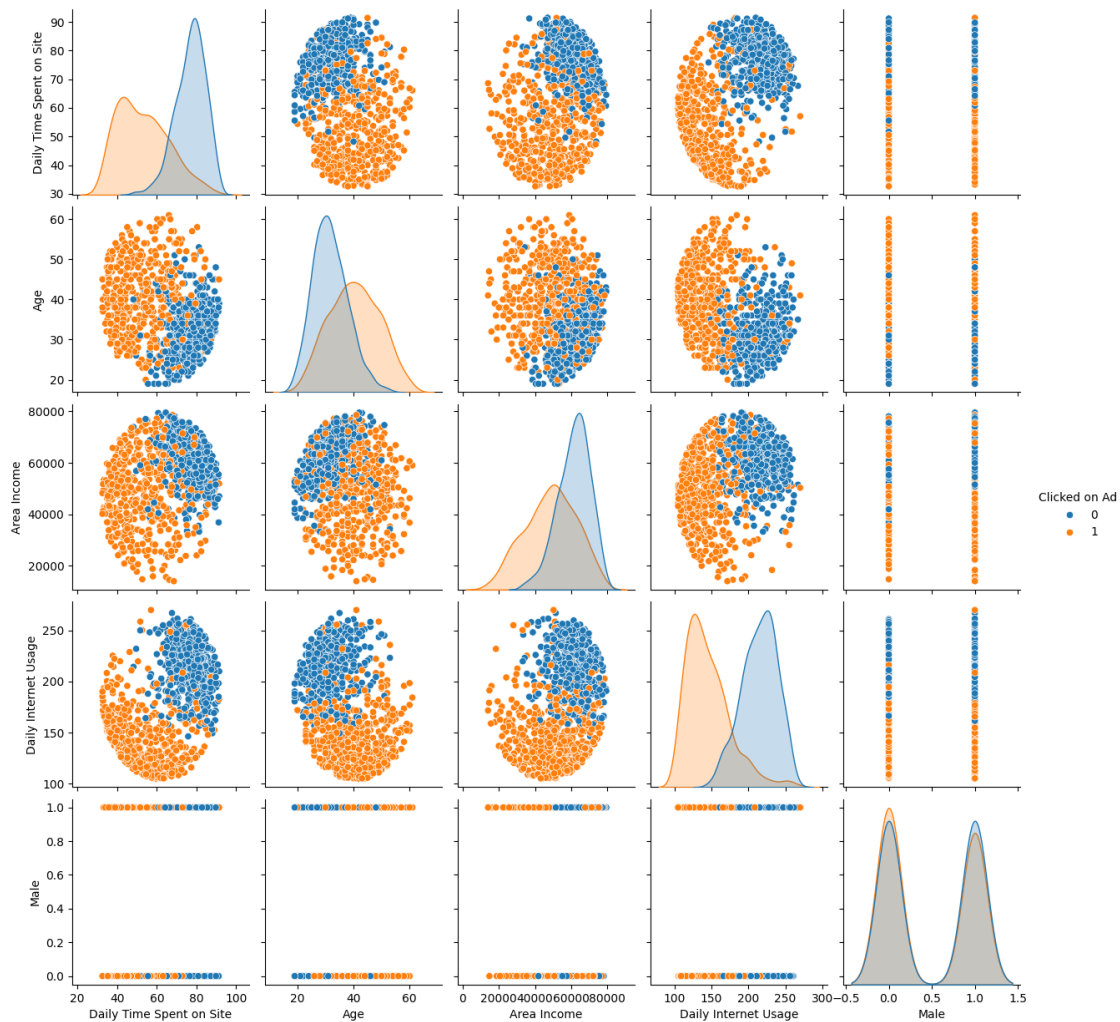
```
internet_age_corr = data[['Daily Time Spent on Site', 'Daily Internet Usage']].  
    .corr().loc['Daily Time Spent on Site', 'Daily Internet Usage']  
plt.annotate(f'pearson: {internet_age_corr:.2f}', xy=(0.86, 0.9),  
    xycoords='axes fraction', ha='center', fontsize=11)
```

[20]: Text(0.86, 0.9, 'pearson: 0.52')



Finalmente, crie um parplot com o matiz definido pelo recurso de coluna 'Clicked on Ad'.

[21]: parplot = sns.pairplot(data, hue='Clicked on Ad')



## 2 Regressão Logística

Agora é hora de quebrar nossos dados em treino e teste e fitar nosso modelo.

Você terá a liberdade aqui para escolher colunas em que deseja treinar!

Divida os dados em conjunto de treinamento e conjunto de testes usando `train_test_split`

```
[22]: x= data[['Daily Time Spent on Site', 'Daily Internet Usage', 'Age', 'Area
↪Income']]
y= data['Clicked on Ad']
```

Treine e ajuste um modelo de regressão logística no conjunto de treinamento.

```
[26]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
↳ random_state=42)
```

```
[28]: log = LogisticRegression(max_iter=1000)
log.fit(x_train, y_train)
print("Os coeficientes do modelo são:", log.coef_)
```

Os coeficientes do modelo são:  $\begin{bmatrix} -1.87506526e-01 & -7.15803891e-02 & 1.75900626e-01 \\ -1.41095975e-04 \end{bmatrix}$

## 2.1 Previsões e avaliações

Agora preveja valores para os dados de teste.

```
[29]: log.predict(x_test)
```

```
[29]: array([1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1,
1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1,
0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1,
1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0,
0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0,
1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1,
1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1,
0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1,
0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1,
1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0,
0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0])
```

Crie um relatório de classificação para o modelo.

```
[ ]: y_pred = log.predict(x_test)
```

```
[31]: print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.95	0.98	0.97	146
1	0.98	0.95	0.97	154
accuracy			0.97	300
macro avg	0.97	0.97	0.97	300
weighted avg	0.97	0.97	0.97	300