Proj2_resolvido

March 19, 2025

1 Projeto de Regressão Logística

Neste projeto iremos trabalhar com um DataSet falso de publicidade, que indica se um utilizador específico da internet clicou ou não em publicidade. Vamos tentar criar um modelo que preveja se clicará ou não num anúncio baseado nos recursos deste utilizador.

Este DataSet 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 dia, 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 necessárias **

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

1.2 Obter dados

** Leia o ficheiro advertising.csv e grave-o num DataFrame com o nome ad_data. **

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

** Verifique o cabeçalho do ad_data **

[3]:
```
ad_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]:
```
ad_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: 78.2+ KB
```

[5]:
```
ad_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     1000.00000
mean             180.000100     0.481000        0.50000
std               43.902339     0.499889        0.50025
min              104.780000     0.000000        0.00000
25%              138.830000     0.000000        0.00000
50%              183.130000     0.000000        0.50000
75%              218.792500     1.000000        1.00000
max              269.960000     1.000000        1.00000
```

1.3 Análise exploratória de dados

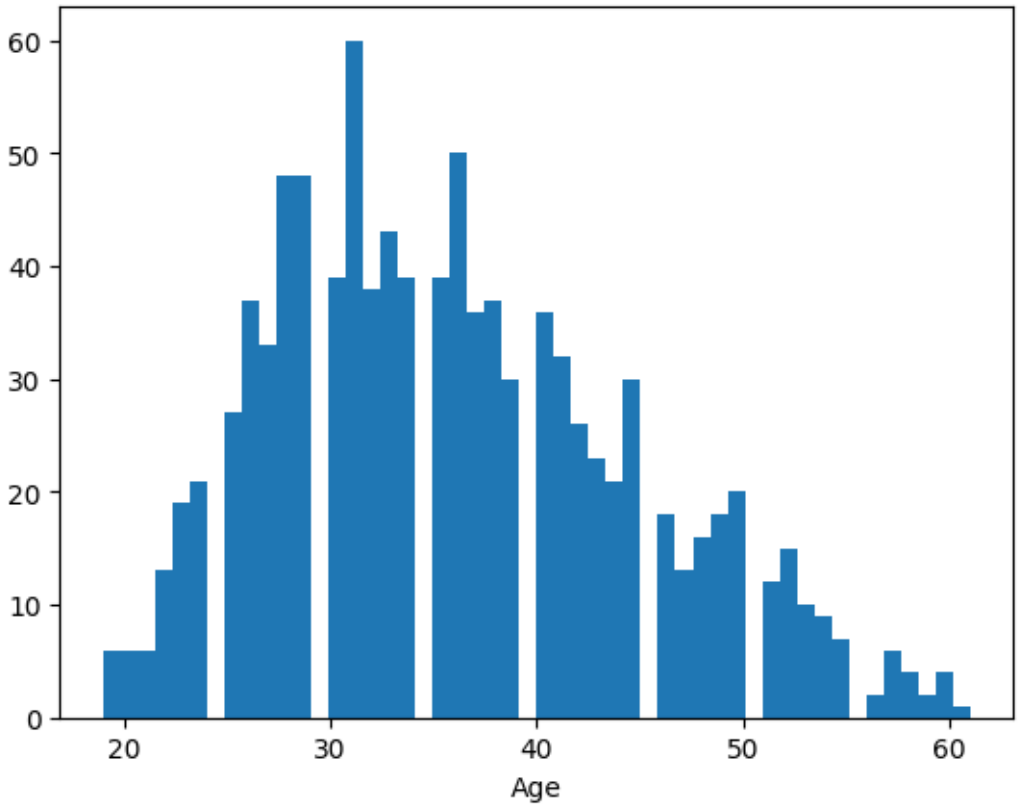
Vamos explorar os dados!

Tente recriar os gráficos abaixo.

** Crie um histograma de "Age", com os bins = 30 **

[6]:
```
# plt.figure(figsize=(12,6))
plt.hist(ad_data['Age'], bins=50)
plt.xlabel("Age")
```

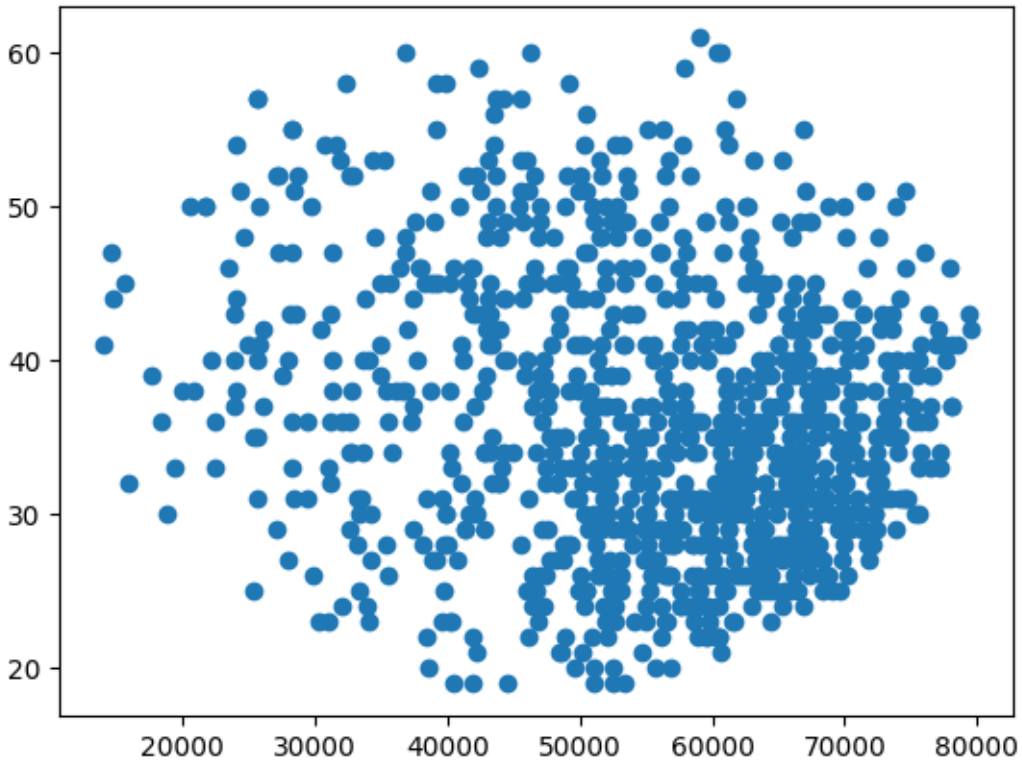
[6]:
```
Text(0.5, 0, 'Age')
```



** Crie um scatter plot que mostre a “Area Income” versus “Age” **

[7]:
```
plt.scatter(ad_data['Area Income'], ad_data['Age'])
```

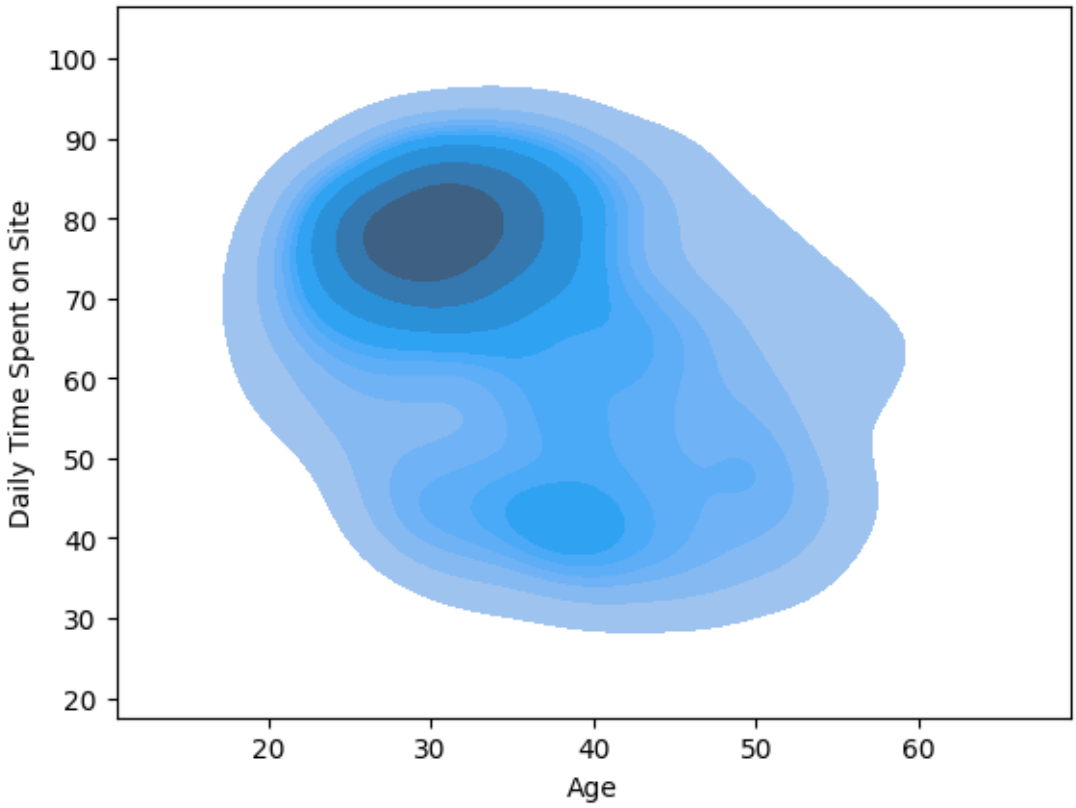
[7]:
```
<matplotlib.collections.PathCollection at 0x127f2e280>
```



** Crie um kdeplot que mostre as distribuições KDE do “Daily Time spent on Site” vs “Age”. **

[8]:
```
sns.kdeplot(x=ad_data['Age'], y=ad_data['Daily Time Spent on Site'], fill =␣
↪True)
```

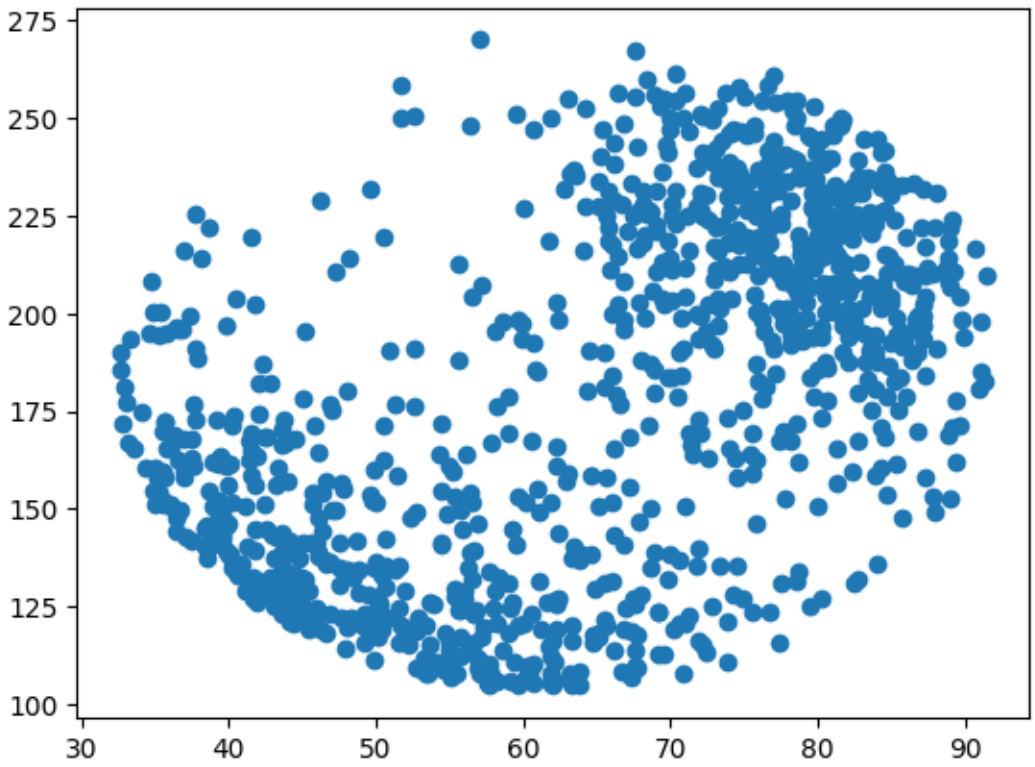
[8]:
```
<Axes: xlabel='Age', ylabel='Daily Time Spent on Site'>
```



** Crie um scatter plot do 'Daily Time Spent on Site' vs. 'Daily Internet Usage'**

[9]:
```
plt.scatter(ad_data['Daily Time Spent on Site'], ad_data['Daily Internet␣
↪Usage'])
```

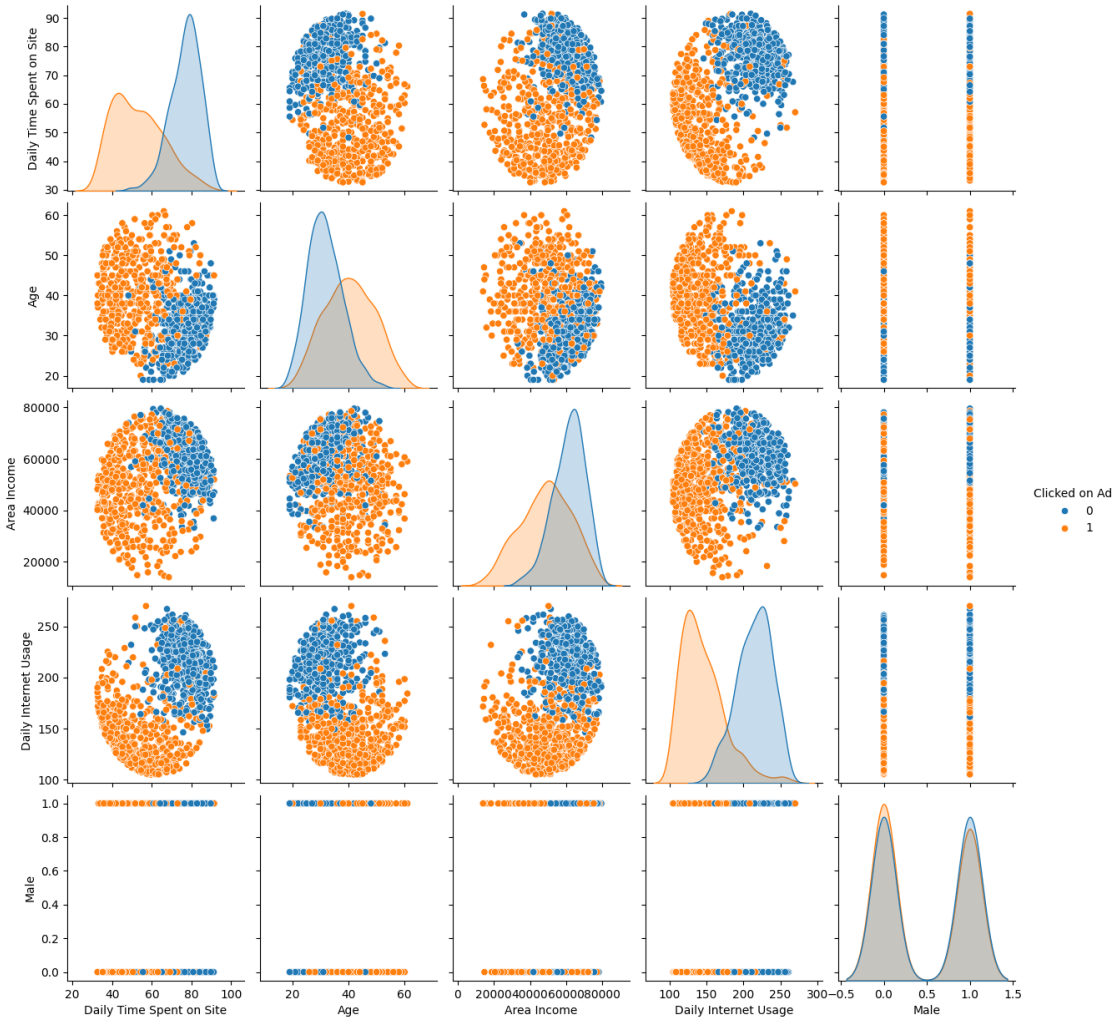
[9]:
```
<matplotlib.collections.PathCollection at 0x1324f4f40>
```



** Finalmente, crie um pairplot com o hue definido com a coluna 'Clicked on Ad'. **

[10]:
```
sns.pairplot(ad_data, hue = 'Clicked on Ad')
```

[10]:
```
<seaborn.axisgrid.PairGrid at 0x132527280>
```



2 Regressão Logística

Agora dividimos os dados em treino e teste.

Defina para o X as colunas 'Daily Time Spent on Site', 'Age', 'Area Income','Daily Internet Usage' e 'Male'.

Defina para o Y a coluna 'Clicked on Ad'

** Divida os dados em conjunto de treino e conjunto de testes usando train_test_split **

** O test_size = 0.3 e o random_state = 42 **

[11]:
```
from sklearn.model_selection import train_test_split
```

[12]:
```
X = ad_data[['Daily Time Spent on Site', 'Age', 'Area Income','Daily Internet␣
↪Usage', 'Male']]
```

```
Y = ad_data['Clicked on Ad']
```

[13]:
```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3,␣
↪random_state = 42)
```

** Treine e ajuste um modelo de regressão logística no conjunto de treino. **

[14]:
```
from sklearn.linear_model import LogisticRegression
```

[15]:
```
# logmodel = LogisticRegression()
logmodel = LogisticRegression(max_iter=1000)
#logmodel = LogisticRegression(random_state=0, multi_class='ovr', penalty='l2',␣
↪solver='liblinear', max_iter=1000)
```

[16]:
```
logmodel.fit(X_train,y_train)
```

[16]:
```
LogisticRegression(max_iter=1000)
```

2.1 Previsões e avaliações

** Agora preveja valores para os dados de teste. **

[17]:
```
predictions = logmodel.predict(X_test)
```

** Crie um relatório de classificação para o modelo. **

[18]:
```
from sklearn.metrics import classification_report
```

[19]:
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.96      0.97      0.97       146
           1       0.97      0.96      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
```

** Crie a Matriz de classificação para o modelo. **

[20]:
```
from sklearn.metrics import confusion_matrix
```

[21]:
```
print(confusion_matrix(y_test,predictions))
```

```
[[142   4]
 [  6 148]]
```

[22]:
```
mat = confusion_matrix(y_test,predictions)
tx_ok=mat[0][0]+mat[1][1]
tx_nok=mat[1][0]+mat[0][1]
print("Tx sucesso:", tx_ok/(tx_ok+tx_nok), ", Tx insucesso:", tx_nok/
 ↪(tx_ok+tx_nok))
```

```
Tx sucesso: 0.9666666666666667 , Tx insucesso: 0.03333333333333333
```