Logistic Regression Project

November 16, 2025

____ # Logistic Regression Project

In this project we will be working with a fake advertising data set, indicating whether or not a particular internet user clicked on an Advertisement. We will try to create a model that will predict whether or not they will click on an ad based off the features of that user.

This data set contains the following features:

- 'Daily Time Spent on Site': consumer time on site in minutes
- 'Age': cutomer age in years
- 'Area Income': Avg. Income of geographical area of consumer
- 'Daily Internet Usage': Avg. minutes a day consumer is on the internet
- 'Ad Topic Line': Headline of the advertisement
- 'City': City of consumer
- 'Male': Whether or not consumer was male
- 'Country': Country of consumer
- 'Timestamp': Time at which consumer clicked on Ad or closed window
- 'Clicked on Ad': 0 or 1 indicated clicking on Ad

0.1 Import Libraries

Import a few libraries you think you'll need (Or just import them as you go along!)

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

0.2 Get the Data

Read in the advertising.csv file and set it to a data frame called ad_data.

[3]:
```
ad_data = pd.read_csv('advertising.csv')
```

Check the head of ad_data

[4]:
```
ad_data.head()
```

[4]:
```
   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 and describe() on ad_data**

[5]:
```
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: 78.3+ KB
```

[6]:
```
ad_data.describe()
```

[6]:
```
       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
```

0.3 Exploratory Data Analysis

Let's use seaborn to explore the data!

Try recreating the plots shown below!

** Create a histogram of the Age**

[7]:
```
sns.histplot(data=ad_data['Age'], bins=40)
plt.grid(True)
plt.grid(linestyle='--', alpha=0.7)
```

Create a jointplot showing Area Income versus Age.

[8]:
```
g = sns.jointplot(data=ad_data, x='Age', y='Area Income')
```

Create a jointplot showing the kde distributions of Daily Time spent on site vs. Age.

[9]:
```
sns.jointplot(data=ad_data, x='Age', y='Daily Time Spent on Site', kind='kde',␣
↪fill='True',cmap='Reds')
```

[9]:
```
<seaborn.axisgrid.JointGrid at 0x1d2204bd1d0>
```

** Create a jointplot of 'Daily Time Spent on Site' vs. 'Daily Internet Usage'**

[10]:
```
sns.jointplot(data=ad_data, y='Daily Internet Usage', x='Daily Time Spent on␣
↪Site', kind='kde', fill='True',cmap='Reds')
```

[10]:
```
<seaborn.axisgrid.JointGrid at 0x1d2208e2710>
```

** Finally, create a pairplot with the hue defined by the 'Clicked on Ad' column feature.**

[11]:
```
sns.pairplot(ad_data, hue='Clicked on Ad', diag_kind='hist')
```

[11]:
```
<seaborn.axisgrid.PairGrid at 0x1d22045b0e0>
```

1 Logistic Regression

Now it's time to do a train test split, and train our model!

You'll have the freedom here to choose columns that you want to train on!

** Split the data into training set and testing set using train_test_split**

[12]:
```
from sklearn.model_selection import train_test_split
```

[13]:
```
X = ad_data[['Daily Time Spent on Site', 'Age', 'Area Income','Daily Internet␣
 ↪Usage', 'Male']]
y = ad_data['Clicked on Ad']
```

[14]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,␣
 ↪random_state=42)
```

** Train and fit a logistic regression model on the training set.**

[15]:
```
from sklearn.linear_model import LogisticRegression
```

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

```
C:\Users\cakaj\anaconda3\Lib\site-
packages\sklearn\linear_model\_logistic.py:465: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[16]:
```
LogisticRegression()
```

1.1 Predictions and Evaluations

** Now predict values for the testing data.**

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

** Create a classification report for the model.**

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

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

```
              precision    recall  f1-score   support

           0       0.86      0.96      0.91       162
           1       0.95      0.85      0.90       168

    accuracy                           0.90       330
   macro avg       0.91      0.90      0.90       330
weighted avg       0.91      0.90      0.90       330
```

1.2 Great Job!