02-Logistic Regression Project

July 16, 2021

___ # 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!)

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```
%matplotlib inline
```

0.2 Get the Data

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

[4]:
```
ad_data = pd.read_csv('advertising.csv')
```

Check the head of ad_data

[5]:
```
ad_data.head()
```

[5]:
```
   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**

[6]:
```
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
```

[ ]:
```

```

[7]:
```
ad_data.describe()
```

[7]:
```
       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

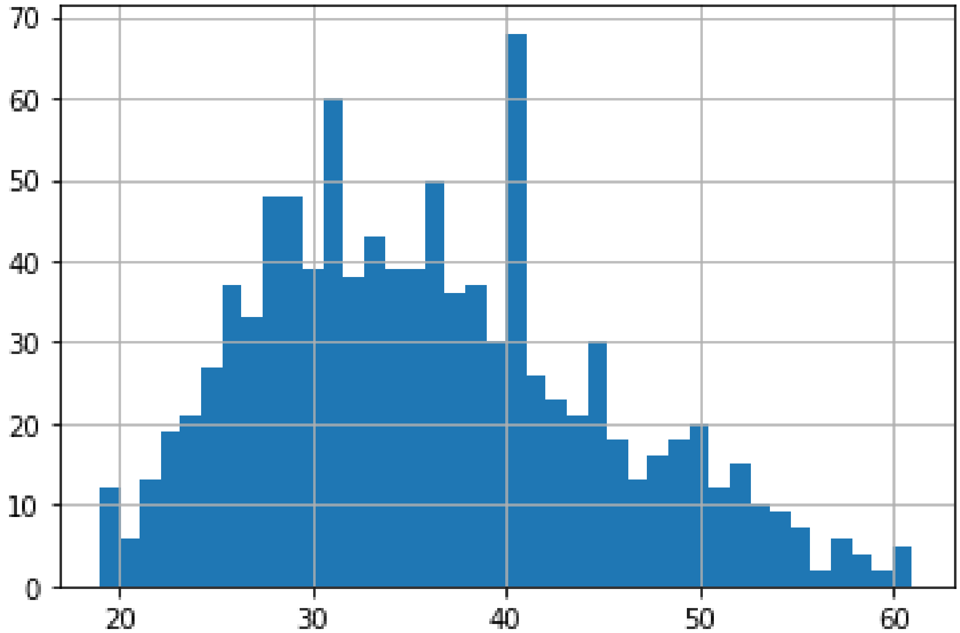
Let's use seaborn to explore the data!

Try recreating the plots shown below!

** Create a histogram of the Age**

[8]:
```
ad_data['Age'].hist(bins=40)
```

[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x162542927c8>
```

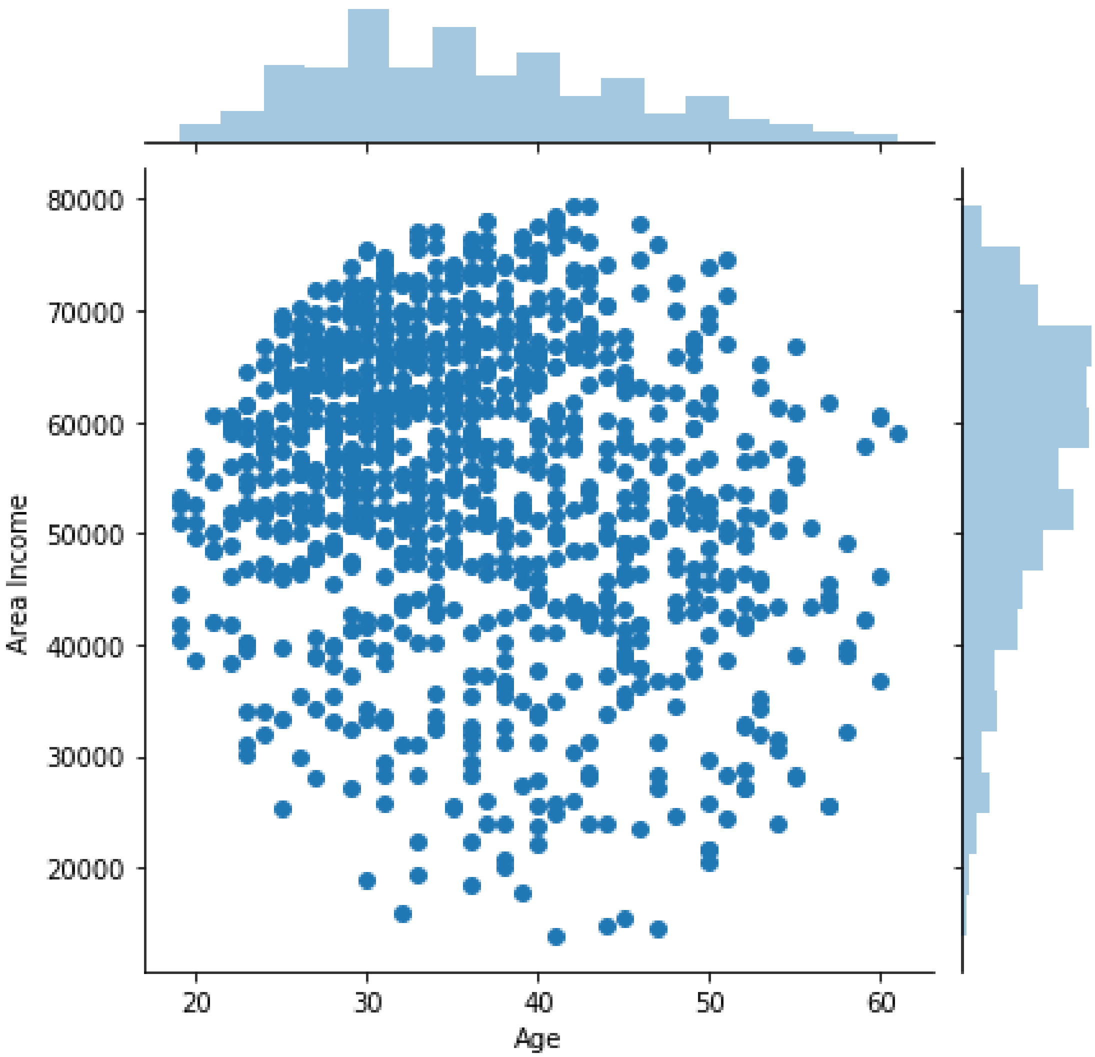



```
[ ]:
```

Create a jointplot showing Area Income versus Age.

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

```
[9]: <seaborn.axisgrid.JointGrid at 0x16254aba0c8>
```

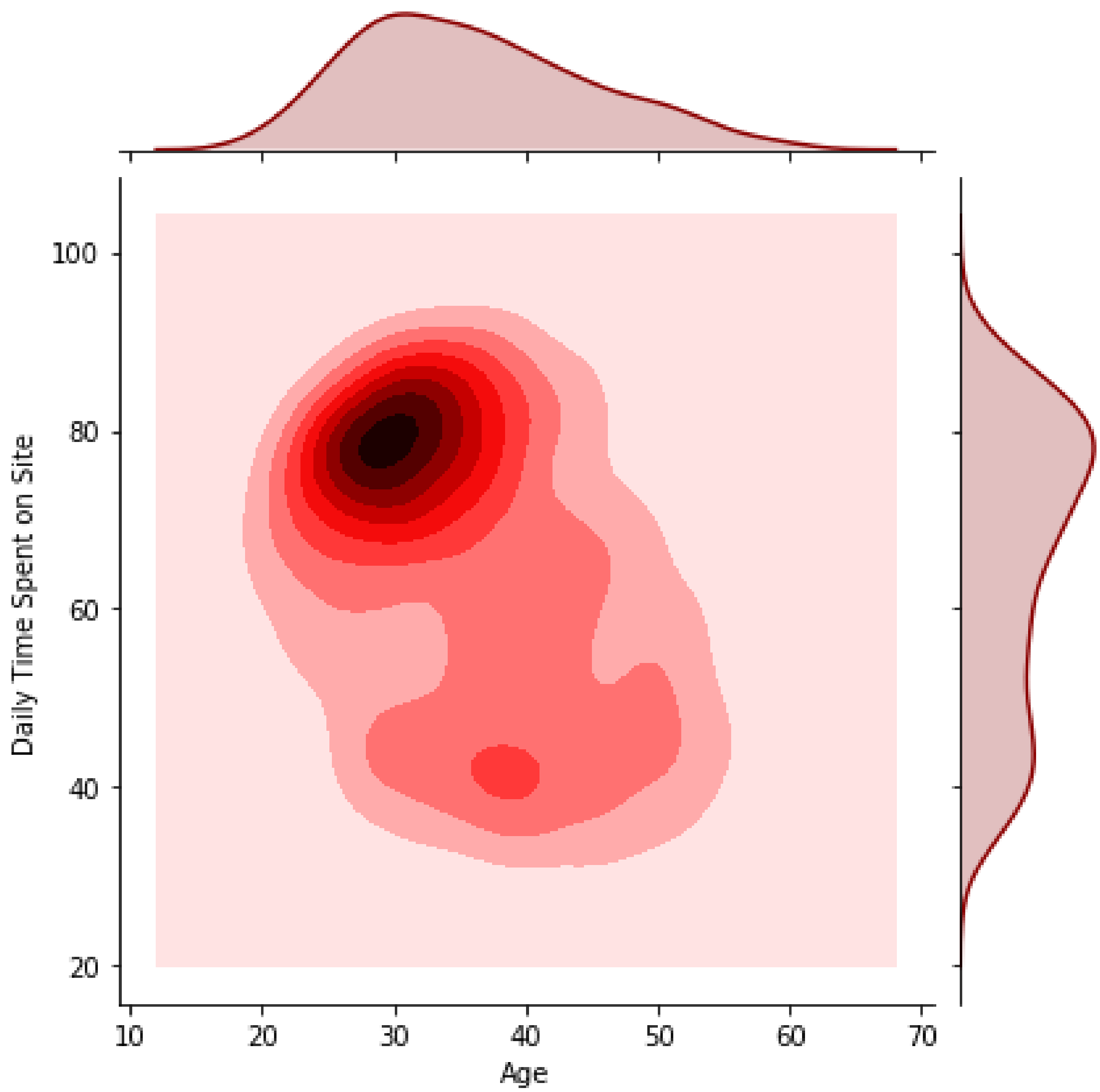



```
[ ]:
```

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

```
[10]: sns.jointplot(x=ad_data['Age'],y=ad_data['Daily Time Spent on␣
      ↪Site'],kind='kdeplot',color='darkred')
```

```
[10]: <seaborn.axisgrid.JointGrid at 0x16254c2ad88>
```

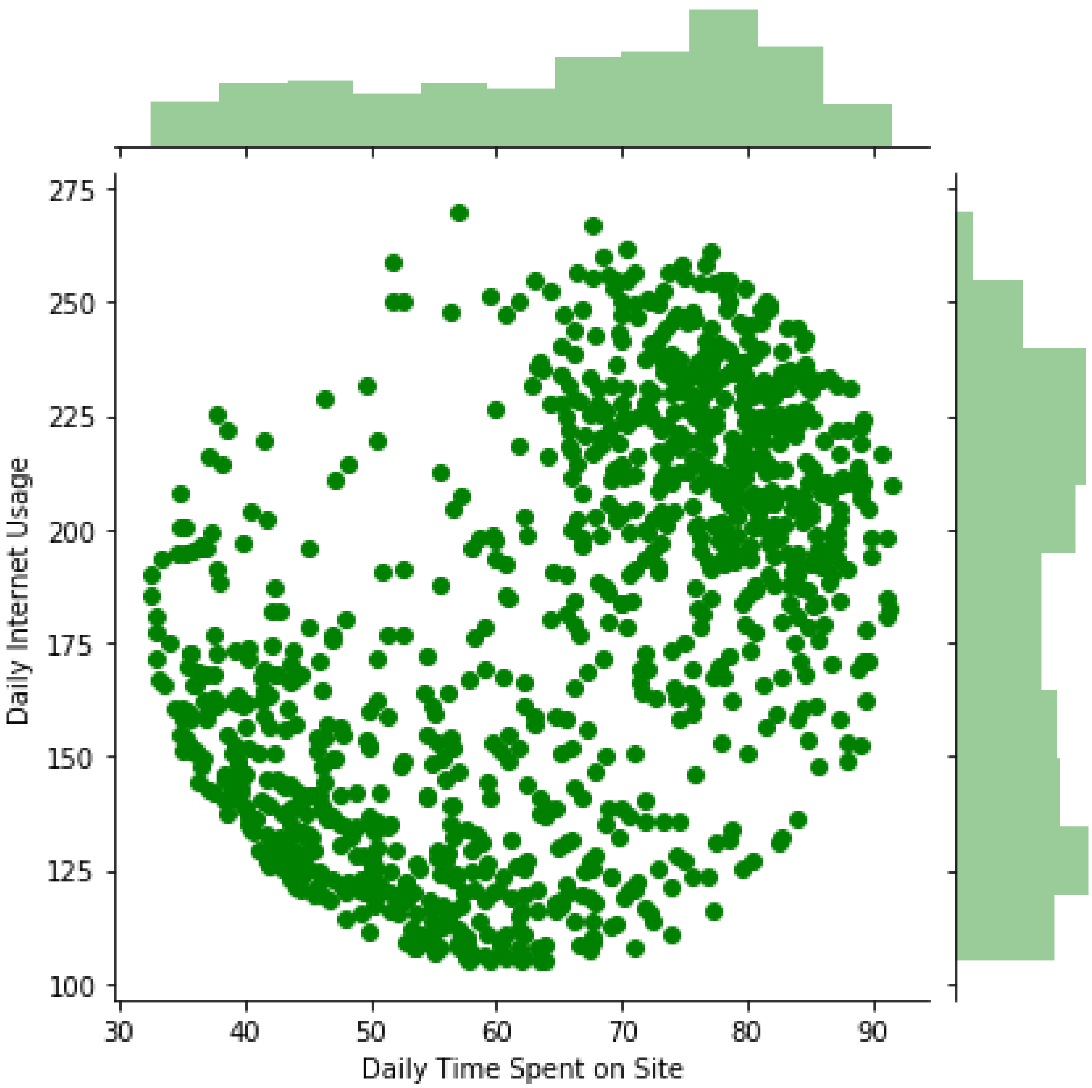



```
[ ]:
```

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

```
[11]: sns.jointplot(x=ad_data['Daily Time Spent on Site'],y=ad_data['Daily Internet␣
       ↪Usage'],color='Green')
```

```
[11]: <seaborn.axisgrid.JointGrid at 0x16254d84f08>
```

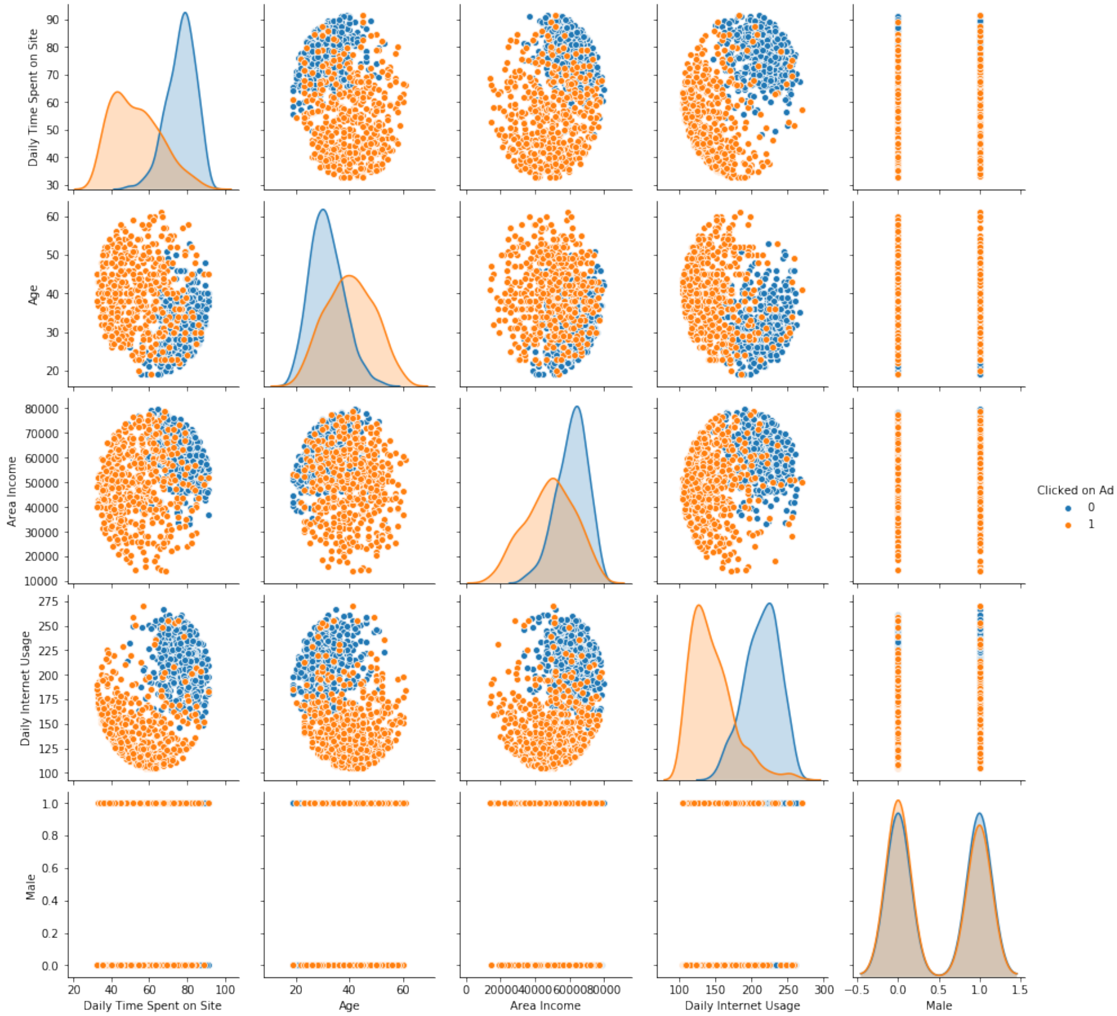


[ ]:

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

[12]:
```
sns.pairplot(ad_data,hue='Clicked on Ad')
```

[12]: `<seaborn.axisgrid.PairGrid at 0x16254ed5208>`



```
[ ]:
```

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**

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

```
[37]:    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
```

[40]:
```python
from sklearn.model_selection import train_test_split
```

[42]:
```python
ad_data['Ad Topic Line'].count()
```

[42]:
```
1000
```

[48]:
```python
ad_data.head()
```

[48]:
```
   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

             City  Male  Clicked on Ad
0     Wrightburgh     0              0
1       West Jodi     1              0
2        Davidton     0              0
3  West Terrifurt     1              0
4    South Manuel     0              0
```

[51]:
```python
Cities=pd.get_dummies(ad_data['City'])
```

[54]:
```python
ad_data = pd.concat([ad_data,Cities],axis=1)
```

[65]:
```python
ad_data.drop(Cities,axis=1)
```

[65]:
```
     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
..                         ...  ...           ...                     ...
995                      72.97   30      71384.57                  208.58
996                      51.30   45      67782.17                  134.42
997                      51.63   51      42415.72                  120.37
998                      55.55   19      41920.79                  187.95
999                      45.01   26      29875.80                  178.35

               City  Male  Clicked on Ad
0       Wrightburgh     0              0
1         West Jodi     1              0
2          Davidton     0              0
3    West Terrifurt     1              0
4      South Manuel     0              0
..              ...   ...            ...
995       Duffystad     1              1
996     New Darlene     1              1
997   South Jessica     1              1
998     West Steven     0              0
999     Ronniemouth     0              1

[1000 rows x 7 columns]
```

[75]:
```python
ad_data.columns
```

[75]:
```
Index(['Daily Time Spent on Site', 'Age', 'Area Income',
       'Daily Internet Usage', 'Male', 'Clicked on Ad'],
      dtype='object')
```

[76]:
```python
X = ad_data[['Daily Time Spent on Site', 'Age', 'Area Income',
       'Daily Internet Usage', 'Male']]
```

[78]:
```python
y= ad_data['Clicked on Ad']
```

[82]:
```python
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.
 ↪3,random_state=101)
```

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

[79]:
```python
from sklearn.linear_model import LogisticRegression
```

[81]:
```python
logr = LogisticRegression()
logr
```

[81]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)

```
[83]: logr.fit(X_train,y_train)
```

```
[83]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

1.1 Predictions and Evaluations

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

```
[84]: predictions = logr.predict(X_test)
```

```
[ ]:
```

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

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

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

```
              precision    recall  f1-score   support

           0       0.91      0.95      0.93       157
           1       0.94      0.90      0.92       143

    accuracy                           0.93       300
   macro avg       0.93      0.93      0.93       300
weighted avg       0.93      0.93      0.93       300
```

```
[ ]:
```

1.2 Great Job!