Logistic Regression Project

خب، خب، در این پروژه قصد داریم با کار روی دیتاست یک شرکت تبلیغاتی، این که کاربر روی تبلیغی که به او نمایش داده شده کلیک می کند یا خیر را پیش بینی کنیم.

Import Libraries

کتابخانه های مورد نیاز را ایمپورت کنید.

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

Get the Data

دیتاست مورد نظر را فرخوانی کنید و نگاه خلاصه ای به آن بیندازید و همچنین اطلاعات کلی و آماری آن را نمایش دهید.

In [96]:
```
data = pd.read_csv('advertising.csv')
```

In [8]:
```
data
```

Out[8]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timestamp
0	68.95	35	61833.90	256.09	Cloned 5thgeneration orchestration	Wrightburgh	0	Tunisia	2016-03-27 00:53:11
1	80.23	31	68441.85	193.77	Monitored national standardization	West Jodi	1	Nauru	2016-04-04 01:39:02
2	69.47	26	59785.94	236.50	Organic bottom-line service-desk	Davidton	0	San Marino	2016-03-13 20:35:42
3	74.15	29	54806.18	245.89	Triple-buffered reciprocal time-frame	West Terrifurt	1	Italy	2016-01-10 02:31:19
4	68.37	35	73889.99	225.58	Robust logistical utilization	South Manuel	0	Iceland	2016-06-03 03:36:18
...	...	...	...	...	...	...	...	...	...
995	72.97	30	71384.57	208.58	Fundamental modular algorithm	Duffystad	1	Lebanon	2016-02-11 21:49:00
996	51.30	45	67782.17	134.42	Grass-roots cohesive monitoring	New Darlene	1	Bosnia and Herzegovina	2016-04-22 02:07:01
997	51.63	51	42415.72	120.37	Expanded intangible solution	South Jessica	1	Mongolia	2016-02-01 17:24:57
998	55.55	19	41920.79	187.95	Proactive bandwidth-monitored policy	West Steven	0	Guatemala	2016-03-24 02:35:54
999	45.01	26	29875.80	178.35	Virtual 5thgeneration emulation	Ronniemouth	0	Brazil	2016-06-03 21:43:21

1000 rows × 10 columns

In [5]:
```
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
```

In [6]:
```
data.describe()
```

Out[6]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Male	Clicked on Ad
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000
mean	65.000200	36.009000	55000.000080	180.000100	0.481000	0.50000
std	15.853615	8.785562	13414.634022	43.902339	0.499889	0.50025
min	32.600000	19.000000	13996.500000	104.780000	0.000000	0.00000
25%	51.360000	29.000000	47031.802500	138.830000	0.000000	0.00000
50%	68.215000	35.000000	57012.300000	183.130000	0.000000	0.50000
75%	78.547500	42.000000	65470.635000	218.792500	1.000000	1.00000
max	91.430000	61.000000	79484.800000	269.960000	1.000000	1.00000

Exploratory Data Analysis (EDA)

حالا کمی در دیتاست گشت و گذار کنیم تا ببینیم به چه اطلاعاتی دست پیدا می کنیم.

برای شروع، نمودار توزیع آماری را بر حسب سن رسم می کنیم.

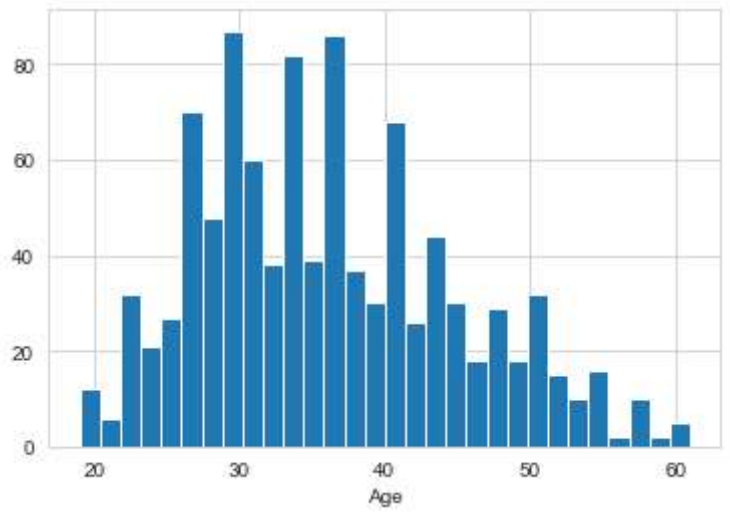
In [7]:
```
data.corr()
```

Out[7]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Male	Clicked on Ad
Daily Time Spent on Site	1.000000	-0.331513	0.310954	0.518658	-0.018951	-0.748117
Age	-0.331513	1.000000	-0.182605	-0.367209	-0.021044	0.492531
Area Income	0.310954	-0.182605	1.000000	0.337496	0.001322	-0.476255
Daily Internet Usage	0.518658	-0.367209	0.337496	1.000000	0.028012	-0.786539
Male	-0.018951	-0.021044	0.001322	0.028012	1.000000	-0.038027
Clicked on Ad	-0.748117	0.492531	-0.476255	-0.786539	-0.038027	1.000000

In [30]:
```
sns.set_style('whitegrid')
data['Age'].hist(bins=30)
plt.xlabel('Age')
```

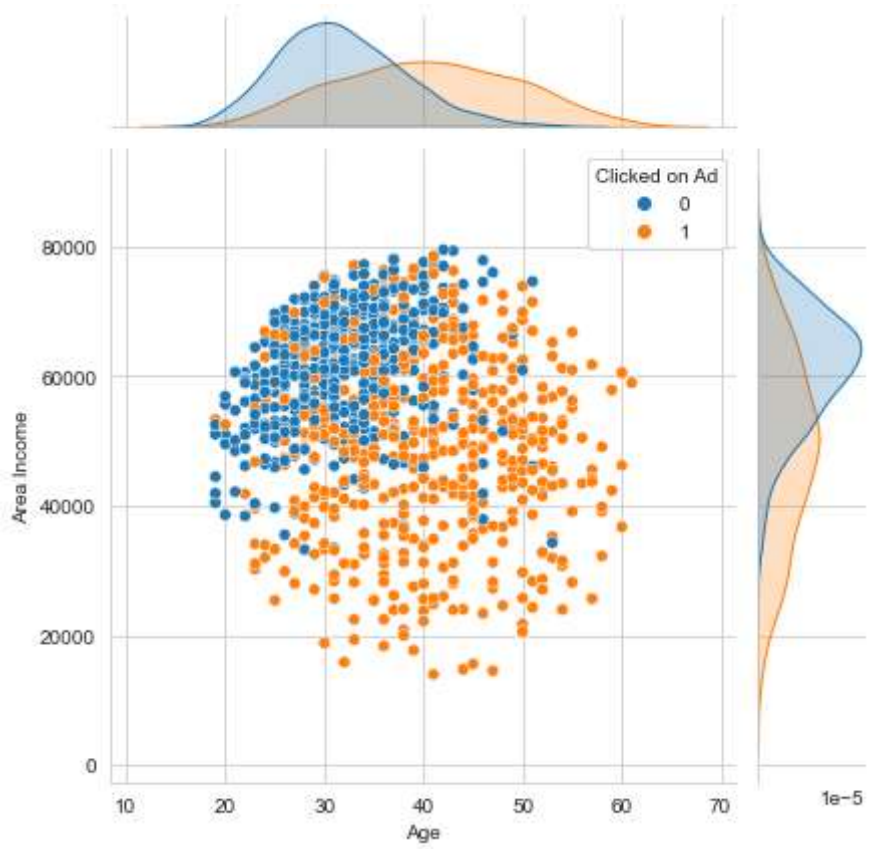
Out[30]:
```
Text(0.5, 0, 'Age')
```



جوینت پلات درآمد بر حسب سن را رسم می کنیم.

In [26]:
```
sns.jointplot(x='Age' ,y='Area Income', data=data, hue='Clicked on Ad')
```

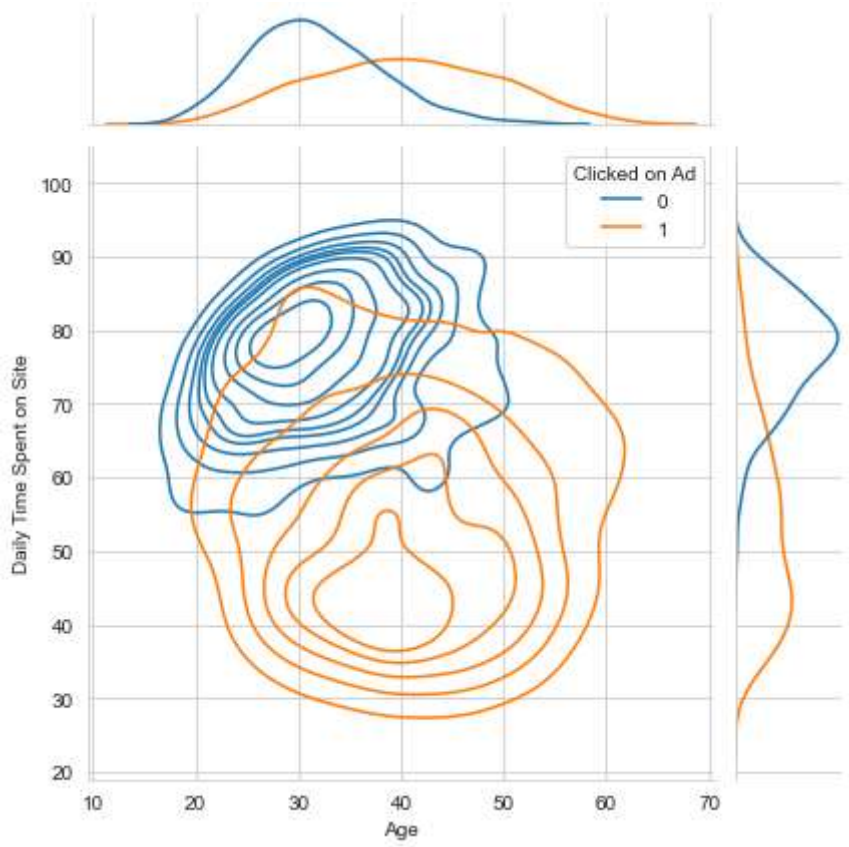
Out[26]: <seaborn.axisgrid.JointGrid at 0x17ba03f5f10>



پلات میزان وقت گذاشته شده روی سایت تبلیغ بر حسب سن را رسم می کنیم. KDE

In [25]:
```
sns.jointplot(x='Age', y='Daily Time Spent on Site', data=data, kind='kde', co
```

Out[25]:
```
<seaborn.axisgrid.JointGrid at 0x17ba0280e50>
```



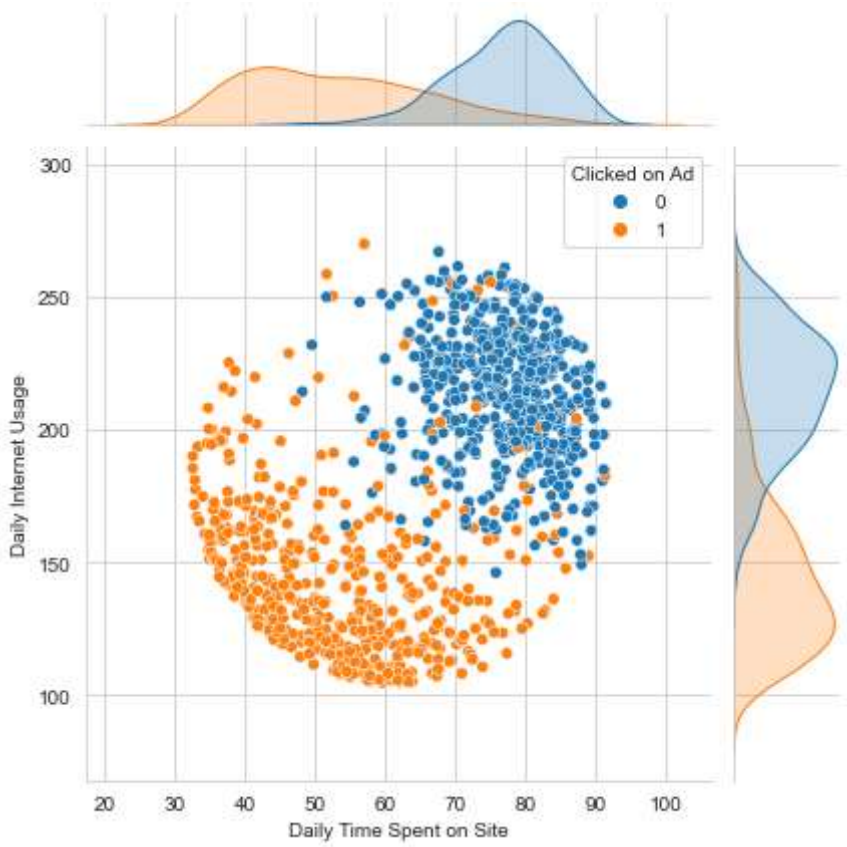
جوینت پلات میزان استفاده از اینترنت بر حسب میزان وقت گذاشته شده روی سایت تبلیغ را رسم می کنیم.

In [21]:
```
data.columns
```

Out[21]:
```
Index(['Daily Time Spent on Site', 'Age', 'Area Income',
       'Daily Internet Usage', 'Ad Topic Line', 'City', 'Male', 'Country',
       'Timestamp', 'Clicked on Ad'],
      dtype='object')
```

In [24]:
```
sns.jointplot(x='Daily Time Spent on Site', y='Daily Internet Usage', data=daf
```

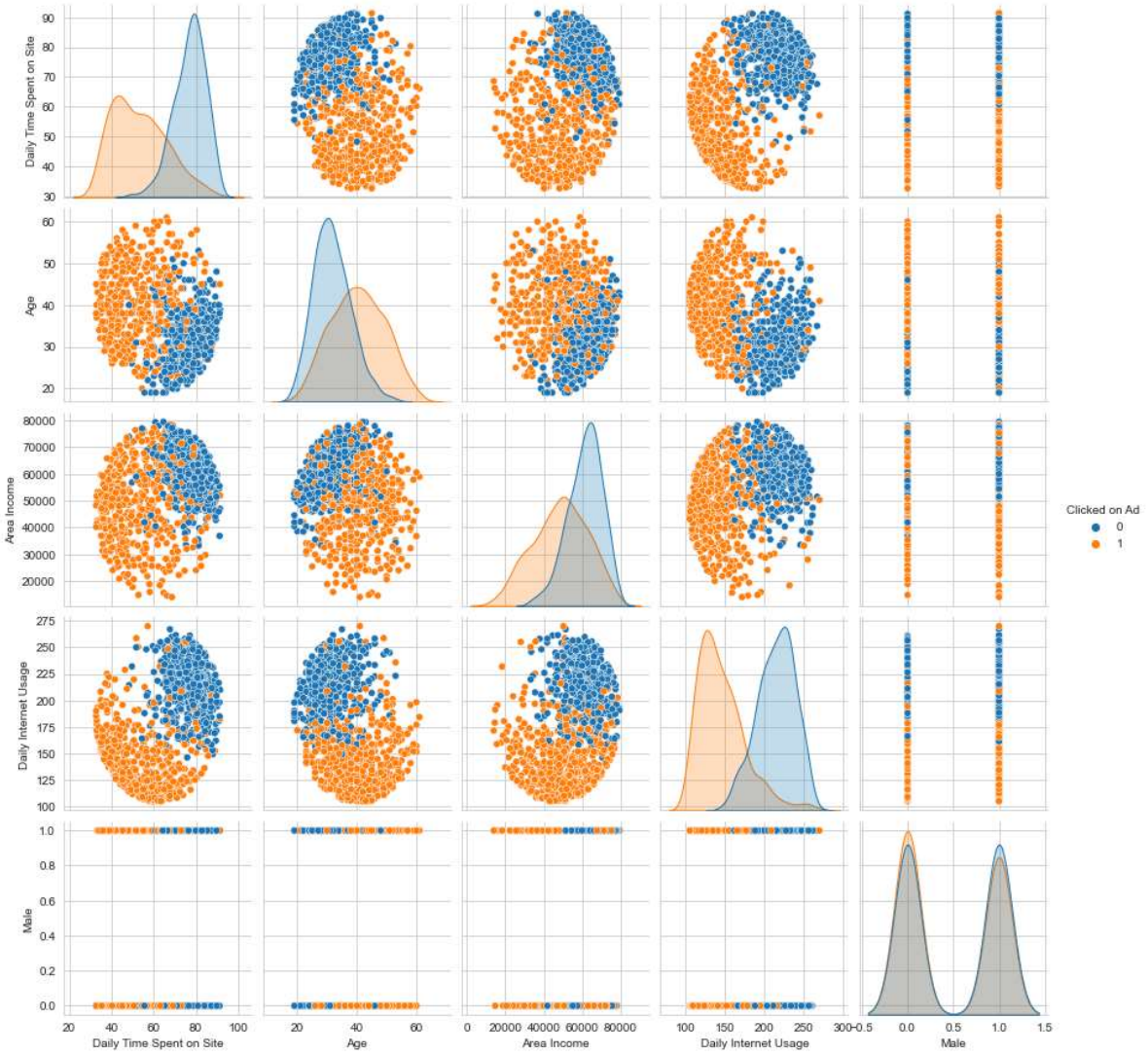
Out[24]: <seaborn.axisgrid.JointGrid at 0x17ba01b4970>



در نهایت یک Pain Plot کلی برای دیتاست، با مشخص کردن تفاوت کسانی که روی تبلیغ کلیک کرده اند و نکرده اند را رسم کنید.

In [23]:
```
sns.pairplot(data, hue='Clicked on Ad')
```

Out[23]: <seaborn.axisgrid.PairGrid at 0x17b9e46be20>



Logistic Regression

پس از ایمپورت متد مربوطه، داده ها را به دو بخش فیچر و لیبل تقسیم کنید و سپس آن ها را به دو بخش ترین و تست تقسیم نمایید.

In [97]:
```
data.head()
```

Out[97]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timestamp	Clicke on A
0	68.95	35	61833.90	256.09	Cloned 5thgeneration orchestration	Wrightburgh	0	Tunisia	2016-03-27 00:53:11	
1	80.23	31	68441.85	193.77	Monitored national standardization	West Jodi	1	Nauru	2016-04-04 01:39:02	
2	69.47	26	59785.94	236.50	Organic bottom-line service-desk	Davidton	0	San Marino	2016-03-13 20:35:42	
3	74.15	29	54806.18	245.89	Triple-buffered reciprocal time-frame	West Terrifurt	1	Italy	2016-01-10 02:31:19	
4	68.37	35	73889.99	225.58	Robust logistical utilization	South Manuel	0	Iceland	2016-06-03 03:36:18	

In [98]:
```
x = data.drop(['Clicked on Ad','Ad Topic Line','City','Country','Timestamp'],
y = data['Clicked on Ad']
```

In [100]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, rando
```

حالا کلاس رگرسیون لاجستیک را بارگزاری نمایید و یک مدل از آن نوع بسازید و با داده های ترین آموزش دهید.

In [101]:
```
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
```

In [102]:
```
logmodel.fit(x_train, y_train)
```

Out[102]:
```
LogisticRegression()
```

Predictions and Evaluations

حالا به کمک مدلی که آموزش داده اید، برای داده های تست پیش بینی کنید که کدام یک روی تبلیغ ها کلیک می کنند.

In [103]:
```
pred = logmodel.predict(x_test)
```

حالا برای مدلی که آموزش داده اید، معیار های ارزیابی همچون Score و Confusion Matrix و Classification Report

In [104]:
```
logmodel.score(x_test, y_test)
```

Out[104]:
```
0.9266666666666666
```

In [105]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(pred, y_test))
print('***********','\n')
print(confusion_matrix(pred, y_test))
```

```
              precision    recall  f1-score   support

           0       0.95      0.91      0.93       163
           1       0.90      0.94      0.92       137

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

***********

[[149  14]
 [  8 129]]
```

حالا میخوام روز های خفته رو هم دخالت بدم توو یادگیری مدلم

In [106]:
```
data['Timestamp'] = pd.to_datetime(data['Timestamp'])
```

In [107]:
```
data.rename({'Timestamp' : 'DOF'}, axis=1, inplace=True)
```

In [108]:
```
data
```

Out[108]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	DOF	C
0	68.95	35	61833.90	256.09	Cloned 5thgeneration orchestration	Wrightburgh	0	Tunisia	2016-03-27 00:53:11	
1	80.23	31	68441.85	193.77	Monitored national standardization	West Jodi	1	Nauru	2016-04-04 01:39:02	
2	69.47	26	59785.94	236.50	Organic bottom-line service-desk	Davidton	0	San Marino	2016-03-13 20:35:42	
3	74.15	29	54806.18	245.89	Triple-buffered reciprocal time-frame	West Terrifurt	1	Italy	2016-01-10 02:31:19	
4	68.37	35	73889.99	225.58	Robust logistical utilization	South Manuel	0	Iceland	2016-06-03 03:36:18	
...	...	...	...	...	...	...	...	...	...	
995	72.97	30	71384.57	208.58	Fundamental modular algorithm	Duffystad	1	Lebanon	2016-02-11 21:49:00	
996	51.30	45	67782.17	134.42	Grass-roots cohesive monitoring	New Darlene	1	Bosnia and Herzegovina	2016-04-22 02:07:01	
997	51.63	51	42415.72	120.37	Expanded intangible solution	South Jessica	1	Mongolia	2016-02-01 17:24:57	
998	55.55	19	41920.79	187.95	Proactive bandwidth-monitored policy	West Steven	0	Guatemala	2016-03-24 02:35:54	
999	45.01	26	29875.80	178.35	Virtual 5thgeneration emulation	Ronniemouth	0	Brazil	2016-06-03 21:43:21	

1000 rows × 10 columns

In [109]:
```
categorical_day = pd.get_dummies(data['DOF'].apply(lambda x:x.dayofweek))
data = pd.concat([data, categorical_day], axis=1)
```

In [110]:
```
data
```

Out[110]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	DOF	C
0	68.95	35	61833.90	256.09	Cloned 5thgeneration orchestration	Wrightburgh	0	Tunisia	2016-03-27 00:53:11	
1	80.23	31	68441.85	193.77	Monitored national standardization	West Jodi	1	Nauru	2016-04-04 01:39:02	
2	69.47	26	59785.94	236.50	Organic bottom-line service-desk	Davidton	0	San Marino	2016-03-13 20:35:42	
3	74.15	29	54806.18	245.89	Triple-buffered reciprocal time-frame	West Terrifurt	1	Italy	2016-01-10 02:31:19	
4	68.37	35	73889.99	225.58	Robust logistical utilization	South Manuel	0	Iceland	2016-06-03 03:36:18	
...	...	...	...	...	...	...	...	...	...	
995	72.97	30	71384.57	208.58	Fundamental modular algorithm	Duffystad	1	Lebanon	2016-02-11 21:49:00	
996	51.30	45	67782.17	134.42	Grass-roots cohesive monitoring	New Darlene	1	Bosnia and Herzegovina	2016-04-22 02:07:01	
997	51.63	51	42415.72	120.37	Expanded intangible solution	South Jessica	1	Mongolia	2016-02-01 17:24:57	
998	55.55	19	41920.79	187.95	Proactive bandwidth-monitored policy	West Steven	0	Guatemala	2016-03-24 02:35:54	
999	45.01	26	29875.80	178.35	Virtual 5thgeneration emulation	Ronniemouth	0	Brazil	2016-06-03 21:43:21	

1000 rows × 17 columns

In [111]:
```
data.drop(['DOF'], axis=1, inplace=True)
```

In [112]:
```
data
```

Out[112]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Clicked on Ad	0
0	68.95	35	61833.90	256.09	Cloned 5thgeneration orchestration	Wrightburgh	0	Tunisia	0	0
1	80.23	31	68441.85	193.77	Monitored national standardization	West Jodi	1	Nauru	0	1
2	69.47	26	59785.94	236.50	Organic bottom-line service-desk	Davidton	0	San Marino	0	0
3	74.15	29	54806.18	245.89	Triple-buffered reciprocal time-frame	West Terrifurt	1	Italy	0	0
4	68.37	35	73889.99	225.58	Robust logistical utilization	South Manuel	0	Iceland	0	0
...	...	...	...	...	...	...	...	...	...	...
995	72.97	30	71384.57	208.58	Fundamental modular algorithm	Duffystad	1	Lebanon	1	0
996	51.30	45	67782.17	134.42	Grass-roots cohesive monitoring	New Darlene	1	Bosnia and Herzegovina	1	0
997	51.63	51	42415.72	120.37	Expanded intangible solution	South Jessica	1	Mongolia	1	1
998	55.55	19	41920.79	187.95	Proactive bandwidth-monitored policy	West Steven	0	Guatemala	0	0
999	45.01	26	29875.80	178.35	Virtual 5thgeneration emulation	Ronniemouth	0	Brazil	1	0

1000 rows × 16 columns

In [113]:
```
x = data.drop(['Daily Internet Usage','City','Ad Topic Line','Country','Clicke
y = data['Clicked on Ad']
```

In [114]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, rando
```

In [115]:
```
from sklearn.linear_model import LogisticRegression
logmodel2 = LogisticRegression()
```

In [116]:
```
logmodel2.fit(x_train, y_train)
```

```
C:\Users\ASUS\anaconda3\lib\site-packages\sklearn\utils\validation.py:1688:
FutureWarning: Feature names only support names that are all strings. Got fe
ature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

Out[116]:
```
LogisticRegression()
```

In [117]:
```
pred2 = logmodel2.predict(x_test)
```

```
C:\Users\ASUS\anaconda3\lib\site-packages\sklearn\utils\validation.py:1688:
FutureWarning: Feature names only support names that are all strings. Got fe
ature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

In [118]:
```
logmodel2.score(x_test, y_test)
```

```
C:\Users\ASUS\anaconda3\lib\site-packages\sklearn\utils\validation.py:1688:
FutureWarning: Feature names only support names that are all strings. Got fe
ature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

Out[118]:
```
0.9266666666666666
```

In [119]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

In [120]:
```
print(confusion_matrix(pred2, y_test))
```

```
[[147  12]
 [ 10 131]]
```

In [121]:
```
print(classification_report(pred2, y_test))
```

```
              precision    recall  f1-score   support

           0       0.94      0.92      0.93       159
           1       0.92      0.93      0.92       141

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

The End :)