Advertisement Click - Logistic Regression

February 3, 2022

Advertisement Click - Logistic Regression

by Joseph Ramon

Will be working with an advertising data set, indicating whether or not a particular internet user clicked on an Advertisement.

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

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

0.1 Get the Data

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

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

[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
```

[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
```

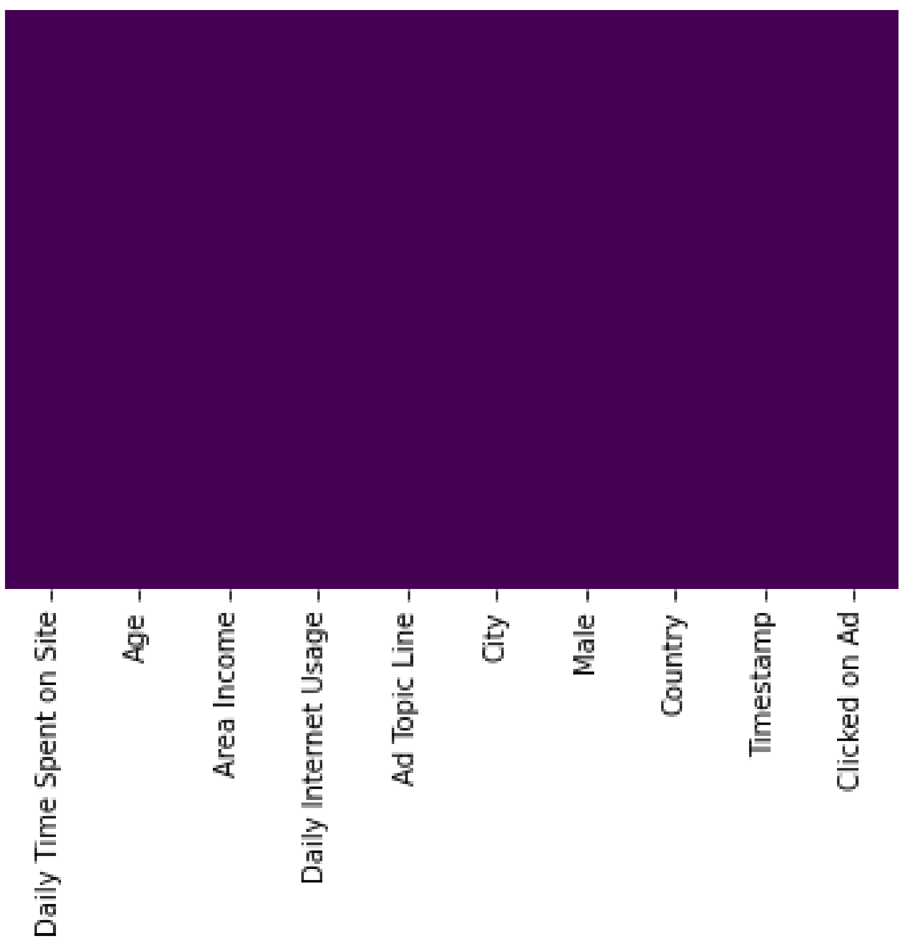
```
max                   91.430000    61.000000  79484.800000

       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
```

0.2 Exploratory Data Analysis

0.3 Missing Data

[6]:
```python
sns.heatmap(ad_data.isnull(),yticklabels=False,cbar=False,cmap='viridis')
# the heatmap below does not show any nulls
```

[6]: <AxesSubplot:>



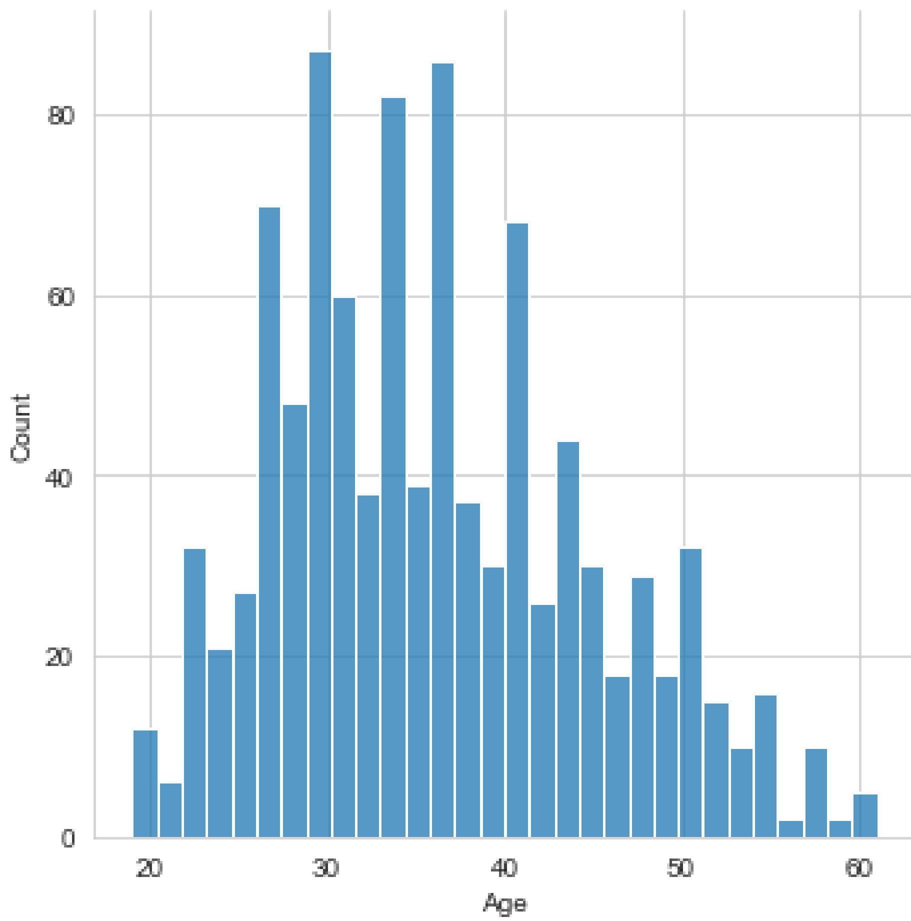
0.4 Some Plots

[7]:
```
sns.set_style('whitegrid')
```

Age Histogram - using seaborn

[41]:
```
sns.displot(ad_data['Age'],bins=30)
```

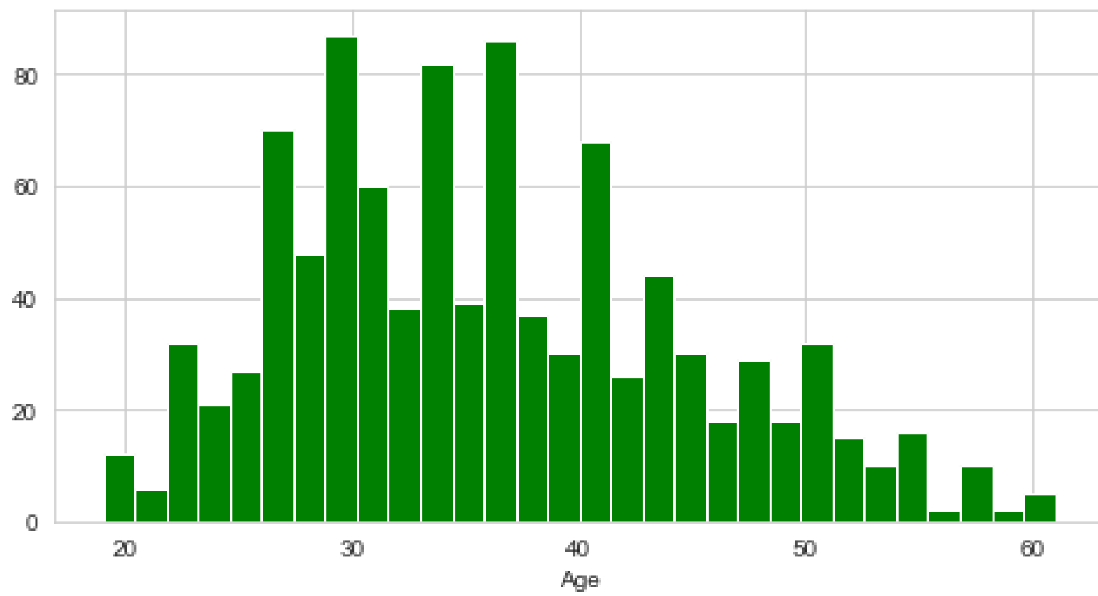
[41]: <seaborn.axisgrid.FacetGrid at 0x20e40861cd0>



Age Histogram - using built-in

[37]:
```
ad_data['Age'].hist(color='green',bins=30,figsize=(8,4))
plt.xlabel('Age')
```

[37]: Text(0.5, 0, 'Age')



Age Histogram - interactive, using cufflinks

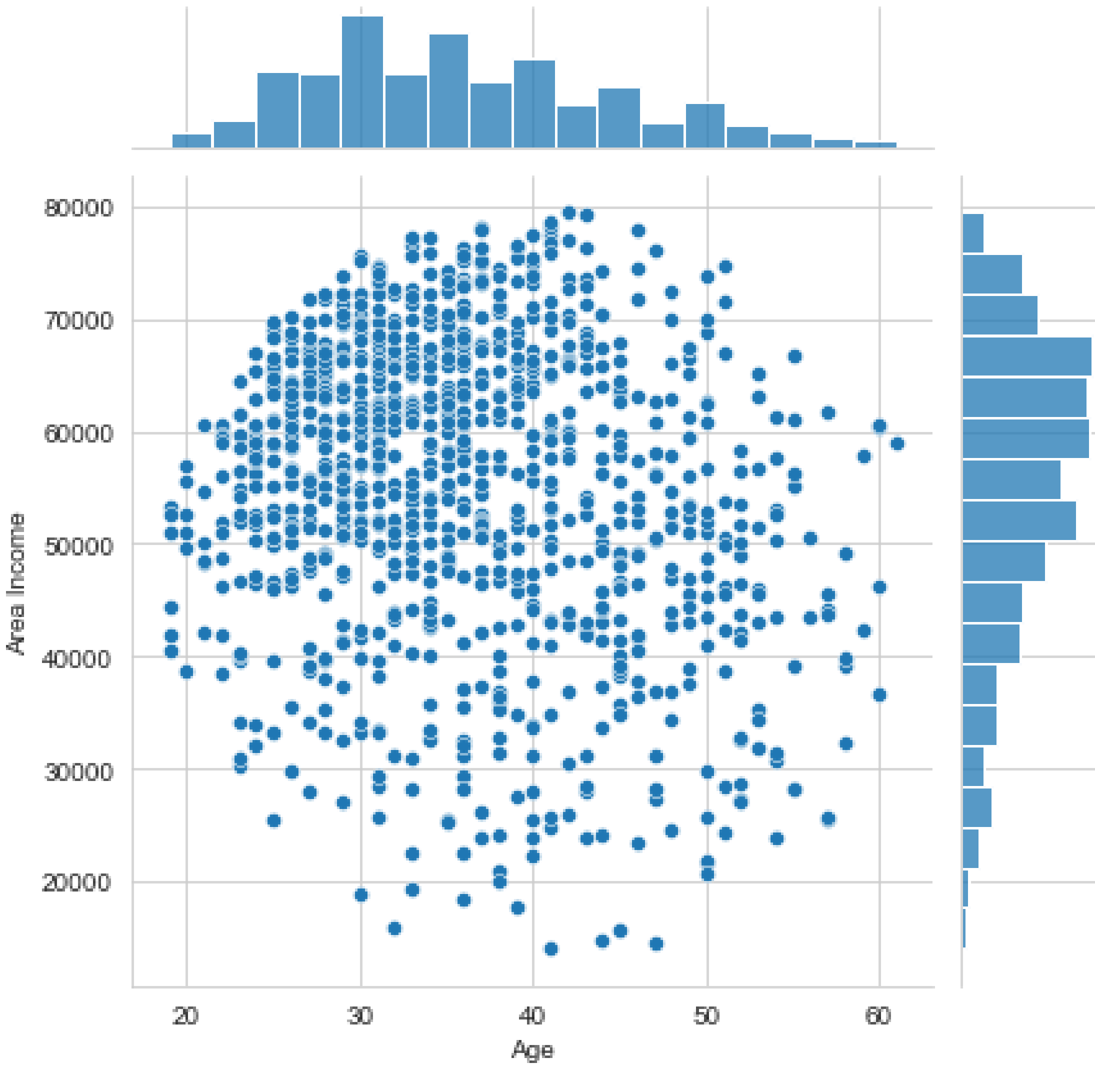
[86]:
```
import cufflinks as cf
cf.go_offline()
```

[87]:
```
# more interactive version using cufflinks
ad_data['Age'].iplot(kind='hist',bins=30,color='green',\
                     xTitle="Age", yTitle="Count", title="Age Count")
```

jointplot showing Area Income versus Age

[23]:
```
sns.jointplot(x='Age',y='Area Income',data=ad_data,kind='scatter')
```

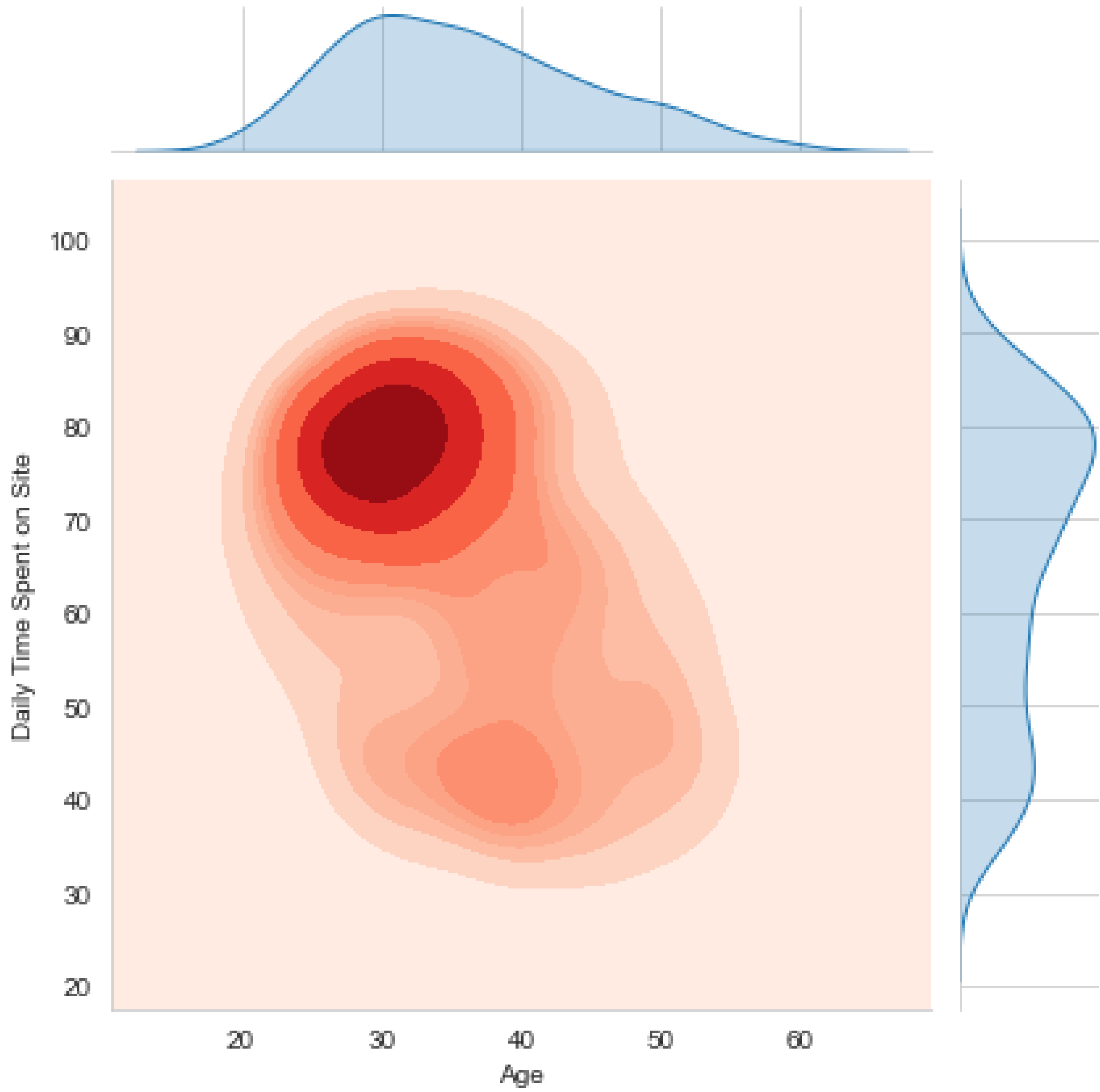
[23]:
```
<seaborn.axisgrid.JointGrid at 0x20e3eff7d90>
```



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

[50]:
```
sns.jointplot(x='Age',y='Daily Time Spent on Site',data=ad_data,\
              fill=True, thresh=0,cmap='Reds',kind='kde')
```

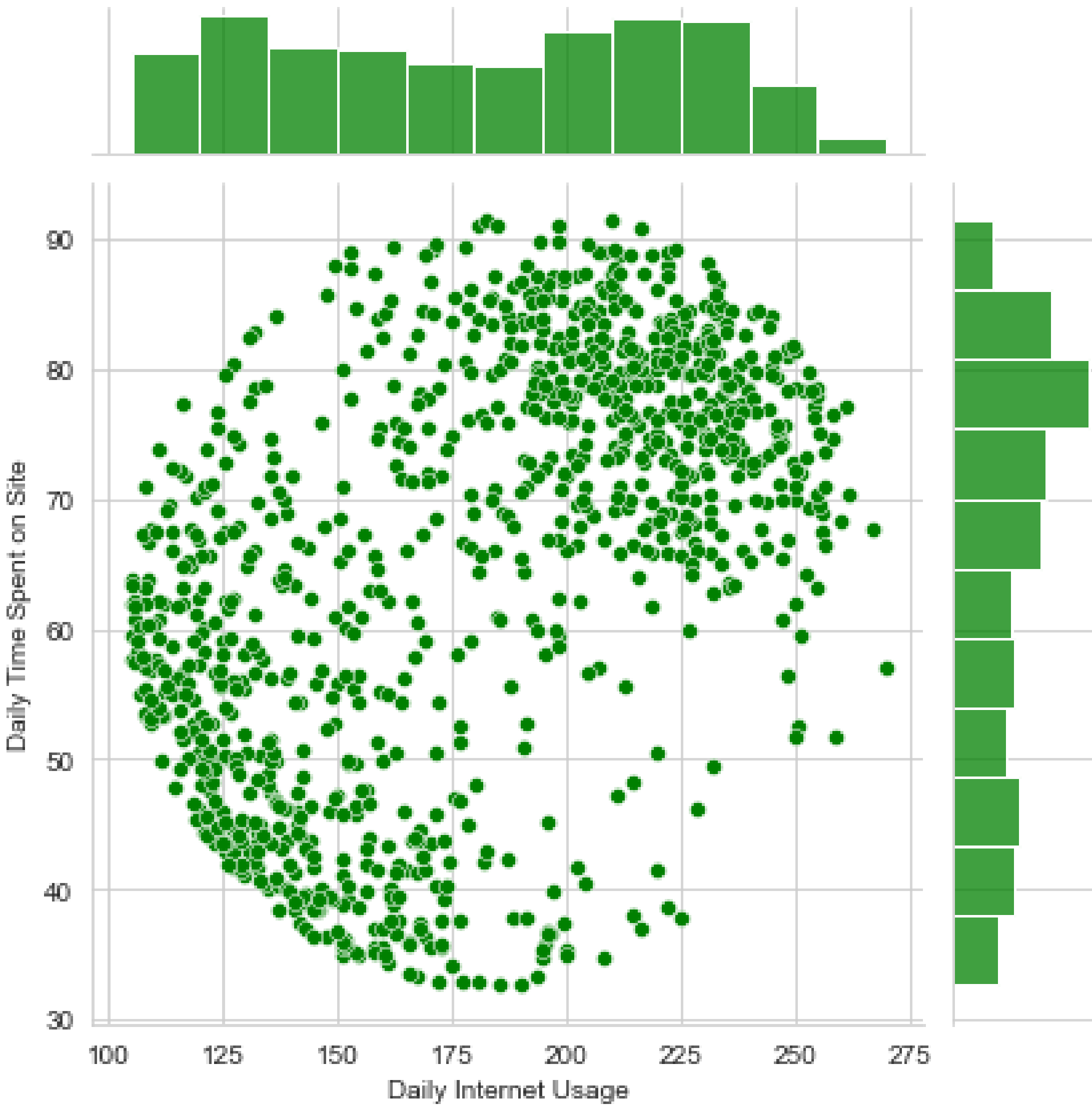
[50]: <seaborn.axisgrid.JointGrid at 0x20e41daa9a0>



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

[55]:
```
sns.jointplot(x='Daily Internet Usage',y='Daily Time Spent on Site',\
              data=ad_data,color="Green")
```

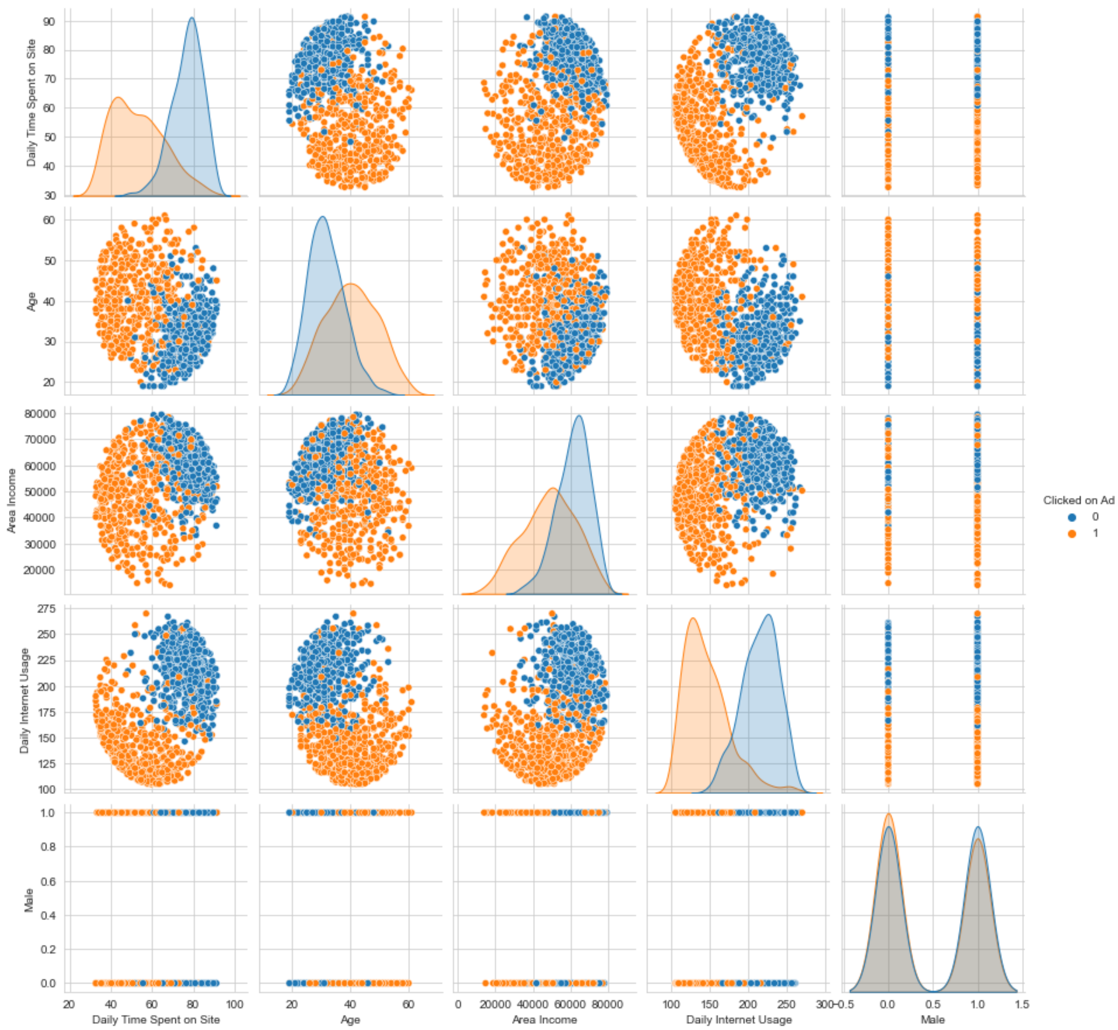
[55]: <seaborn.axisgrid.JointGrid at 0x20e43344d30>



pairplot with the hue defined by the 'Clicked on Ad' column feature.

[58]:
```
sns.pairplot(ad_data,hue='Clicked on Ad')
```

[58]: <seaborn.axisgrid.PairGrid at 0x20e447fe640>



1 Build a Logistic Regression model

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

1.1 Train Test Split

[64]:
```
ad_data.head(2)
```

[64]:
```
   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

                         Ad Topic Line         City  Male  Country  \
0  Cloned 5thgeneration orchestration  Wrightburgh     0  Tunisia
1  Monitored national standardization    West Jodi     1    Nauru
```

```
            Timestamp  Clicked on Ad
0  2016-03-27 00:53:11              0
1  2016-04-04 01:39:02              0
```

[65]:
```python
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
```

[68]:
```python
ad_data.columns
```

[68]:
```
Index(['Daily Time Spent on Site', 'Age', 'Area Income',
       'Daily Internet Usage', 'Ad Topic Line', 'City', 'Male', 'Country',
       'Timestamp', 'Clicked on Ad'],
      dtype='object')
```

[69]:
```python
from sklearn.model_selection import train_test_split
```

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

[77]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y,\
                test_size=0.30, random_state=101)
```

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

[78]:
```python
from sklearn.linear_model import LogisticRegression
```

[79]:
```python
logmodel = LogisticRegression()
```

[80]:
```
logmodel.fit(X_train,y_train)
```

[80]:
```
LogisticRegression()
```

1.2 Predictions and Evaluations

predict values for the testing data

[94]:
```
predictions = logmodel.predict(X_test)
```

Create a classification report for the model.

[83]:
```
from sklearn.metrics import classification_report
```

[85]:
```
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
```