Logistic_Regression Example

December 12, 2019

1 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

[1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/matplotlib/__init__.py:886: MatplotlibDeprecationWarning:
examples.directory is deprecated; in the future, examples will be found relative
to the 'datapath' directory.
  "found relative to the 'datapath' directory.".format(key))
```

1.1 Get the Data

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

[3]:

```
df = pd.read_csv('advertising.csv')
```

Check the head of ad_data

[4]:

```
df.head(5)
```

```
[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]: df.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
```

```
[8]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
Daily Time Spent on Site    1000 non-null float64
Age                         1000 non-null int64
Area Income                 1000 non-null float64
Daily Internet Usage        1000 non-null float64
Ad Topic Line               1000 non-null object
City                        1000 non-null object
Male                        1000 non-null int64
Country                     1000 non-null object
Timestamp                   1000 non-null object
Clicked on Ad               1000 non-null int64
dtypes: float64(3), int64(3), object(4)
memory usage: 78.2+ KB
```

1.2 Exploratory Data Analysis

Let's use seaborn to explore the data!

Try recreating the plots shown below!

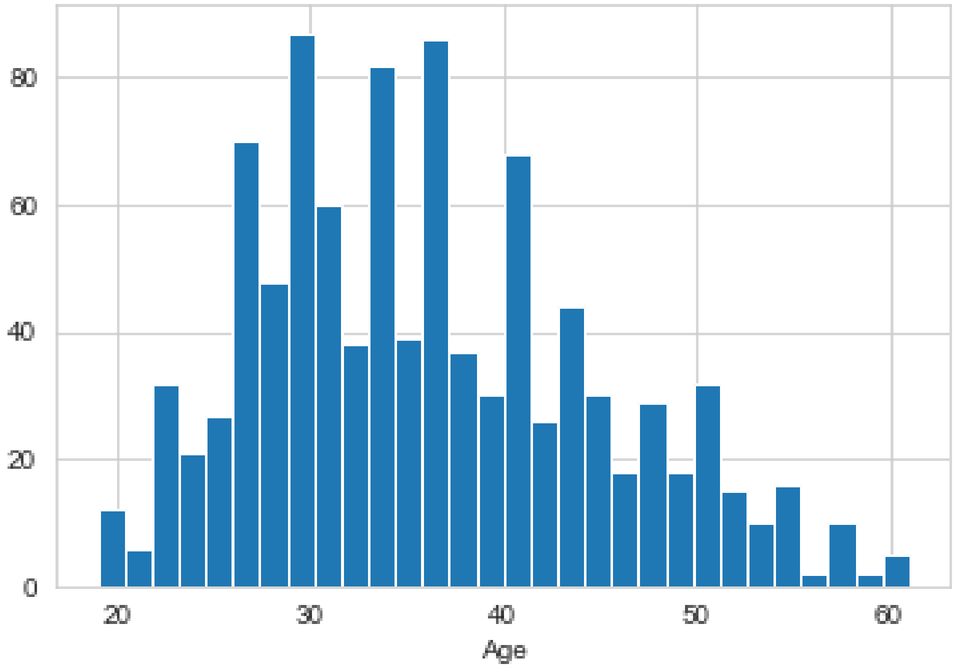
** Create a histogram of the Age**

[10]:

```python
sns.set_style('whitegrid')
df['Age'].hist(bins=30)
plt.xlabel('Age')
```

[10]:

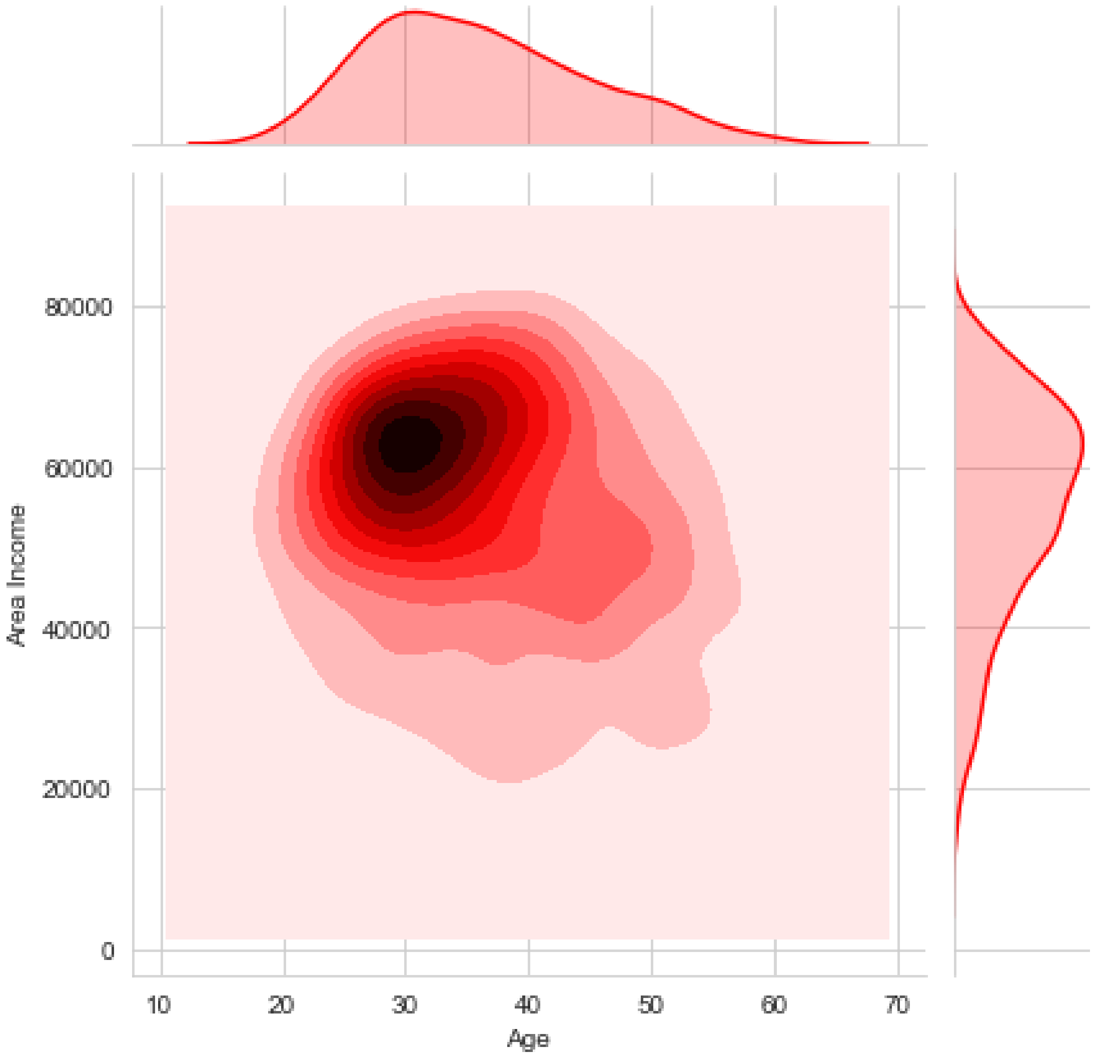
```
Text(0.5, 0, 'Age')
```



Create a jointplot showing Area Income versus Age.

```
[17]: sns.jointplot(x = 'Age', y = 'Area Income', data = df, kind = 'kde', color =␣
      ↪'r')
```

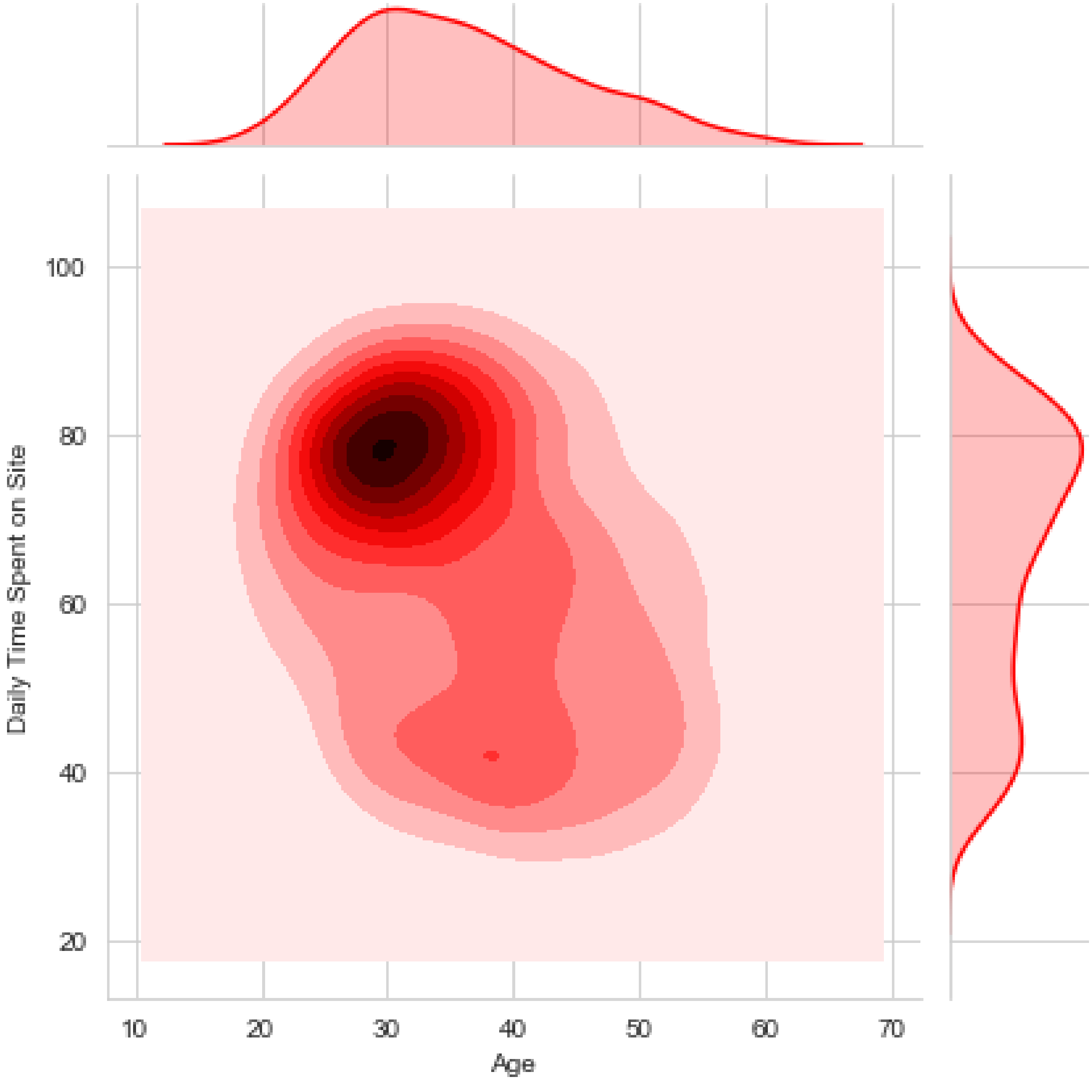
```
[17]: <seaborn.axisgrid.JointGrid at 0x12802f7d0>
```



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

```
[18]: sns.jointplot(x = 'Age', y = 'Daily Time Spent on Site', data = df, kind =␣
      ↪'kde', color = 'r')
```

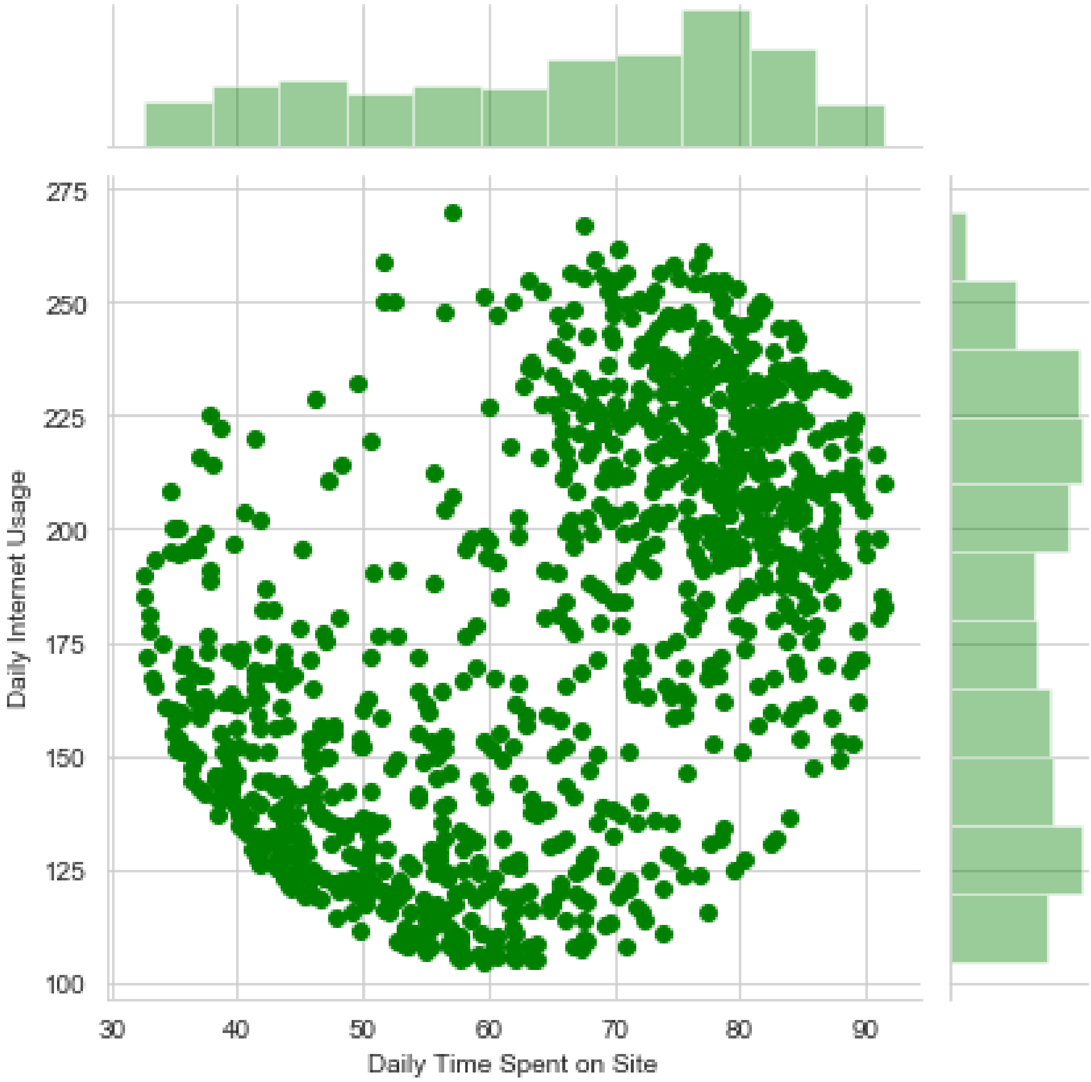
```
[18]: <seaborn.axisgrid.JointGrid at 0x128222fd0>
```



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

```
[20]: sns.jointplot(x = 'Daily Time Spent on Site', y = 'Daily Internet Usage', data
      ↪= df, color = 'g')
```

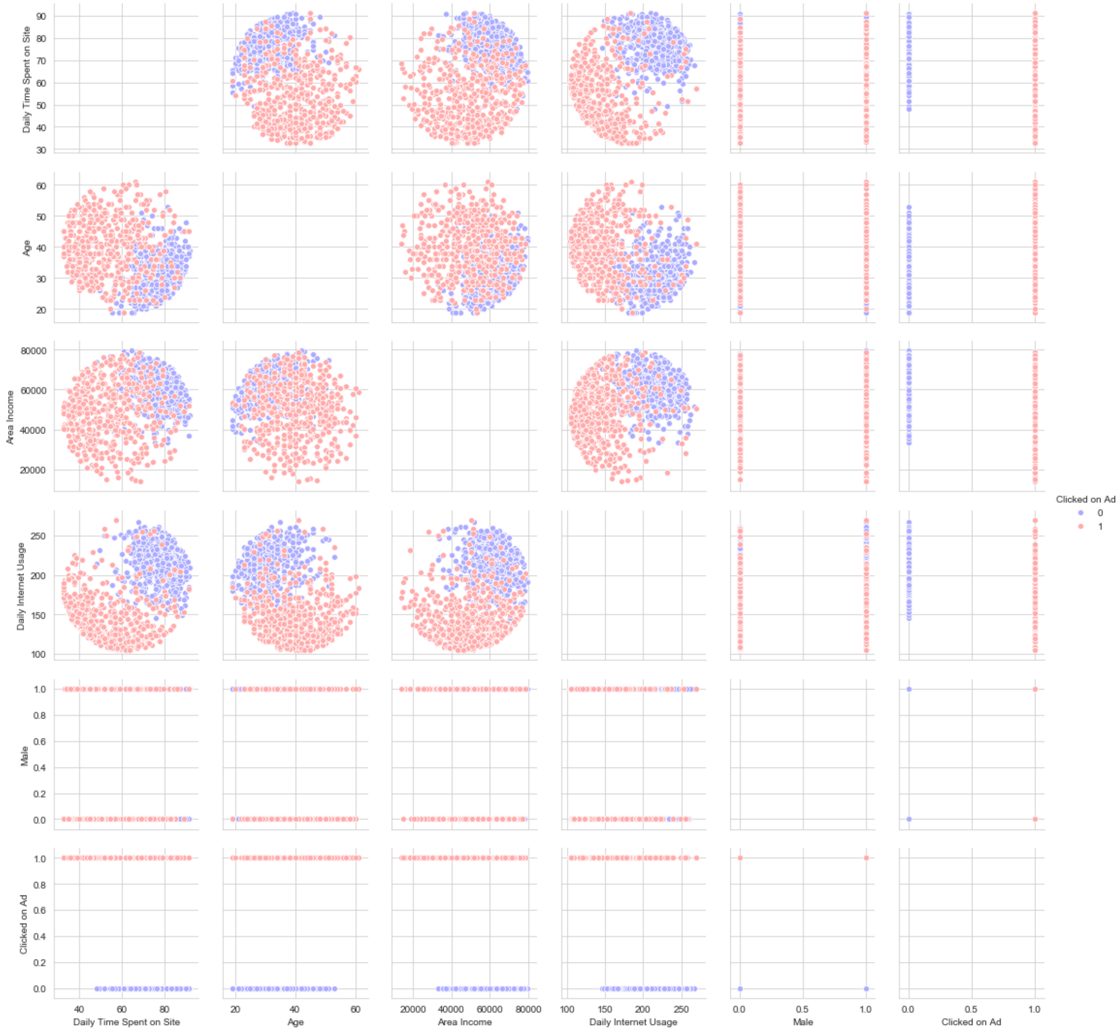
```
[20]: <seaborn.axisgrid.JointGrid at 0x125b2a350>
```



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

```
[28]: sns.pairplot(df,hue='Clicked on Ad',palette='bwr', diag_kind = 'scatter')
```

```
[28]: <seaborn.axisgrid.PairGrid at 0x1329536d0>
```



2 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**

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

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

```
[34]: 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.**

[35]:

```
from sklearn.linear_model import LogisticRegression
```

[36]:

```
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
```

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

[36]:

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

2.1 Predictions and Evaluations

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

[38]:

```
predictions = logmodel.predict(X_test)
```

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

[39]:

```
from sklearn.metrics import classification_report
```

[40]:

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.87      0.96      0.91       162
           1       0.96      0.86      0.91       168

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