Excercise_6_solutions_part_2

May 30, 2019

1 PART 2 - Logistic Regression Project

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

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

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

```
Out[2]:    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
```

```
In [3]: ad_data.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
```

```
In [4]: ad_data.describe()
```

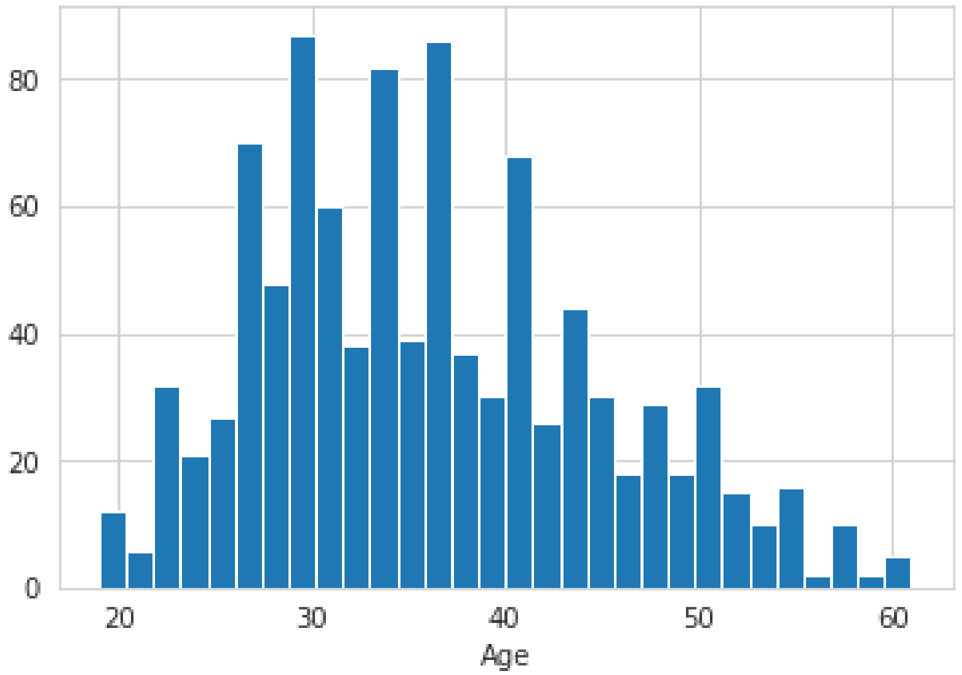
```
Out[4]:        Daily Time Spent on Site          Age   Area Income  \
        count               1000.000000  1000.000000   1000.000000
        mean                  65.000200    36.009000  55000.000080
        std                   15.853615     8.785562  13414.634022
        min                   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
```

2 Exploratory Data Analysis

```
In [5]: sns.set_style('whitegrid')
        ad_data['Age'].hist(bins=30)
        plt.xlabel('Age')
```

```
Out[5]: Text(0.5, 0, 'Age')
```

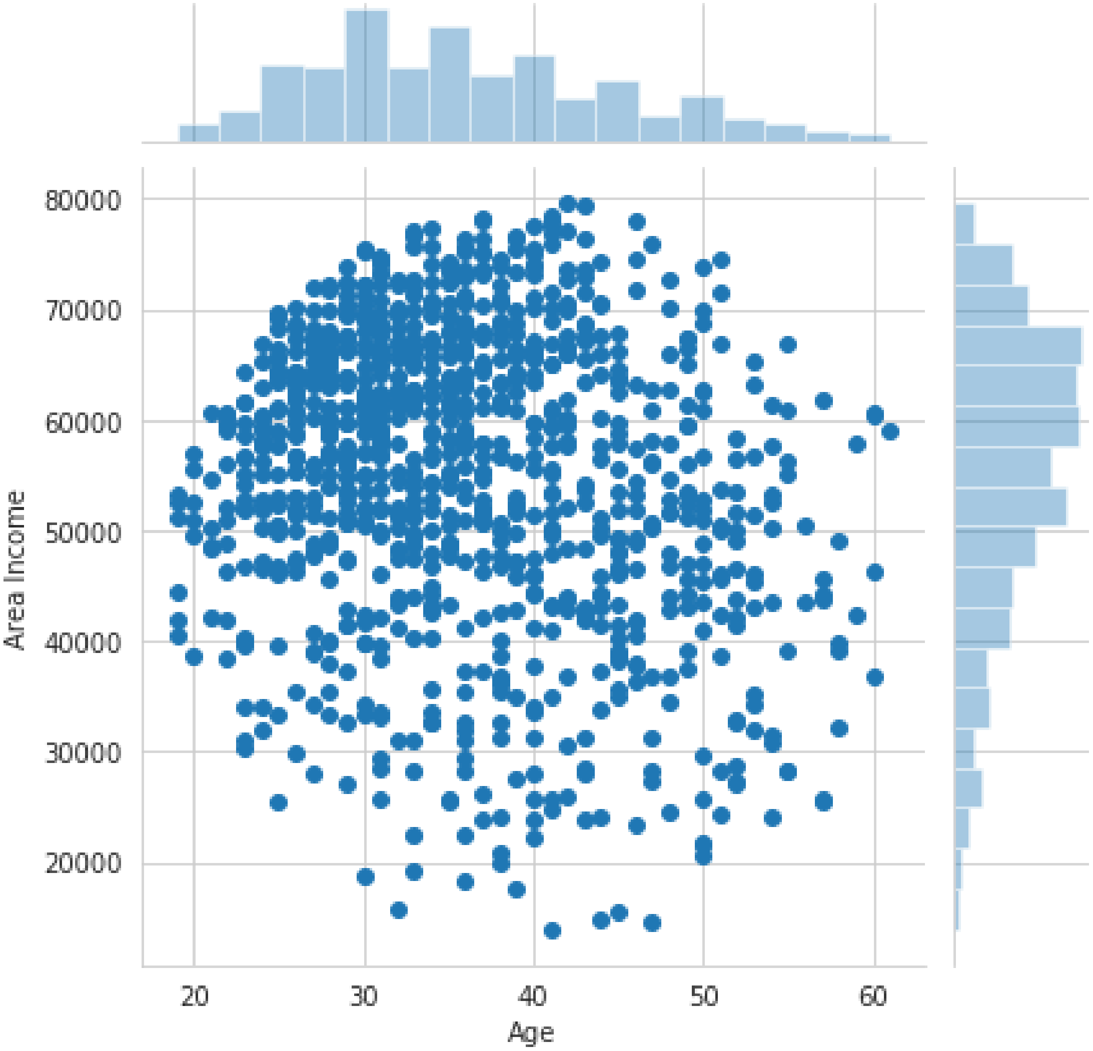


Create a jointplot showing Area Income versus Age.

```
In [6]: sns.jointplot(x='Age',y='Area Income',data=ad_data)
```

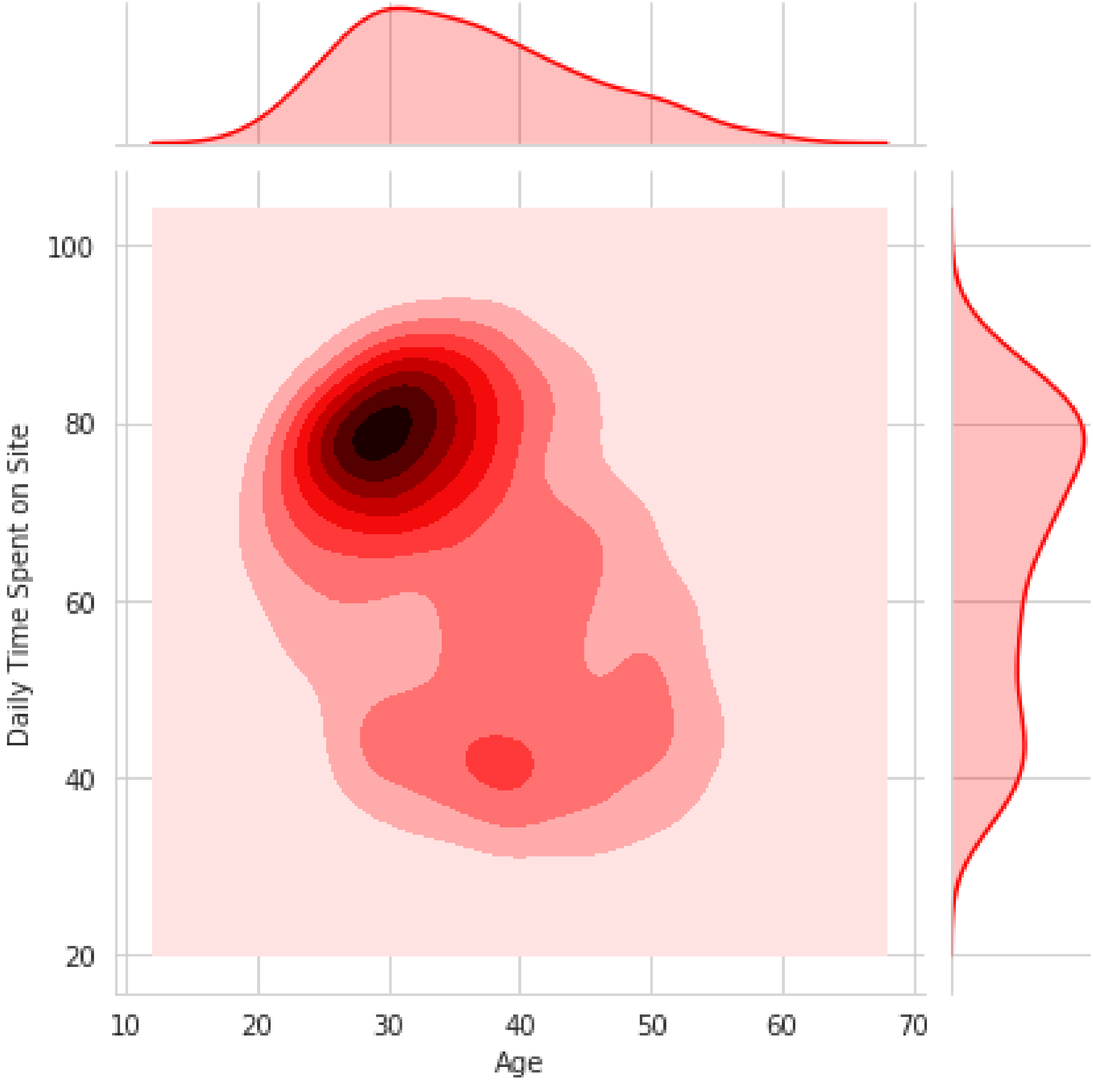
```
/home/kamil/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:1713: FutureWarning: Usi
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

```
Out[6]: <seaborn.axisgrid.JointGrid at 0x7ff04c566f28>
```



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

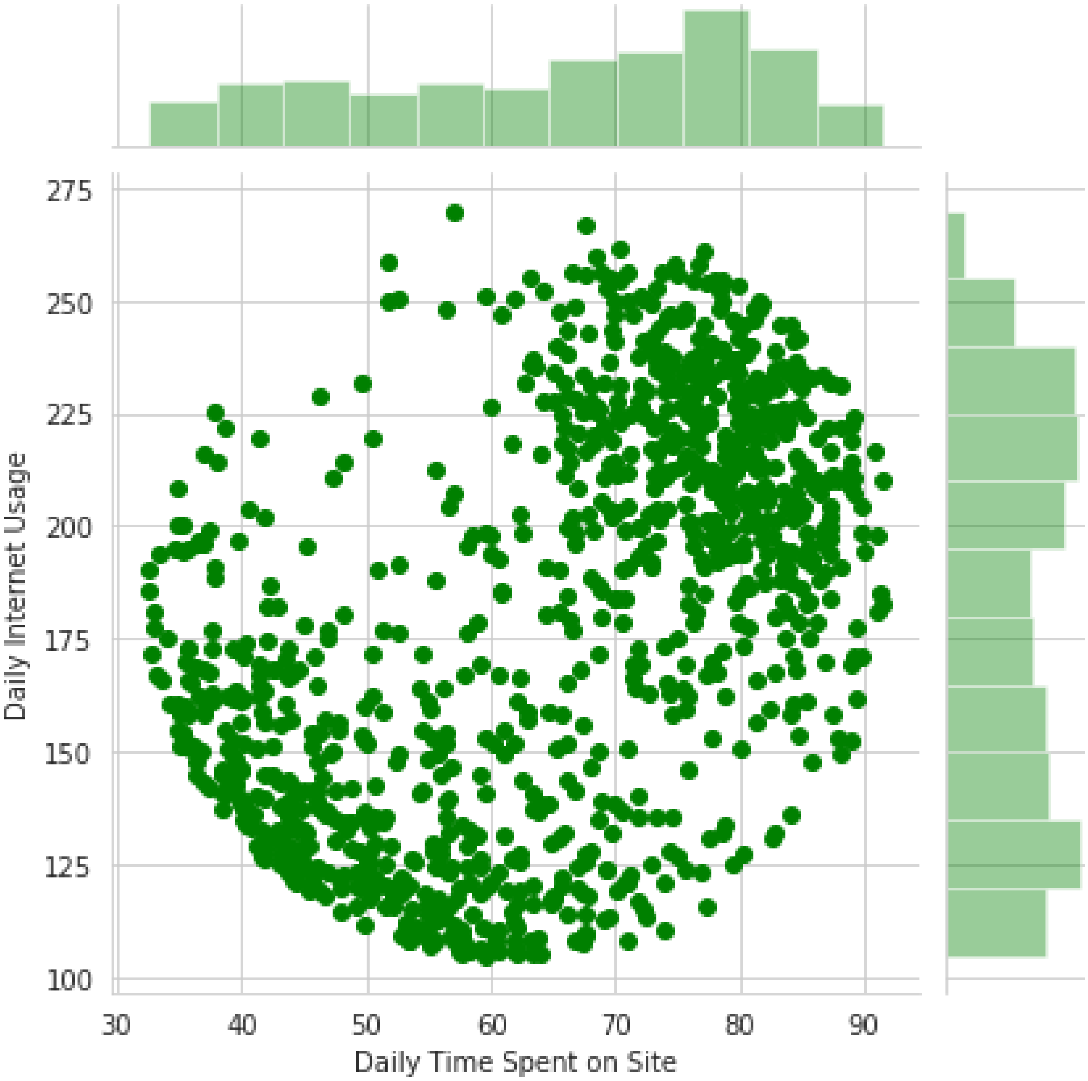
```
In [7]: sns.jointplot(x='Age',y='Daily Time Spent on Site',data=ad_data,color='red',kind='kde'
```



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

```
In [8]: sns.jointplot(x='Daily Time Spent on Site',y='Daily Internet Usage',data=ad_data,color=
```

```
Out[8]: <seaborn.axisgrid.JointGrid at 0x7ff01b3bb198>
```

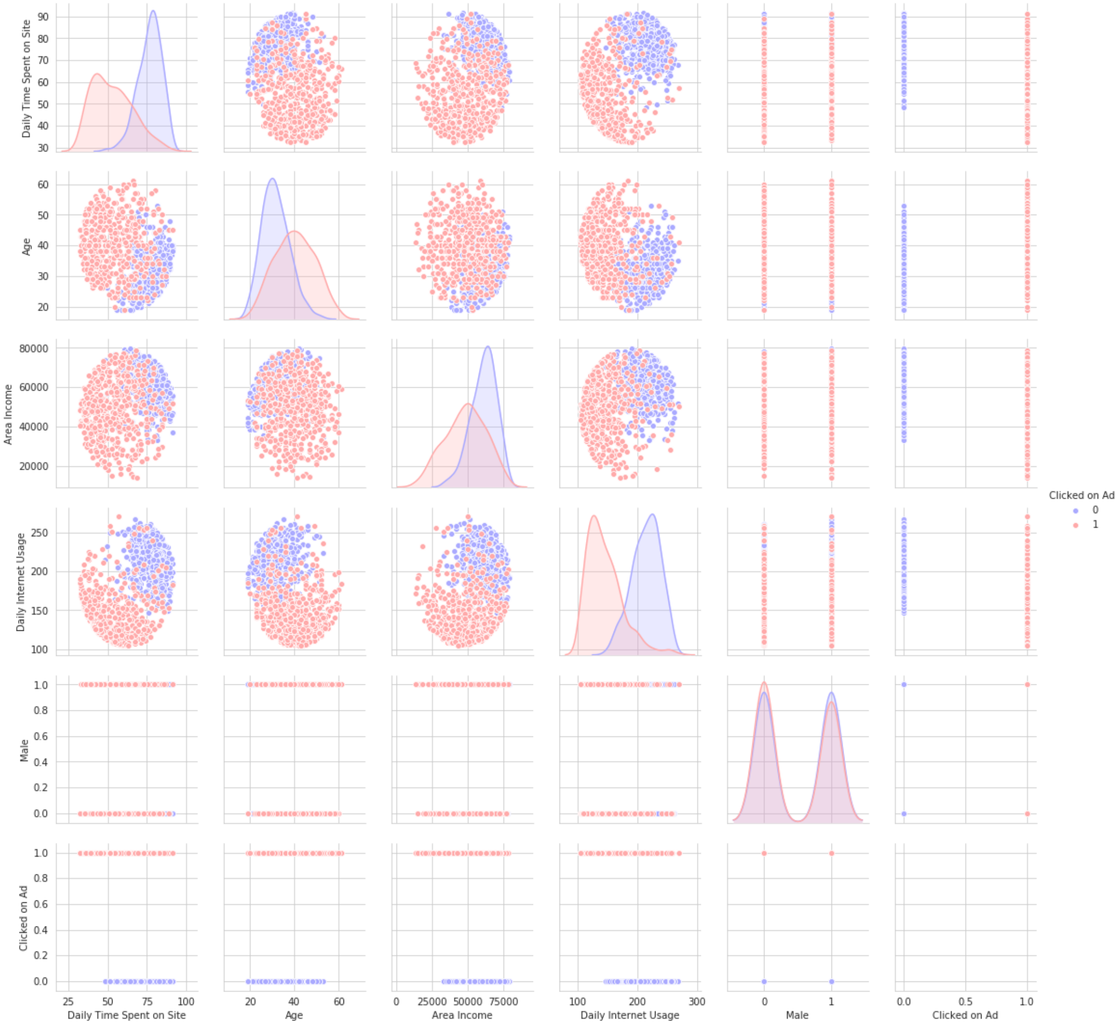


Finally, create a pairplot with the hue defined by the 'Clicked on Ad' column feature.

```
In [9]: sns.pairplot(ad_data,hue='Clicked on Ad',palette='bwr')

/home/kamil/anaconda3/lib/python3.7/site-packages/statsmodels/nonparametric/kde.py:488: Runtime
  binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
/home/kamil/anaconda3/lib/python3.7/site-packages/statsmodels/nonparametric/kdetools.py:34: Run
  FAC1 = 2*(np.pi*bw/RANGE)**2
/home/kamil/anaconda3/lib/python3.7/site-packages/numpy/core/fromnumeric.py:83: RuntimeWarning
  return ufunc.reduce(obj, axis, dtype, out, **passkwargs)


Out[9]: <seaborn.axisgrid.PairGrid at 0x7ff01b2d1c88>
```



3 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

```
In [10]: from sklearn.model_selection import train_test_split
```

```
In [11]: ad_data.head()
```

```
Out[11]:    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
```

```
In [12]: X = ad_data[['Daily Time Spent on Site', 'Age', 'Area Income','Daily Internet Usage',
         y = ad_data['Clicked on Ad']
```

```
In [13]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_stat
```

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

```
In [14]: from sklearn.linear_model import LogisticRegression
```

```
In [15]: lm = LogisticRegression()
         lm.fit(X_train,y_train)
```

```
/home/kamil/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:433: Future
  FutureWarning)
```

```
Out[15]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='warn',
                   n_jobs=None, penalty='l2', random_state=None, solver='warn',
                   tol=0.0001, verbose=0, warm_start=False)
```

4 Predictions and Evaluations

Now predict values for the testing data.

```
In [23]: y_pred = lm.predict(X_test)
```

Create a classification report for the model.

```
In [24]: from sklearn.metrics import classification_report
```

```
In [25]: print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.97      0.90       146
           1       0.96      0.82      0.89       154

   micro avg       0.89      0.89      0.89       300
   macro avg       0.90      0.90      0.89       300
weighted avg       0.90      0.89      0.89       300
```

```
In [ ]:
```