Logistic Regression Project

December 9, 2017

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

```
/usr/local/lib/python2.7/dist-packages/pandas/core/computation/__init__.py:18: UserWarning: The
The minimum supported version is 2.4.6

  ver=ver, min_ver=_MIN_NUMEXPR_VERSION), UserWarning)
```

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

In [3]:
```
ad_data.head()
```

Out[3]:

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

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

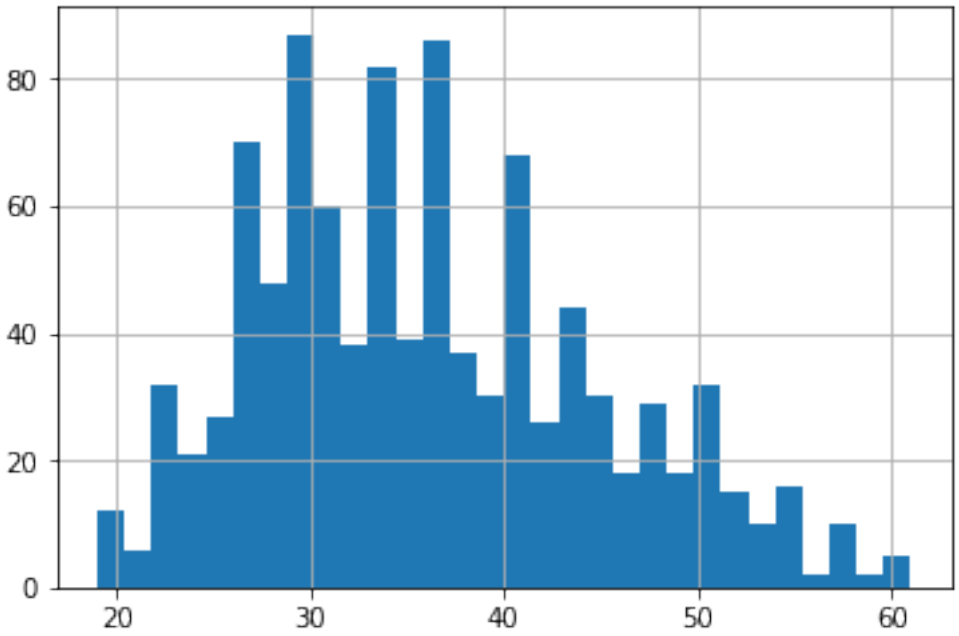
Out[5]:

	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

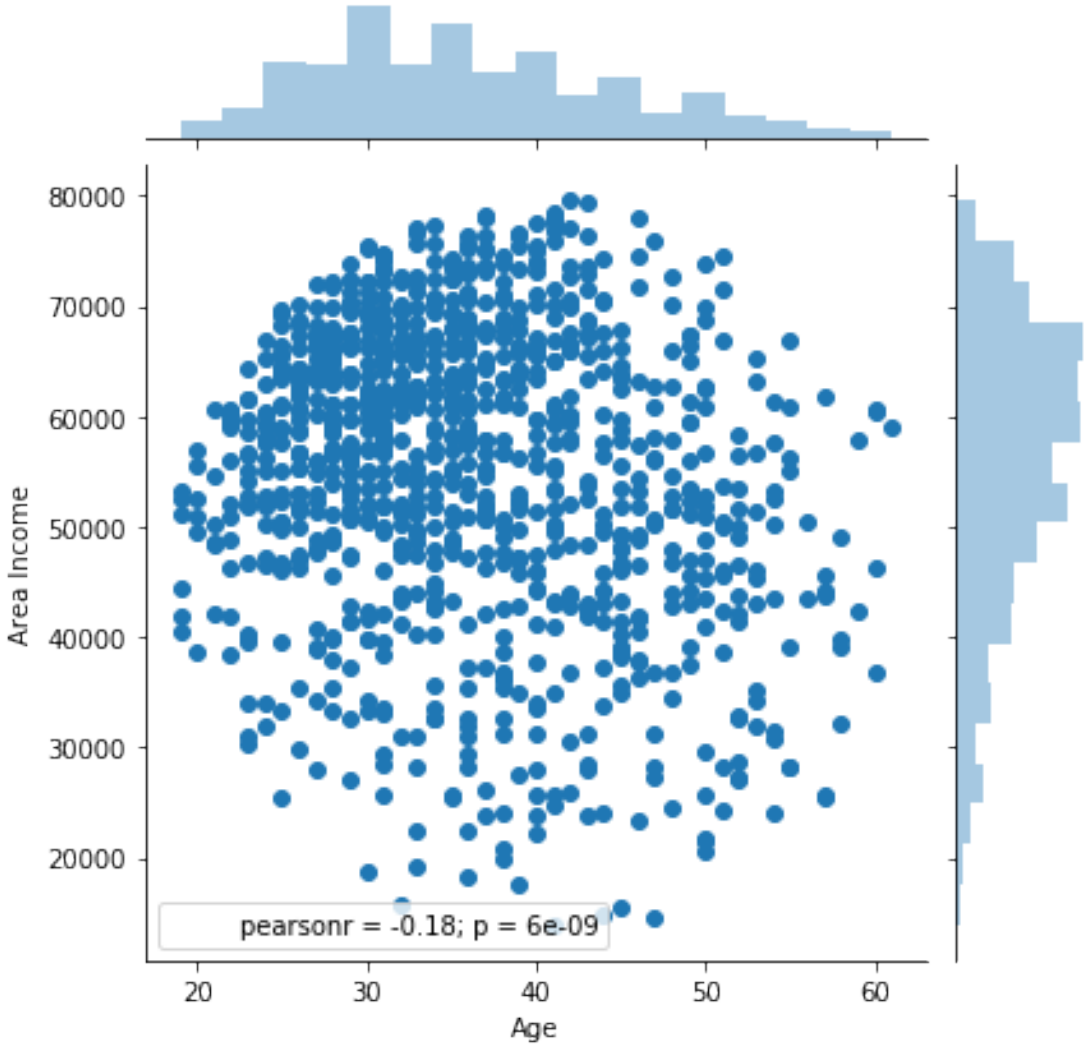
```
In [6]: ad_data['Age'].hist(bins =30)
```

```
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8fb56df410>
```



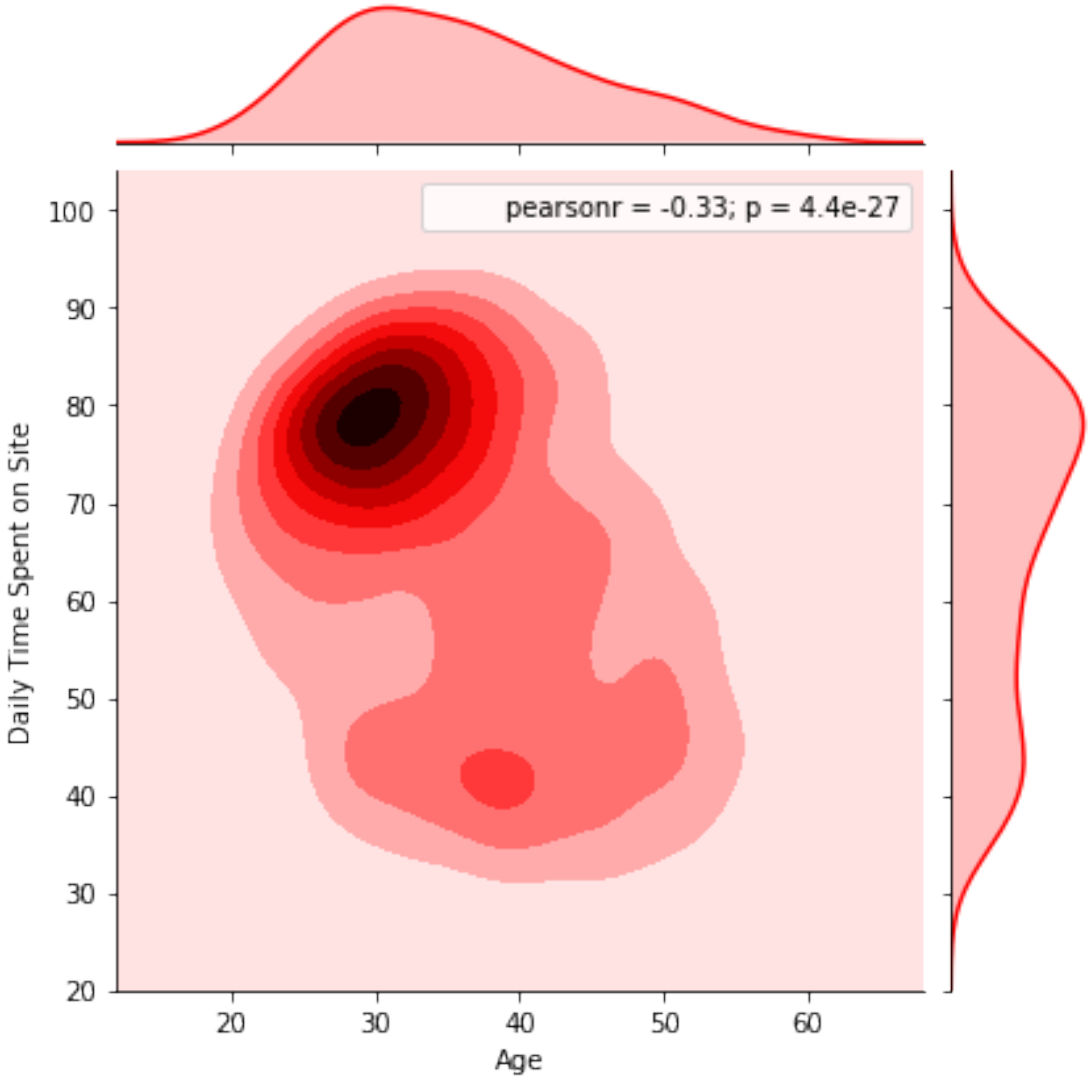

```
In [7]: sns.jointplot('Age', 'Area Income', ad_data)

Out[7]: <seaborn.axisgrid.JointGrid at 0x7f8fb56af210>
```



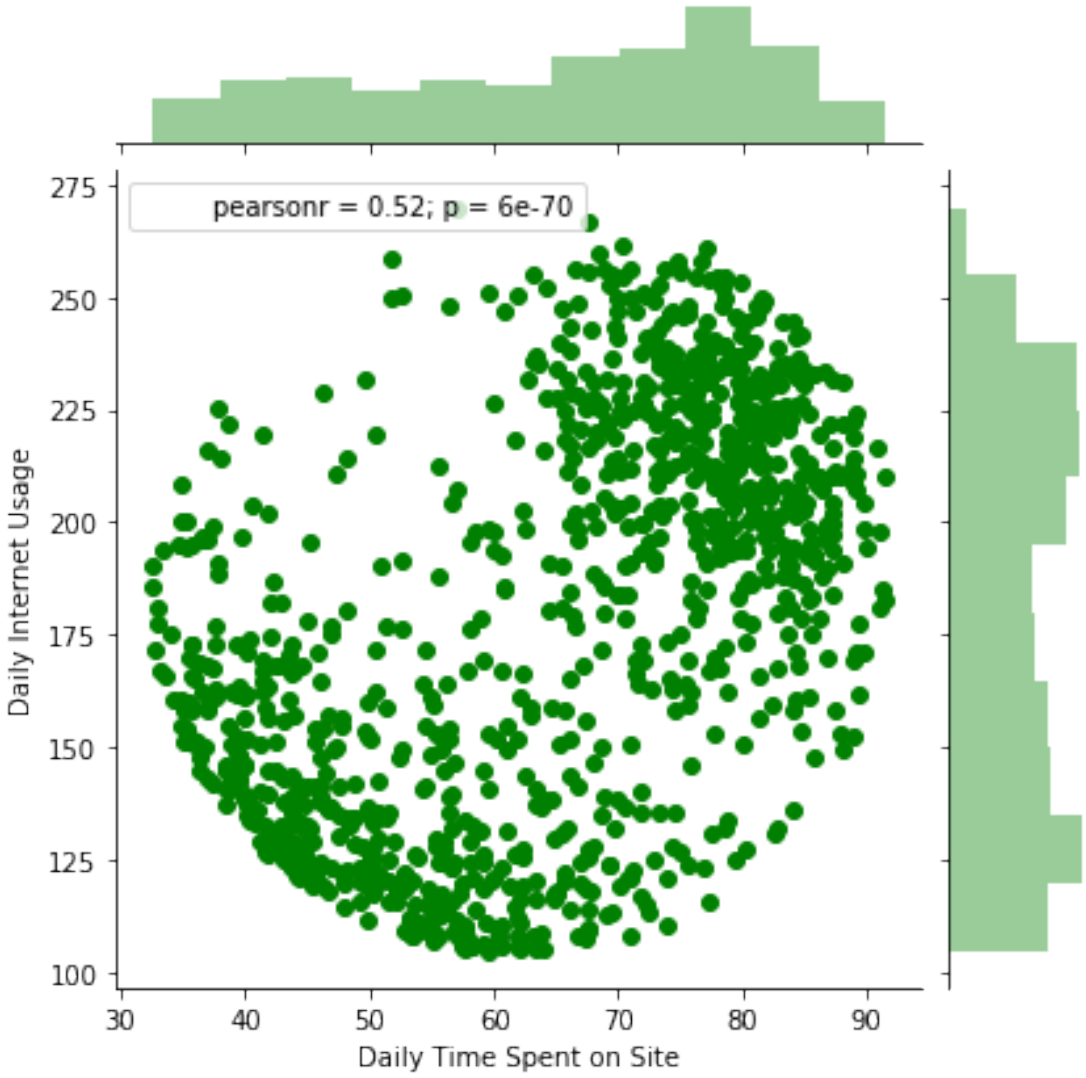

```
In [8]: sns.jointplot('Age', 'Daily Time Spent on Site', ad_data, kind= 'kde', color= 'red')
```

```
Out[8]: <seaborn.axisgrid.JointGrid at 0x7f8fb1cc7110>
```



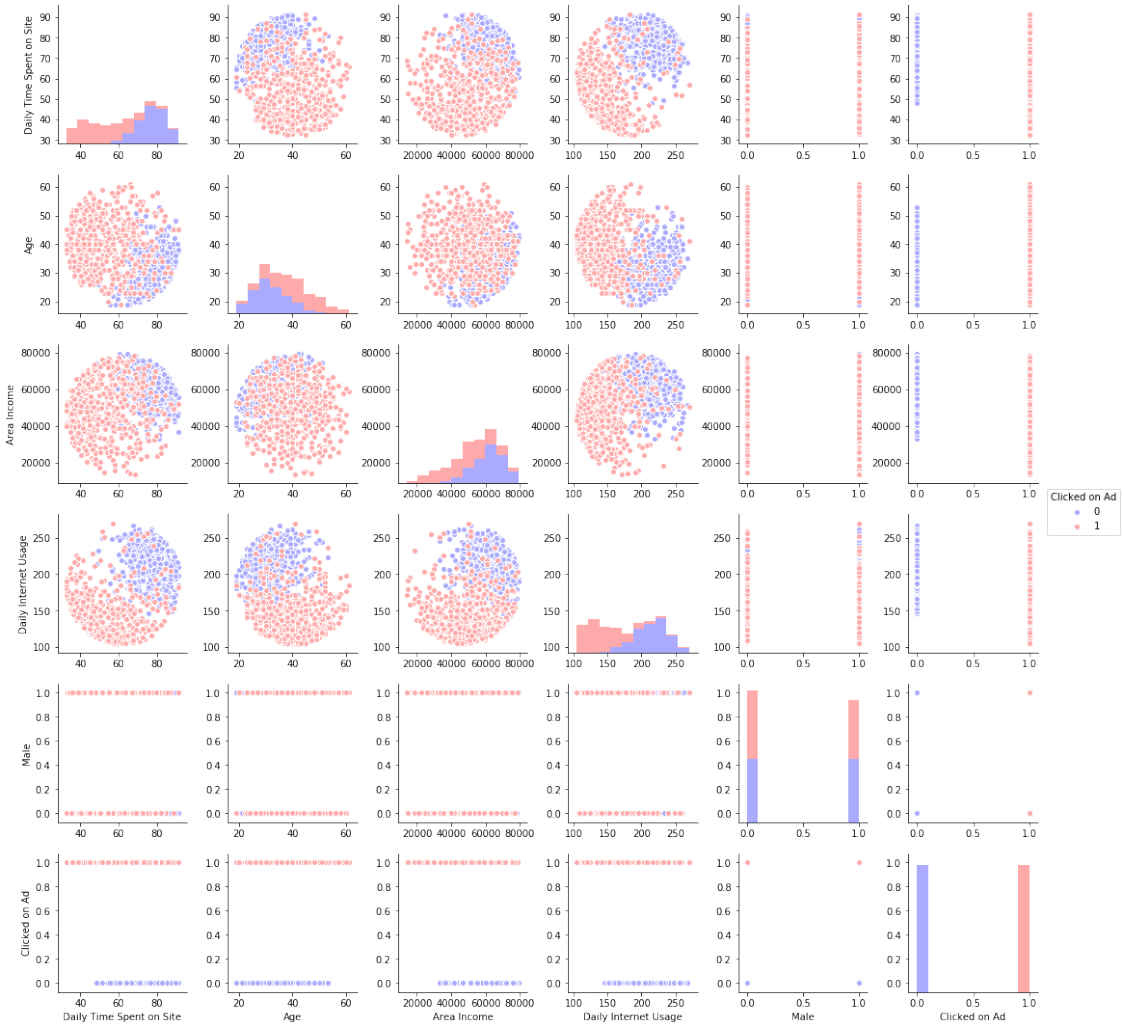

```
In [9]: sns.jointplot(x='Daily Time Spent on Site',y='Daily Internet Usage',data=ad_data,color='
```

```
Out[9]: <seaborn.axisgrid.JointGrid at 0x7f8fb56b8750>
```




```
In [10]: sns.pairplot(ad_data,hue='Clicked on Ad',palette='bwr')
```

```
Out[10]: <seaborn.axisgrid.PairGrid at 0x7f8fb18c8710>
```




```
In [11]: X = ad_data[['Daily Time Spent on Site', 'Age', 'Area Income','Daily Internet Usage', '

In [12]: y= ad_data['Clicked on Ad']

In [13]: 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, random_state=4

In [15]: from sklearn.linear_model import LogisticRegression

In [16]: model= LogisticRegression()
         model.fit(X_train, y_train)

Out[16]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                   penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
                   verbose=0, warm_start=False)
```

```
In [17]: predictions= model.predict(X_test)

In [18]: from sklearn.metrics import classification_report

In [19]: print(classification_report(y_test,predictions))

             precision    recall  f1-score   support

          0       0.84      0.97      0.90       146
          1       0.96      0.82      0.89       154

avg / total       0.90      0.89      0.89       300
```