# 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

In [19]:
```
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
```

In [3]:
```
adv= pd.read_csv('advertising.csv')
```

**Check the head of ad_data**

In [4]:
```
adv.head()
```

Out[4]:

| | 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 |

** Use info and describe() on ad_data**

In [5]:
```
adv.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 [6]:
```
adv.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 |

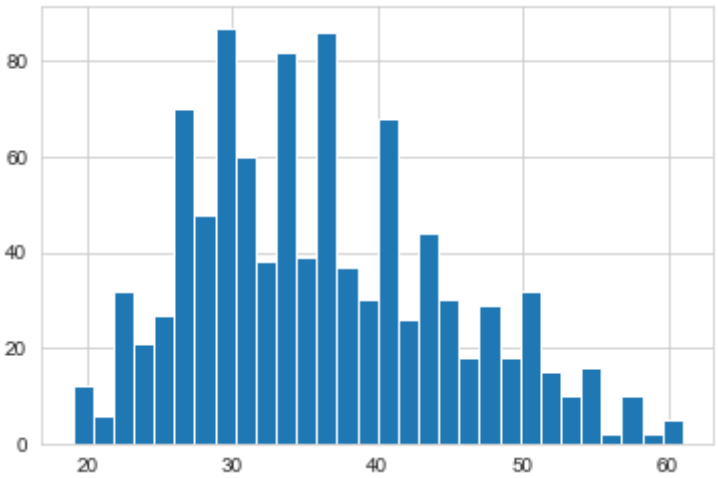
# Exploratory Data Analysis

Let's use seaborn to explore the data!

### histogram of the Age

In [12]:
```
sb.set_style('whitegrid')
plt.hist(adv['Age'], bins=30)
```

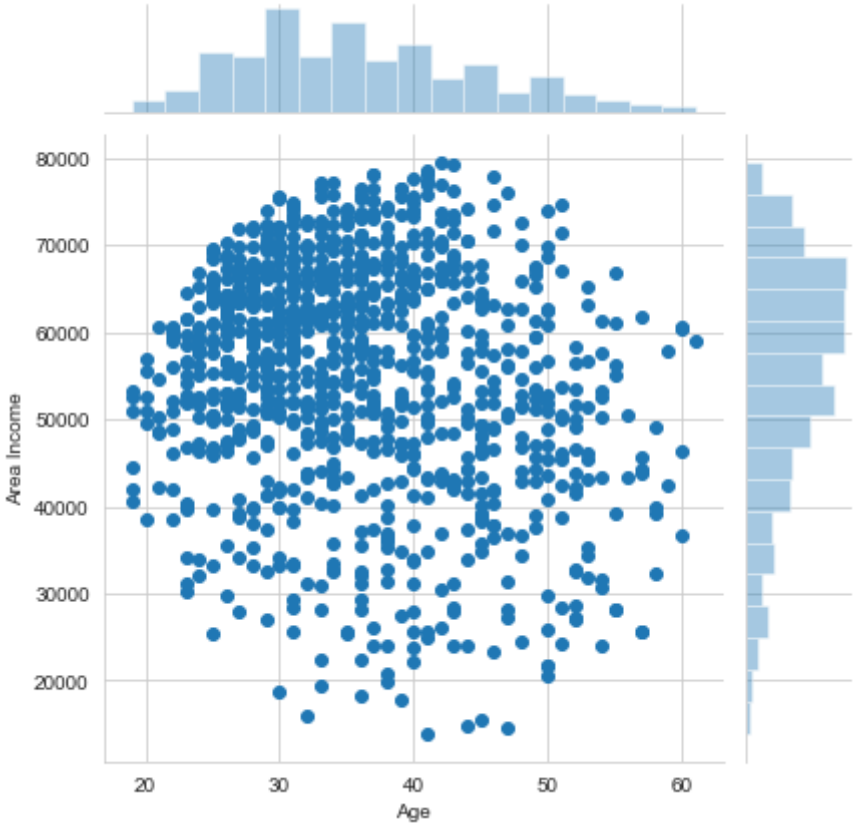
Out[12]:
```
(array([12.,  6., 32., 21., 27., 70., 48., 87., 60., 38., 82., 39., 86.,
        37., 30., 68., 26., 44., 30., 18., 29., 18., 32., 15., 10., 16.,
         2., 10.,  2.,  5.]),
 array([19. , 20.4, 21.8, 23.2, 24.6, 26. , 27.4, 28.8, 30.2, 31.6, 33. ,
        34.4, 35.8, 37.2, 38.6, 40. , 41.4, 42.8, 44.2, 45.6, 47. , 48.4,
        49.8, 51.2, 52.6, 54. , 55.4, 56.8, 58.2, 59.6, 61. ]),
 <a list of 30 Patch objects>)
```




### Create a jointplot showing Area Income versus Age.

In [13]:
```
sb.jointplot('Age','Area Income',data=adv)
```

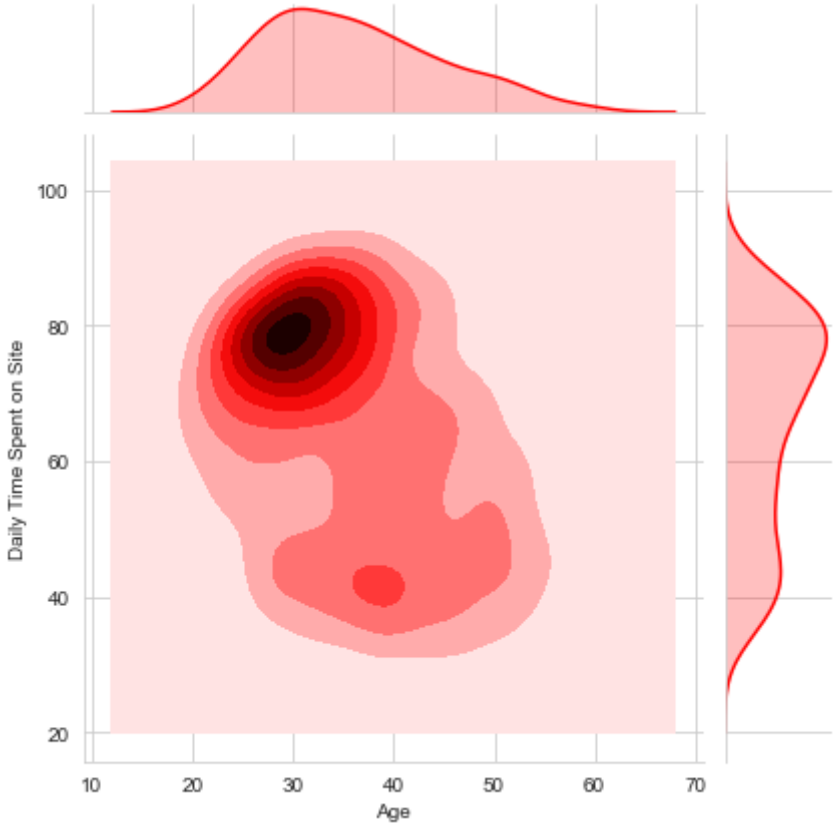
Out[13]:
```
<seaborn.axisgrid.JointGrid at 0x12db06c88>
```




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

In [15]:
```
sb.jointplot('Age','Daily Time Spent on Site',data=adv,kind='kde',color='re
```

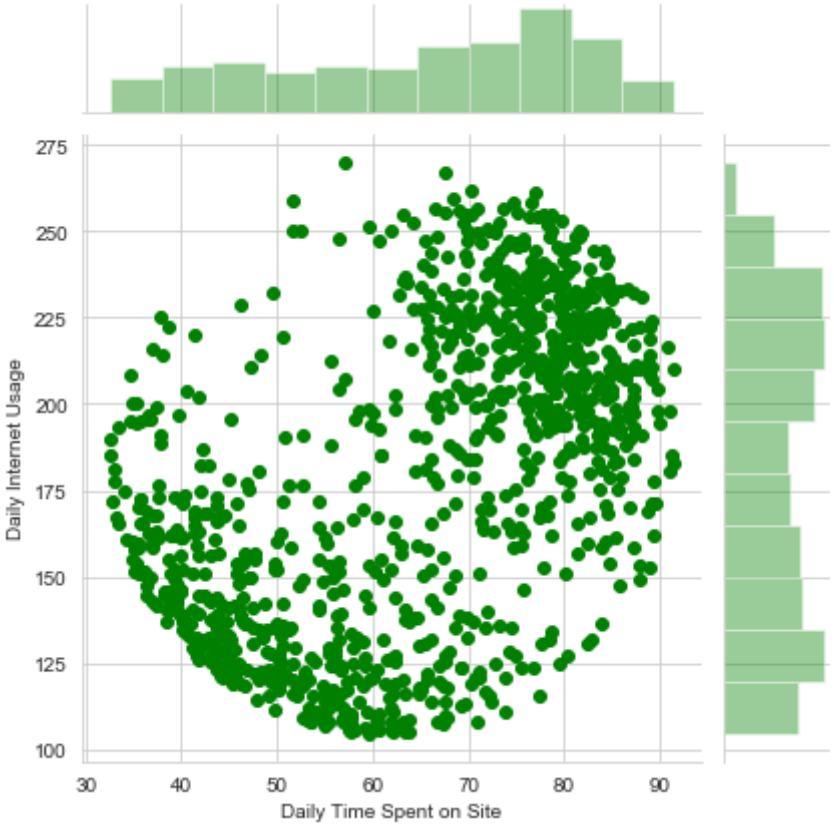
Out[15]:
```
<seaborn.axisgrid.JointGrid at 0x130058898>
```




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

In [16]:
```
sb.jointplot('Daily Time Spent on Site','Daily Internet Usage',data=adv,col
```

Out[16]:
```
<seaborn.axisgrid.JointGrid at 0x1300e8f60>
```

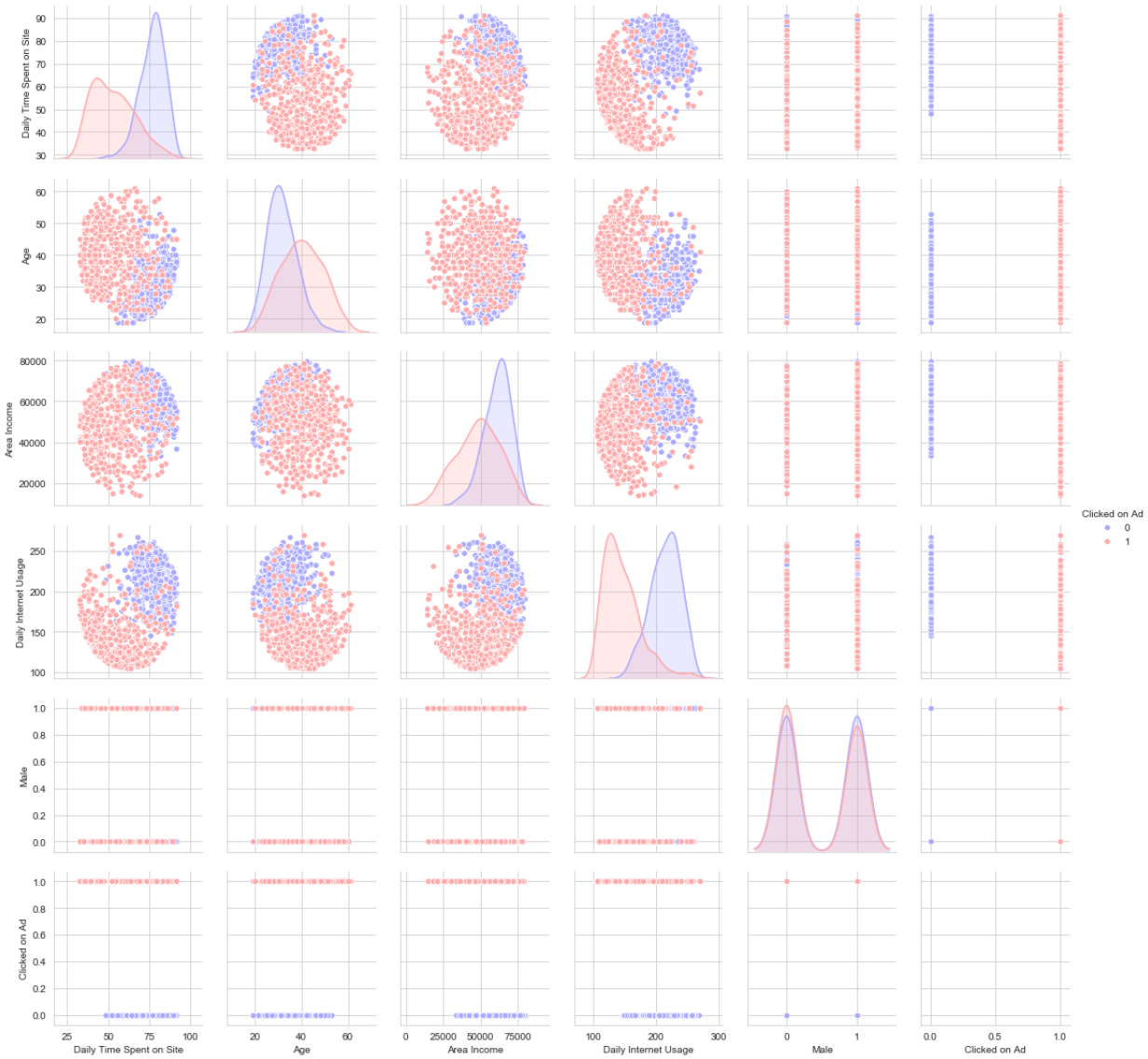



** Finally, create a pairplot with the hue defined by the 'Clicked on Ad' column feature.**

In [18]:
```
sb.pairplot(adv,hue='Clicked on Ad',palette='bwr')
```

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/statsmodels/nonparametric/kde.py:488: RuntimeWarning: invalid value encountered in true_divide
  binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/statsmodels/nonparametric/kdetools.py:34: RuntimeWarning: invalid value encountered in double_scalars
  FAC1 = 2*(np.pi*bw/RANGE)**2
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/numpy/core/fromnumeric.py:83: RuntimeWarning: invalid value encountered in reduce
  return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
```

Out[18]: <seaborn.axisgrid.PairGrid at 0x131b11550>




# 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!

In [21]:
```
X = adv[['Daily Time Spent on Site', 'Age', 'Area Income','Daily Internet U
y = adv['Clicked on Ad']
```

In [23]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, r
```

In [24]:
```
from sklearn.linear_model import LogisticRegression
```

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

In [25]:
```
logm= LogisticRegression()
```

In [26]:
```
logm.fit(X_train,y_train)
```

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-pack
ages/sklearn/linear_model/logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warn
ing.
  FutureWarning)
```

Out[26]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tr
ue,
          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)
```

# Predictions and Evaluations

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

In [27]:
```
predict=logm.predict(X_test)
```

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

In [28]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

In [30]:
```
print(classification_report(y_test,predict))
```

```
              precision    recall  f1-score   support

           0       0.87      0.96      0.91       162
           1       0.96      0.86      0.91       168

   micro avg       0.91      0.91      0.91       330
   macro avg       0.91      0.91      0.91       330
weighted avg       0.91      0.91      0.91       330
```

In [33]:
```
tn, fp, fn, tp=confusion_matrix(y_test,predict).ravel()
```

In [43]:
```
cm=pd.DataFrame(np.asarray((tp,fp,tn,fn)).reshape(2,2),index=['True','False
```

In [44]:
```
cm
```

Out[44]:

| | Positive | Negative |
|---|---|---|
| **True** | 144 | 6 |
| **False** | 156 | 24 |

In [47]:
```
sb.heatmap(cm,cmap='viridis',annot=True)
```

Out[47]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x136926cf8>
```