In [1]:
```
#In this project I will be working with a advertising data set,
#indicating whether or not a particular internet user clicked on an Advertisement.
#I 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.
```

In [9]:
```
import numpy as np
import pandas as pd
```

In [7]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [8]:
```
%matplotlib inline
```

In [10]:
```
ad_data=pd.read_csv('advertising.csv')
```

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

Out[11]:

	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 [12]:
```
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
```

In [13]:
```
ad_data.describe()
```

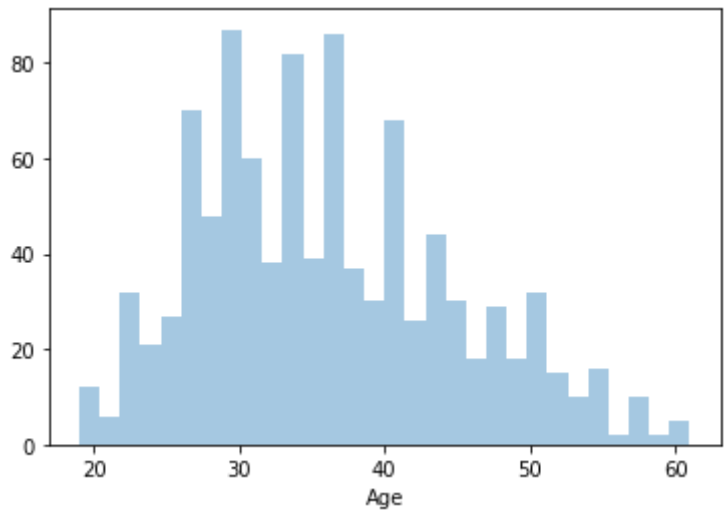
Out[13]:

	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

In [14]:
```
#Let's do some exploratory data analysis
```

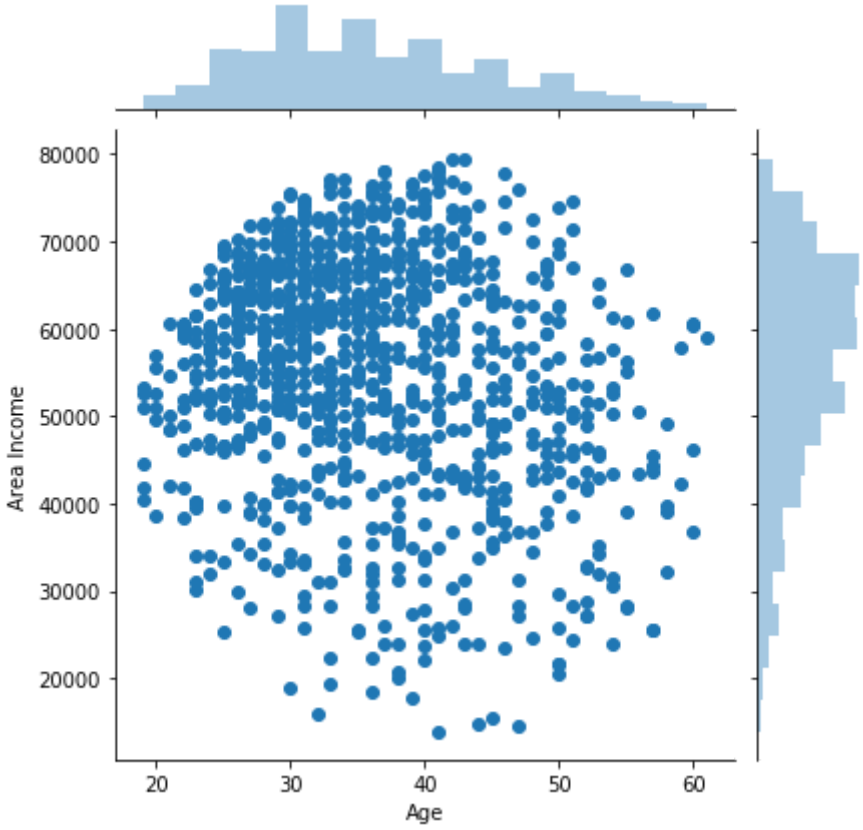
In [16]:
```
sns.distplot(ad_data['Age'],bins=30,kde=False)
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x1cff9885988>



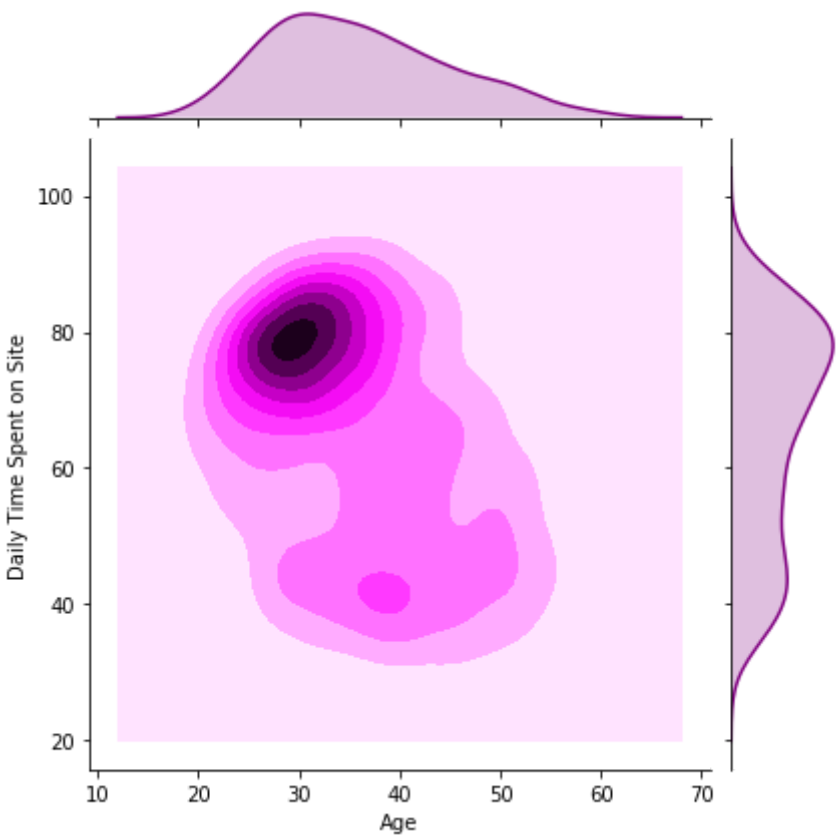
In [17]:
```
sns.jointplot(x='Age',y='Area Income',data=ad_data)
```

Out[17]: <seaborn.axisgrid.JointGrid at 0x1cff9cbf308>



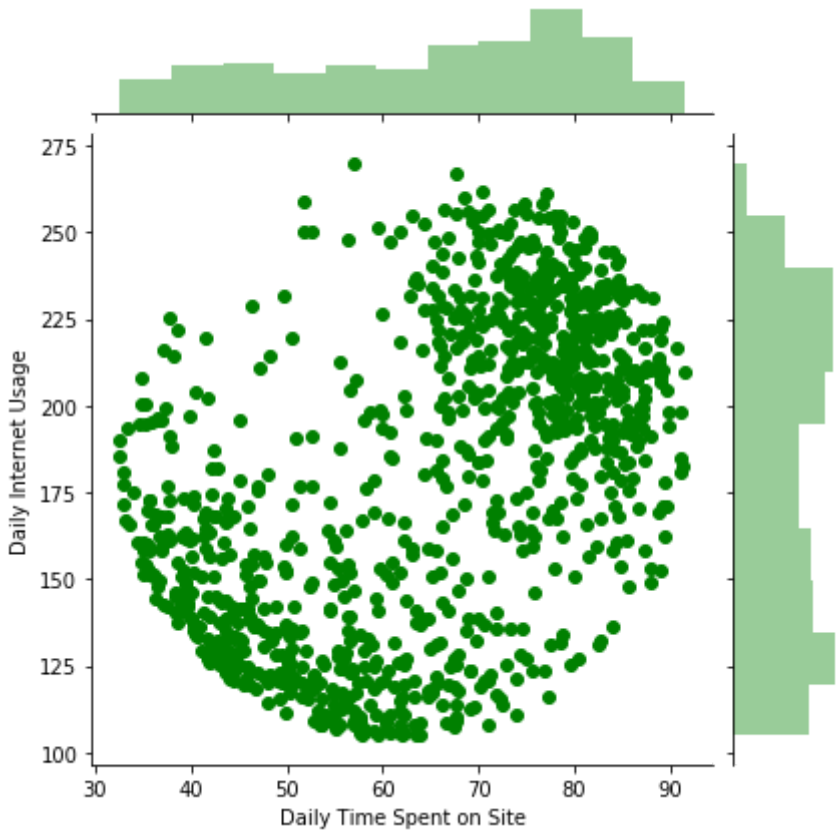
In [20]:
```
#Jointplot showing the kde distributions of Daily Time spent on site vs. Age.
sns.jointplot(x='Age',y='Daily Time Spent on Site',data=ad_data,kind='kde',color='Purple')
```

Out[20]: <seaborn.axisgrid.JointGrid at 0x1cffb46bc08>



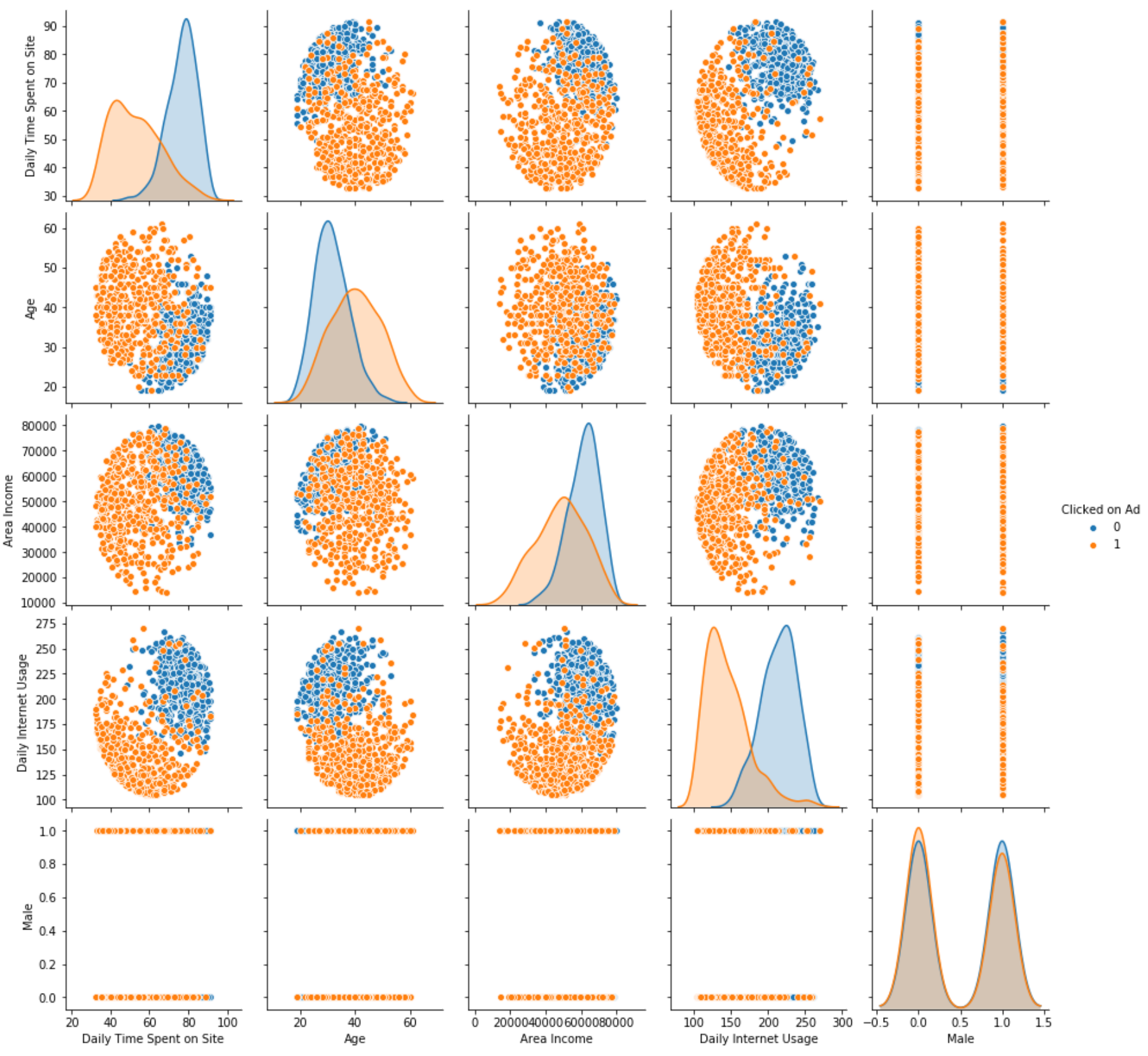
In [22]:
```
sns.jointplot(x='Daily Time Spent on Site',y='Daily Internet Usage',data=ad_data,color='Green')
```

Out[22]: <seaborn.axisgrid.JointGrid at 0x1cffb697148>



In [24]:
```
#Seems like there are two clusters so we can identify couple relationships.
#We can do this for all kinds of numerical data present in the dataset therefore:
sns.pairplot(data=ad_data,hue='Clicked on Ad')
```

Out[24]: <seaborn.axisgrid.PairGrid at 0x1cffc81e088>



In [28]:
```
#Moving on to the logistic regression
from sklearn.model_selection import train_test_split
ad.data.column()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-28-211b3eae342b> in <module>
      1 #Moving on to the logistic regression
      2 from sklearn.model_selection import train_test_split
----> 3 ad.data.column()

NameError: name 'ad' is not defined
```

In [29]:
```
X=ad_data[['Daily Time Spent on Site','Age','Area Income','Daily Internet Usage','Male']]
y=ad_data['Clicked on Ad']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=101)
```

In [30]:
```
from sklearn.linear_model import LogisticRegression
```

In [31]:
```
logmodel=LogisticRegression()
```

In [32]:
```
logmodel.fit(X_train,y_train)
```

```
C:\Users\anike\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:940: ConvergenceWarning: lbfgs failed to
converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[32]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [34]:
```
predictions=logmodel.predict(X_test)
```

In [35]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

In [37]:
```
print(classification_report(y_test,predictions))
print((confusion_matrix(y_test,predictions)))
```

```
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       171
           1       0.99      0.96      0.98       159

    accuracy                           0.98       330
   macro avg       0.98      0.98      0.98       330
weighted avg       0.98      0.98      0.98       330

[[170   1]
 [  6 153]]
```

In [ ]:
```
#Seems like a pretty good model
```