Linear Regression - Project

This data is based on an Ecommerce company located in New York City that sells clothing online but it also has in-store style and clothing advice sessions. Customers come in to the store, have sessions/meetings with a personal stylist, then they can go home and order either on a mobile app or website the clothes they want.

The company is trying to decide whether to focus their efforts on their mobile app experience or their website. This is what we will try and figure out now.

Imports

In [1]:

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

Get the Data

We'll work with the Ecommerce Customers csv file from the company. It has Customer info, suchas Email, Address, and their color Avatar. Then it also has numerical value columns:

- Avg. Session Length: Average session of in-store style advice sessions.
- Time on App: Average time spent on App in minutes
- Time on Website: Average time spent on Website in minutes
- Length of Membership: How many years the customer has been a member.

Read in the Ecommerce Customers csv file as a DataFrame called customers.

In [3]:

```
customers = pd.read_csv('Ecommerce Customers')
```

Check the head of customers, and check out its info() and describe() methods.

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

Out[4]:

	Email	Address	Avatar	Avg. Session Length	Time on App	
0	mstephenson@fernandez.com	835 Frank Tunnel\nWrightmouth, MI 82180-9605	Violet	34.497268	12.655651	3
1	hduke@hotmail.com	4547 Archer Common\nDiazchester, CA 06566-8576	DarkGreen	31.926272	11.109461	3
2	pallen@yahoo.com	24645 Valerie Unions Suite 582\nCobbborough, D...	Bisque	33.000915	11.330278	3
3	riverarebecca@gmail.com	1414 David Throughway\nPort Jason, OH 22070-1220	SaddleBrown	34.305557	13.717514	3
4	mstephens@davidson-herman.com	14023 Rodriguez Passage\nPort Jacobville, PR 3...	MediumAquaMarine	33.330673	12.795189	3

In [5]:
```
customers.describe()
```

Out[5]:

	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
count	500.000000	500.000000	500.000000	500.000000	500.000000
mean	33.053194	12.052488	37.060445	3.533462	499.314038
std	0.992563	0.994216	1.010489	0.999278	79.314782
min	29.532429	8.508152	33.913847	0.269901	256.670582
25%	32.341822	11.388153	36.349257	2.930450	445.038277
50%	33.082008	11.983231	37.069367	3.533975	498.887875
75%	33.711985	12.753850	37.716432	4.126502	549.313828
max	36.139662	15.126994	40.005182	6.922689	765.518462

```
In [6]: customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
Email                   500 non-null object
Address                 500 non-null object
Avatar                  500 non-null object
Avg. Session Length     500 non-null float64
Time on App             500 non-null float64
Time on Website         500 non-null float64
Length of Membership    500 non-null float64
Yearly Amount Spent     500 non-null float64
dtypes: float64(5), object(3)
memory usage: 31.3+ KB
```

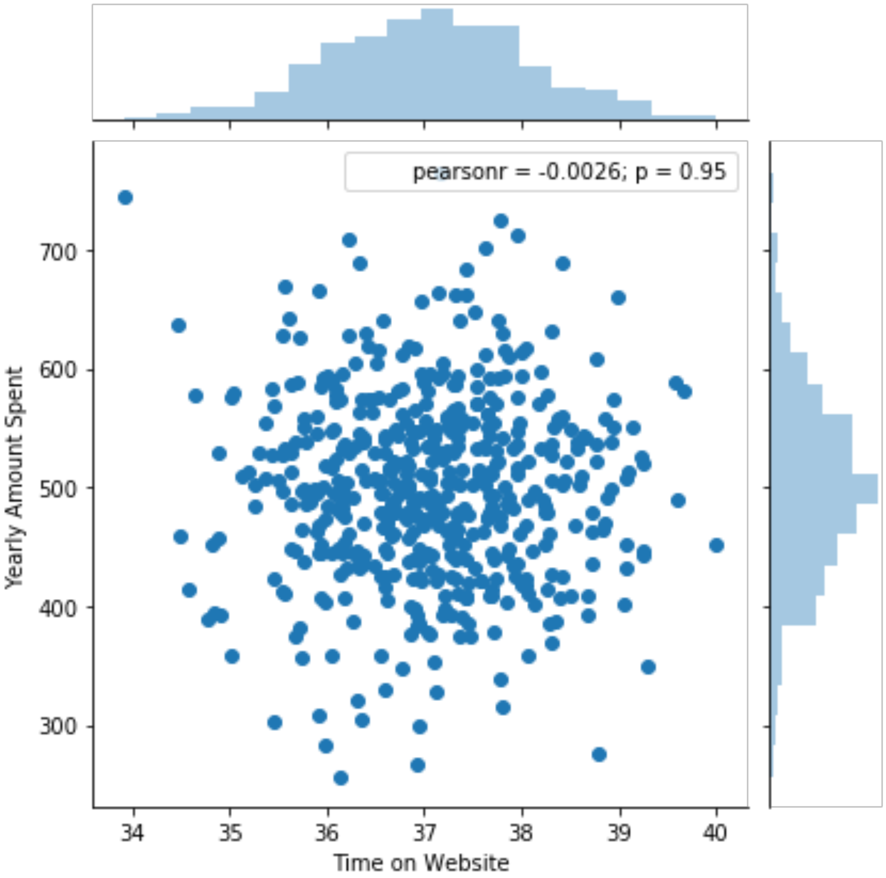
Exploratory Data Analysis

Create a joint plot of 'Yearly Amount Spent' versus 'Time on Website'.

```
In [7]: sns.jointplot(x='Time on Website', y = 'Yearly Amount Spent',
                      data = customers)
```

```
/Users/Jayashri/anaconda/lib/python3.6/site-packages/scipy/stats/stats
.py:1713: FutureWarning: Using a non-tuple sequence for multidimension
al indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`
. In the future this will be interpreted as an array index, `arr[np.ar
ray(seq)]`, which will result either in an error or a different result
.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

```
Out[7]: <seaborn.axisgrid.JointGrid at 0x1a24130588>
```

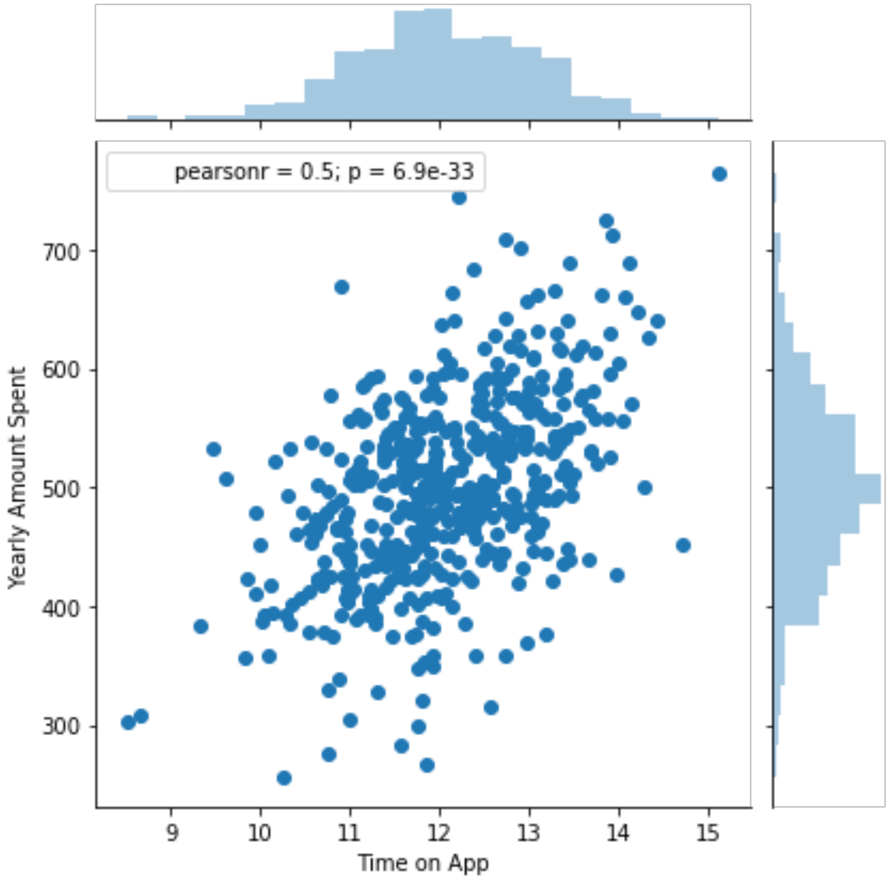


Create a joint plot of 'Yearly Amount Spent' versus 'Time on App' column.

```
In [8]: sns.jointplot(x='Time on App', y = 'Yearly Amount Spent',
                      data = customers)
```

```
/Users/Jayashri/anaconda/lib/python3.6/site-packages/scipy/stats/stats
.py:1713: FutureWarning: Using a non-tuple sequence for multidimension
al indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`
. In the future this will be interpreted as an array index, `arr[np.ar
ray(seq)]`, which will result either in an error or a different result
.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

```
Out[8]: <seaborn.axisgrid.JointGrid at 0x1a24675ac8>
```



Use jointplot to create a 2D hex bin plot comparing Time on App and Length of Membership.

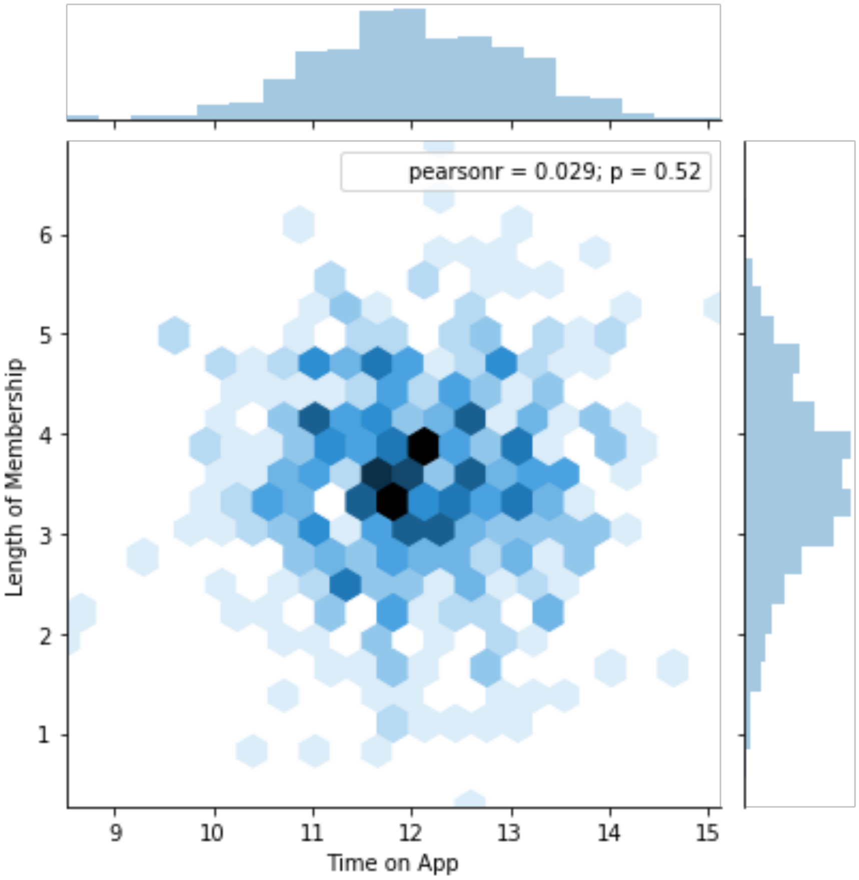
In [9]:

```
sns.jointplot(kind='hex', x = 'Time on App', y = 'Length of Membership',
              data = customers)
```

```
/Users/Jayashri/anaconda/lib/python3.6/site-packages/scipy/stats/stats
.py:1713: FutureWarning: Using a non-tuple sequence for multidimension
al indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`
. In the future this will be interpreted as an array index, `arr[np.ar
ray(seq)]`, which will result either in an error or a different result
.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[9]:

```
<seaborn.axisgrid.JointGrid at 0x1a247d2f60>
```



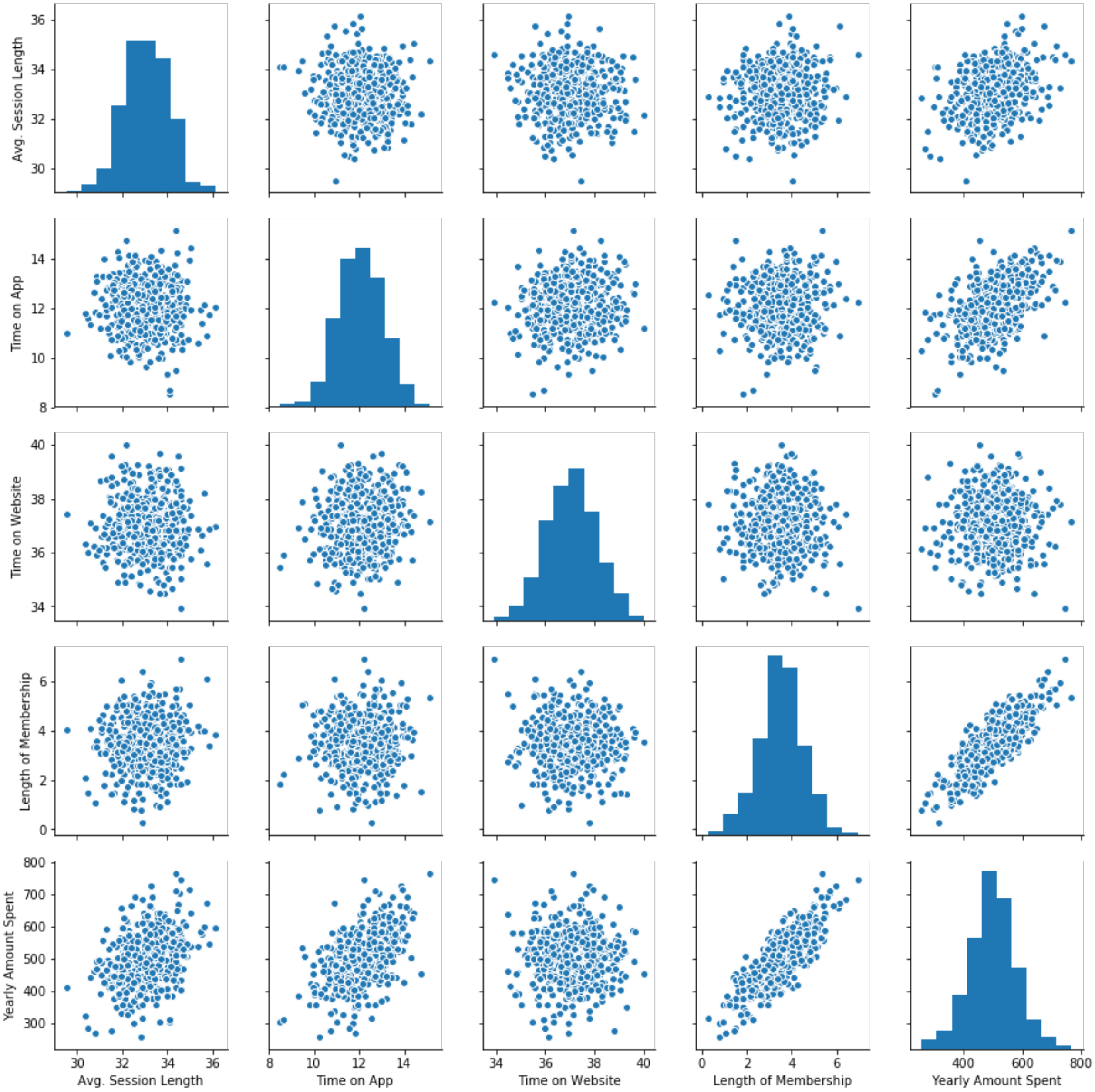
Let's explore these types of relationships across the entire data set using pairplot. Reference: https://stanford.edu/~mwaskom/software/seaborn/tutorial/axis_grids.html#plotting-pairwise-relationships-with-pairgrid-and-pairplot (https://stanford.edu/~mwaskom/software/seaborn/tutorial/axis_grids.html#plotting-pairwise-relationships-with-pairgrid-and-pairplot))

In [12]:

```
sns.pairplot(customers)
```

Out[12]:

```
<seaborn.axisgrid.PairGrid at 0x1a1f69cc88>
```



Observation: Length of Membership is the most correlated with Yearly Amount Spent.

Create a linear model plot (using seaborn's lmplot) of Yearly Amount Spent vs. Length of Membership.

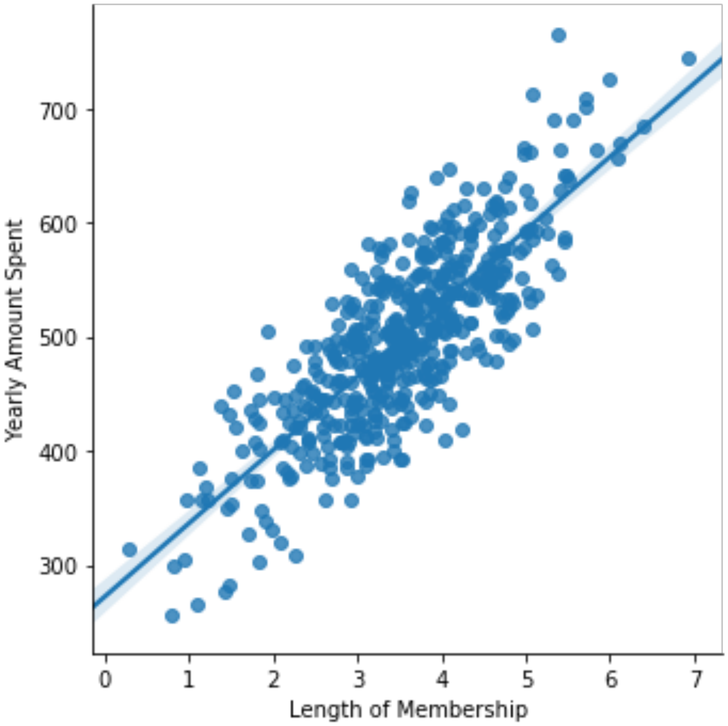
In [10]:

```
sns.lmplot(x = 'Length of Membership', y = 'Yearly Amount Spent',
           data = customers)
```

```
/Users/Jayashri/anaconda/lib/python3.6/site-packages/scipy/stats/stats
.py:1713: FutureWarning: Using a non-tuple sequence for multidimension
al indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`
. In the future this will be interpreted as an array index, `arr[np.ar
ray(seq)]`, which will result either in an error or a different result
.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[10]:

```
<seaborn.axisgrid.FacetGrid at 0x1a24968eb8>
```



Training and Testing Data

Let us split the data into training and testing sets.

Set a variable X equal to the numerical features of the customers and a variable y equal to the "Yearly Amount Spent" column.

In [11]:
```
customers.columns
```

Out[11]:
```
Index(['Email', 'Address', 'Avatar', 'Avg. Session Length', 'Time on A
pp',
       'Time on Website', 'Length of Membership', 'Yearly Amount Spent
'],
      dtype='object')
```

In [13]:
```
X = customers[['Avg. Session Length',
               'Time on App','Time on Website', 'Length of Membership']]
y = customers['Yearly Amount Spent']
```

Import model_selection.train_test_split from sklearn to split the data into training and testing sets. Set test_size=0.3 and random_state=101

In [15]:
```
from sklearn.model_selection import train_test_split
```

In [16]:
```
X_train,X_test,y_train,y_test=train_test_split(X, y,
                                               test_size=0.3,
                                                random_state=101)
```

Training the Model

Now we will train the model on training data

Import LinearRegression from sklearn.linear_model

In [18]:
```
from sklearn.linear_model import LinearRegression
```

Create an instance of a LinearRegression() model named lm.

In [19]:
```
lm = LinearRegression()
```

Train/fit lm on the training data.

In [20]:
```
lm.fit(X_train, y_train)
```

Out[20]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=
False)
```

Print out the coefficients of the model

Print out the coefficients of the model

In [22]:

```
print(lm.coef_)
```

```
[25.98154972 38.59015875  0.19040528 61.27909654]
```

In [23]:

```
print(lm.intercept_)
```

```
-1047.9327822502387
```

In [24]:

```
coeff_df = pd.DataFrame(lm.coef_, X.columns,
                        columns=['Coefficients'])
coeff_df
```

Out[24]:

	Coefficients
Avg. Session Length	25.981550
Time on App	38.590159
Time on Website	0.190405
Length of Membership	61.279097

Evaluating the Model

Let's evaluate our model performance by calculating the residual sum of squares and the explained variance score (R^2).

Calculate the Mean Absolute Error, Mean Squared Error, the Root Mean Squared Error and R-Square for explained variance. Reference: Wikipedia for the formulas

In [34]:

```
from sklearn import metrics
print('MAE: ', metrics.mean_absolute_error(y_test, predictions) )
print('MSE: ', metrics.mean_squared_error(y_test, predictions) )
print('RMSE: ', np.sqrt(metrics.mean_squared_error(y_test,
                                                    predictions)))
print('R-Square:', metrics.explained_variance_score(y_test,
                                                     predictions))
```

```
MAE:  7.228148653430853
MSE:  79.81305165097487
RMSE:  8.933815066978656
R-Square: 0.9890771231889606
```

Observation: The R-Square value looks very good.

Predicting Test Data

Now that we have fitted the model, let us evaluate its performance by predicting off the test values.

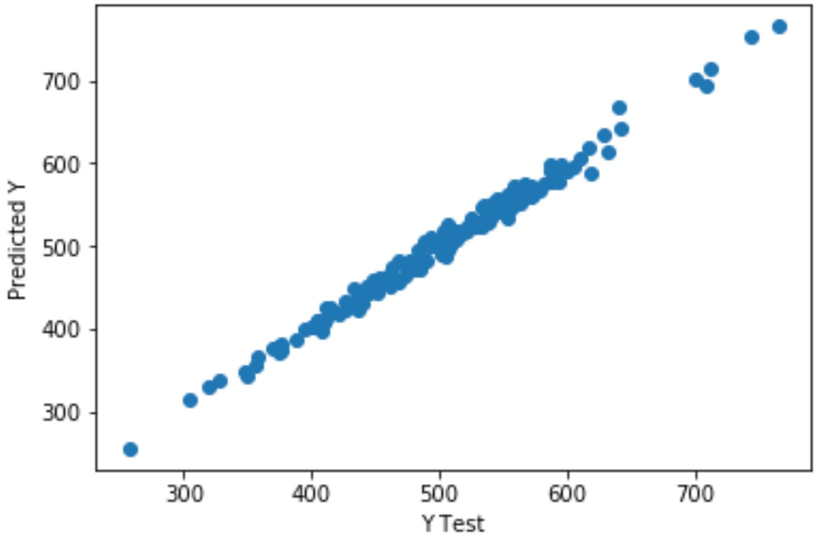
Use lm.predict() to predict off the X_test set of the data.

```
In [35]: predictions = lm.predict(X_test)
```

Create a scatterplot of the real test values versus the predicted values.

```
In [36]:
plt.scatter(y_test, predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

```
Out[36]: <matplotlib.text.Text at 0x1a25ec6a58>
```



Check normal distribution of the residual plot

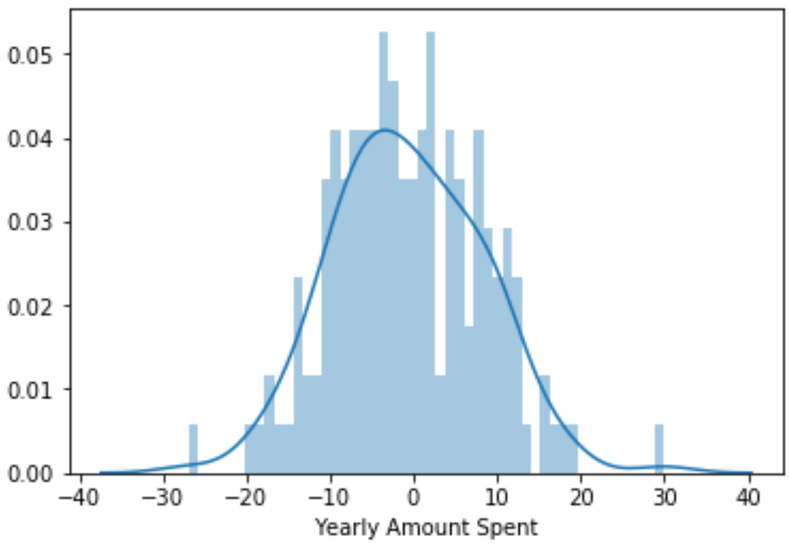
In [37]:

```
sns.distplot((y_test - predictions), bins = 50)
```

```
/Users/Jayashri/anaconda/lib/python3.6/site-packages/scipy/stats/stats
.py:1713: FutureWarning: Using a non-tuple sequence for multidimension
al indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`
. In the future this will be interpreted as an array index, `arr[np.ar
ray(seq)]`, which will result either in an error or a different result
.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[37]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a25f82860>
```



Observation: The above histogram plot of the residuals looks normally distributed with mean = 0.

Predictions to be made

We still have to figure out if we focus our efforst on mobile app or website development.

In [38]:

```
# Set up X and y arrays
X = customers[['Avg. Session Length',
              'Time on App','Time on Website', 'Length of Membership']]
y = customers['Yearly Amount Spent']

lm = LinearRegression()
lm.fit(X, y)
```

Out[38]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=
False)
```

In [39]:

```
# Intercept
print(lm.intercept_)
```

```
-1051.5942553006246
```

In [40]:

```
# The coefficients of X
print(lm.coef_)
```

```
[25.73427108 38.70915381  0.43673884 61.57732375]
```

In [41]:

```
# The Coefficient Matrix
coeff_df = pd.DataFrame(lm.coef_, X.columns,
                               columns=['Coefficient'])
coeff_df
```

Out[41]:

	Coefficient
Avg. Session Length	25.734271
Time on App	38.709154
Time on Website	0.436739
Length of Membership	61.577324

Observation: The company should focus more on Mobile App since it is producing more revenue.