PROJECT REPORT ON ECOMMERCE CUSTOMERS DEVICE USAGE

Y.Murali Krishna

INTRODUCTION:

Ecommerce customer services are very common nowadays. This type of technical device usage is based the technology and how it had been developed and developing in upcoming days.

This is a problem for many companies to analyse the large data and here comes the data scientist who does the work of analysing dataset and data and variables in it.

Importance of project:

This project mainly helps in analysing and classifying the data by testing and training.

This will be done by splitting the data into training data and testing data.

This project will help the business magnets to find data respective of their classification and finding the accuracy in it.

Plotting the graphs will let the analysists to find the data and to analyse the dataset very simply.

Keywords:Linear Regression,Customers,Linear Algeabra,Data Pre-Processing.

```python
# This Python 3 environment comes with many helpful analytics libraries installed # It is defined by the kaggle/python docker image: https://github.com/kaggle/docke r-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
```

```
# For example, running this (by clicking run or pressing Shift+Enter) will list th
e files in the input directory
 import
os
print(os.listdir("../input"))

# Any results you write to the current directory are saved as output. ['Ecommerce
Customers']
```

In [2]:

```
linkcode
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, such as 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.**

linkcode

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

```
= pd.read_csv('../input/Ecommerce Customers')
```

In [4]:

```
customers.head()
```

Out[4]:

	Email	Address	Avatar	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
0	mstephenson@fernandez.com	835 Frank Tunnel\nWrightmouth, MI 82180-9605	Violet	34.497268	12.655651	39.577668	4.082621	587.951054

	Email	Address	Avatar	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
1	hduke@hotmail.com	4547 Archer Common\nDiazchester, CA 06566-8576	DarkGreen	31.926272	11.109461	37.268959	2.664034	392.204933
2	pallen@yahoo.com	24645 Valerie Unions Suite 582\nCobbborough, D...	Bisque	33.000915	11.330278	37.110597	4.104543	487.547505
3	riverarebecca@gmail.com	1414 David Throughway\nPort Jason, OH 22070-1220	SaddleBrown	34.305557	13.717514	36.721283	3.120179	581.852344
4	mstephens@davidson-herman.com	14023 Rodriguez Passage\nPort Jacobville, PR 3...	MediumAquaMarine	33.330673	12.795189	37.536653	4.446	

```
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

	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
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

```
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
```

In [7]:

```
linkcode
sns.set_palette("GnBu_d")
sns.set_style('whitegrid')
```

Exploratory Data Analysis

Let's explore the data!

For the rest of the exercise we'll only be using the numerical data of the csv file.

Use seaborn to create a jointplot to compare the Time on Website and Yearly Amount Spent columns. Does the correlation make sense?

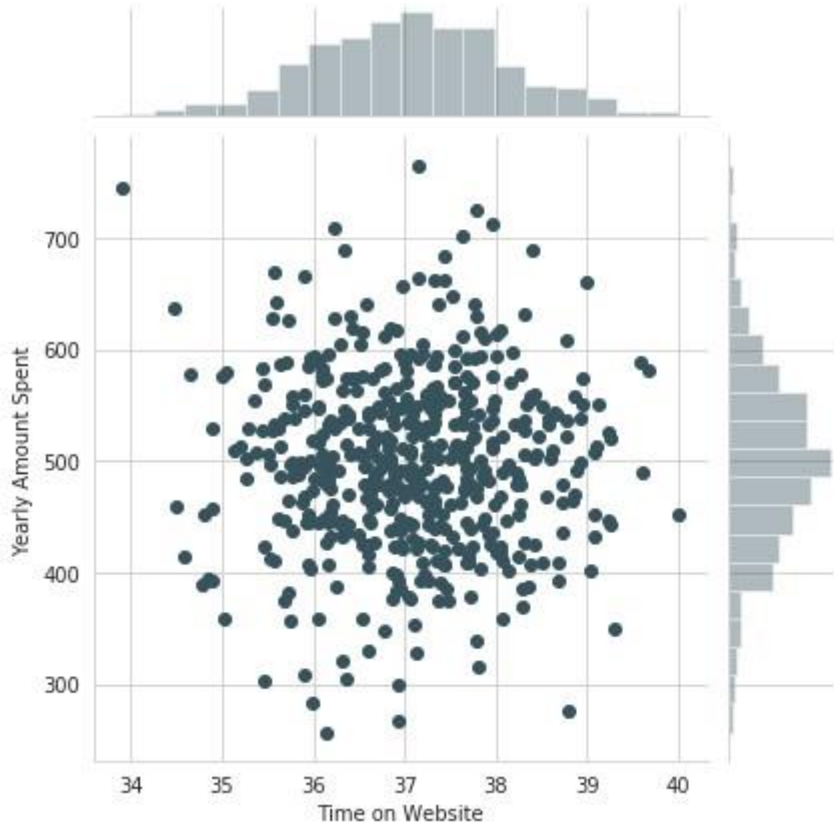
In [8]:

```
sns.jointplot(x='Time on Website',y='Yearly Amount Spent',data=customers)
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarnin
g: Using a non-tuple sequence for multidimensional indexing is deprecated; use
`arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interprete
d as an array index, `arr[np.array(seq)]`, which will result either in an erro
r or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[8]:

```
<seaborn.axisgrid.JointGrid at 0x7f2b0659b9e8>
```



Do the same but with the Time on App column instead.

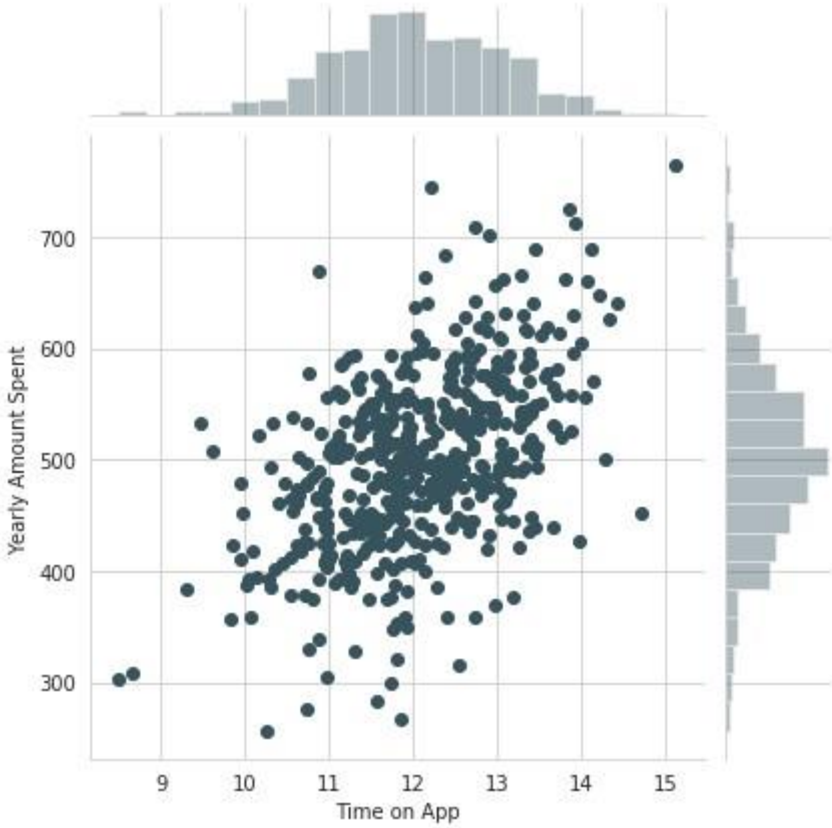
In [9]:

```
sns.jointplot(x='Time on App',y='Yearly Amount Spent',data=customers)
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(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 0x7f2b02492828>
```



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

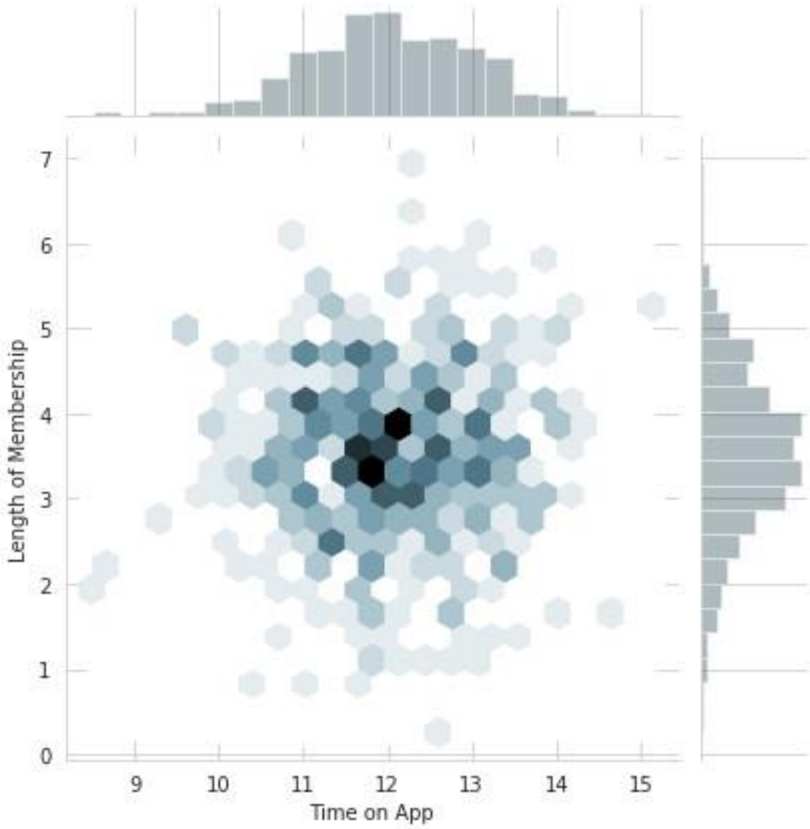
In [10]:

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

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(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.JointGrid at 0x7f2b3b8bf240>
```



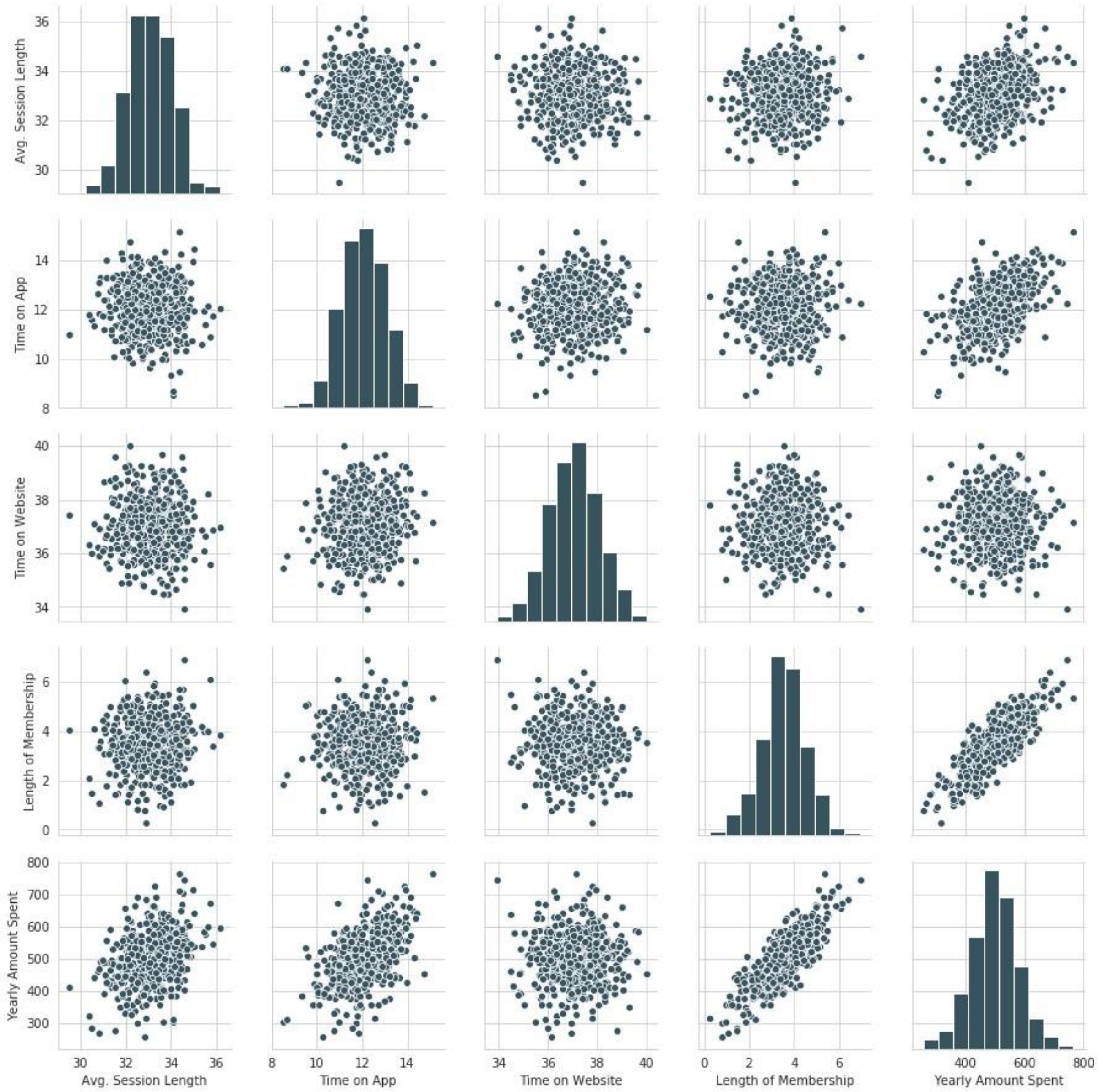
**Let's explore these types of relationships across the entire data set. Use pairplot to recreate the plot below

In [11]:

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

Out[11]:

```
<seaborn.axisgrid.PairGrid at 0x7f2b02210390>
```



Based off this plot what looks to be the most correlated feature with Yearly Amount Spent?

In [12]:

```
# Length of Membership
```

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

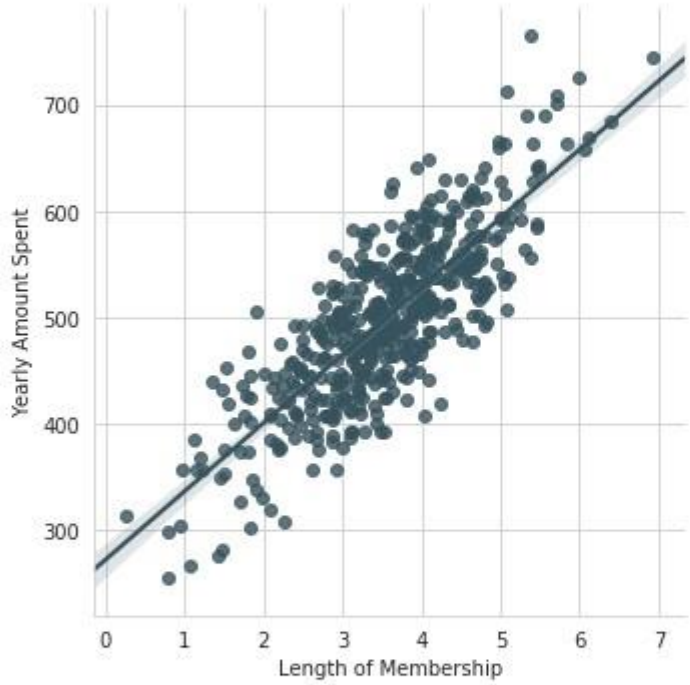
In [13]:

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

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarnin
g: Using a non-tuple sequence for multidimensional indexing is deprecated; use
`arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interprete
d as an array index, `arr[np.array(seq)]`, which will result either in an erro
r or a different result.   return np.add.reduce(sorted[indexer] * weights,
axis=axis) / sumval
```

Out[13]:

```
<seaborn.axisgrid.FacetGrid at 0x7f2b017359b0>
```



Training and Testing Data

Now that we've explored the data a bit, let's go ahead and 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 [14]:

```
X = customers[['Avg. Session Length','Time on App','Time on Website','Length of Me
mbership']]
```

In [15]:

```
y = customers['Yearly Amount Spent']
```

Use 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 [16]:

```
from sklearn.model_selection import train_test_split
```

In [17]:

```
linkcode
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_st
ate=101)
```

Training the Model

Now its time to train our model on our 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=None,
normalize=False)
```

Print out the coefficients of the model

In [21]:

```
lm.coef_
```

Out[21]

```
: array([25.98154972, 38.59015875,  0.19040528, 61.27909654])
```

Predicting Test Data

Now that we have fit our model, let's evaluate its performance by predicting off the test values!

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

In [22]:

```
predictions = lm.predict(X_test)
```

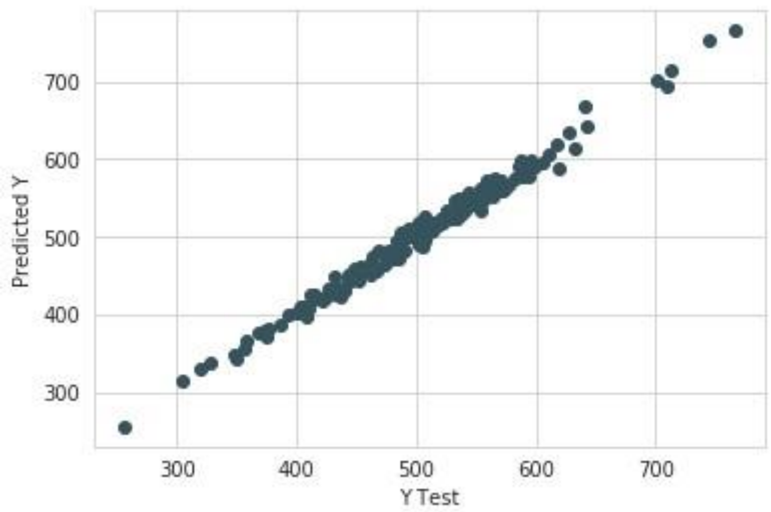
Create a scatterplot of the real test values versus the predicted values.

In [23]:

```
plt.scatter(y_test,predictions) plt.xlabel('Y Test') plt.ylabel('Predicted Y')
```

Out[23]:

```
Text(0,0.5,'Predicted Y')
```



In [24]:

```
from sklearn import metrics
```

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, and the Root Mean Squared Error. Refer to the lecture or to Wikipedia for the formulas

In [25]:

```
print('MAE :'," ", metrics.mean_absolute_error(y_test,predictions)) print('MSE :',"
```

```
",    metrics.mean_squared_error(y_test,predictions))    print('RMAE    :',"    ",
np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

```
MAE :   7.2281486534308295
MSE :   79.8130516509743 RMAE
:   8.933815066978626
```

Residuals

You should have gotten a very good model with a good fit. Let's quickly explore the residuals to make sure everything was okay with our data.

Plot a histogram of the residuals and make sure it looks normally distributed. Use either seaborn distplot, or just plt.hist().

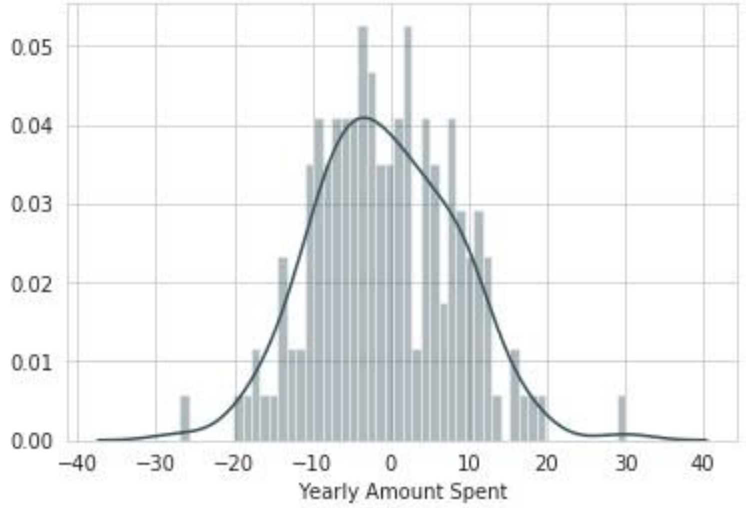
In [26]:

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

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarnin
g: Using a non-tuple sequence for multidimensional indexing is deprecated; use
`arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interprete
d as an array index, `arr[np.array(seq)]`, which will result either in an erro
r or a different result.   return np.add.reduce(sorted[indexer] * weights,
axis=axis) / sumval
```

Out[26]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af7704358>
```



Conclusion

We still want to figure out the answer to the original question, do we focus our efforst on mobile app or website development? Or maybe that doesn't even really matter, and Membership Time is what is really important. Let's see if we can interpret the coefficients at all to get an idea.

Recreate the dataframe below.

In [27]:

```
coeffecients    =    pd.DataFrame(lm.coef_,X.columns)    coeffecients.columns    =
['Coeffecient'] coeffecients
```

Out[27]:

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