Linear Regression

January 8, 2021

1 Linear Regression

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

[3]:
```
customers = pd.read_csv('Ecommerce Customers')
```

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

[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
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	
3	riverarebecca@gmail.com	1414 David Throughway\nPort Jason, OH 22070-1220	SaddleBrown	34.305557	13.717514	36.721283	3.120179	
4	mstephens@davidson-herman.com	14023 Rodriguez Passage\nPort Jacobville, PR 3...	MediumAquaMarine	33.330673	12.795189	37.536653	4.446308	

```
2           487.547505
3           581.852344
4           599.406092
```

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

[5]:
```
       Avg. Session Length  Time on App  Time on Website  \
count           500.000000   500.000000       500.000000
mean             33.053194    12.052488        37.060445
std               0.992563     0.994216         1.010489
min              29.532429     8.508152        33.913847
25%              32.341822    11.388153        36.349257
50%              33.082008    11.983231        37.069367
75%              33.711985    12.753850        37.716432
max              36.139662    15.126994        40.005182

       Length of Membership  Yearly Amount Spent
count            500.000000           500.000000
mean               3.533462           499.314038
std                0.999278            79.314782
min                0.269901           256.670582
25%                2.930450           445.038277
50%                3.533975           498.887875
75%                4.126502           549.313828
max                6.922689           765.518462
```

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

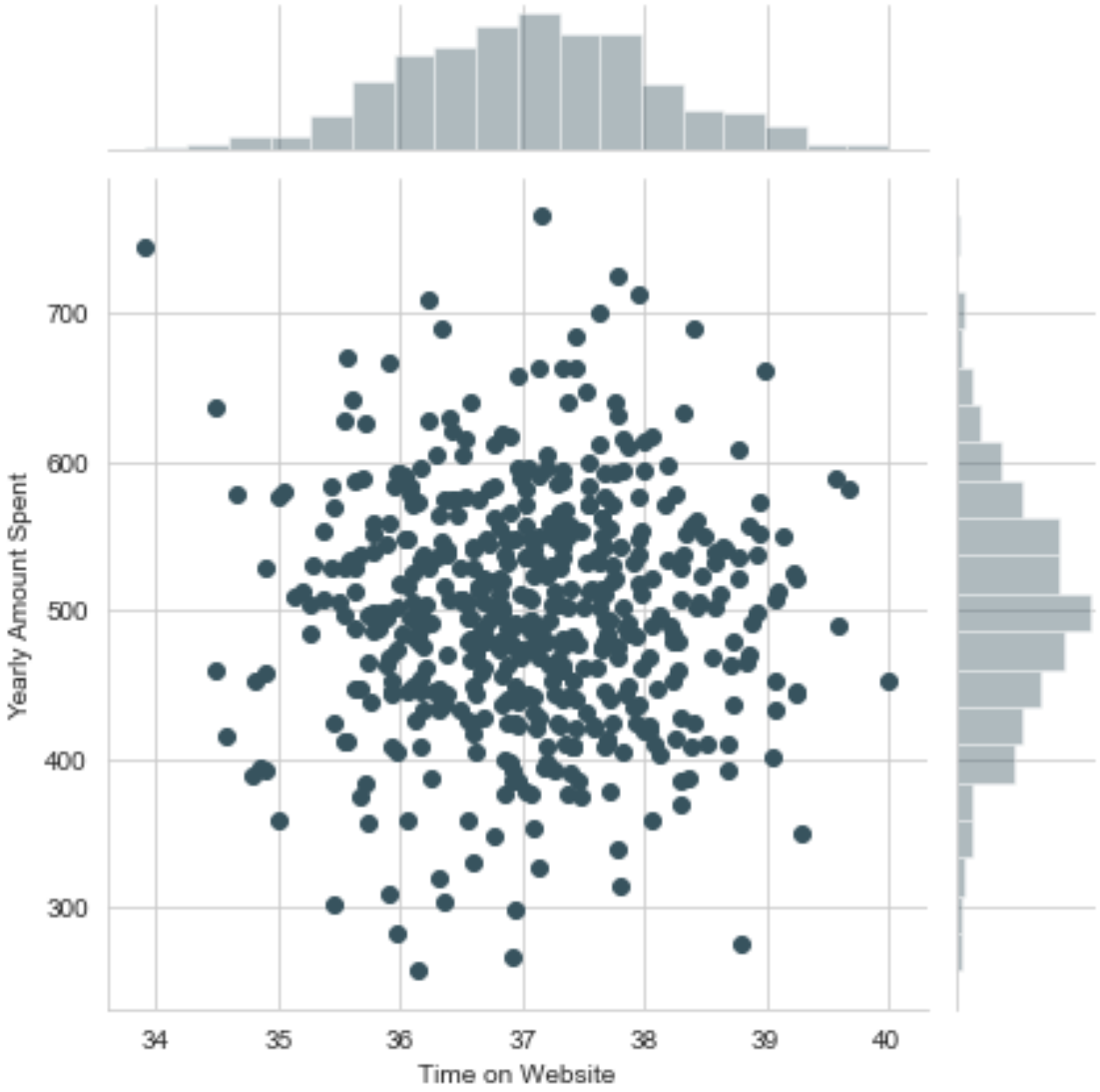
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
Email                   500 non-null object
Address                 500 non-null object
Avatar                  500 non-null object
Avg. Session Length     500 non-null float64
Time on App             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: 31.4+ KB
```

1.0.1 Exploratory Data Analysis

Jointplot to compare the Time on Website and Yearly Amount Spent columns using seaborn.

[11]:
```
sns.set_palette("GnBu_d")
sns.set_style('whitegrid')
sns.jointplot(x='Time on Website',y='Yearly Amount Spent',data=customers)
```

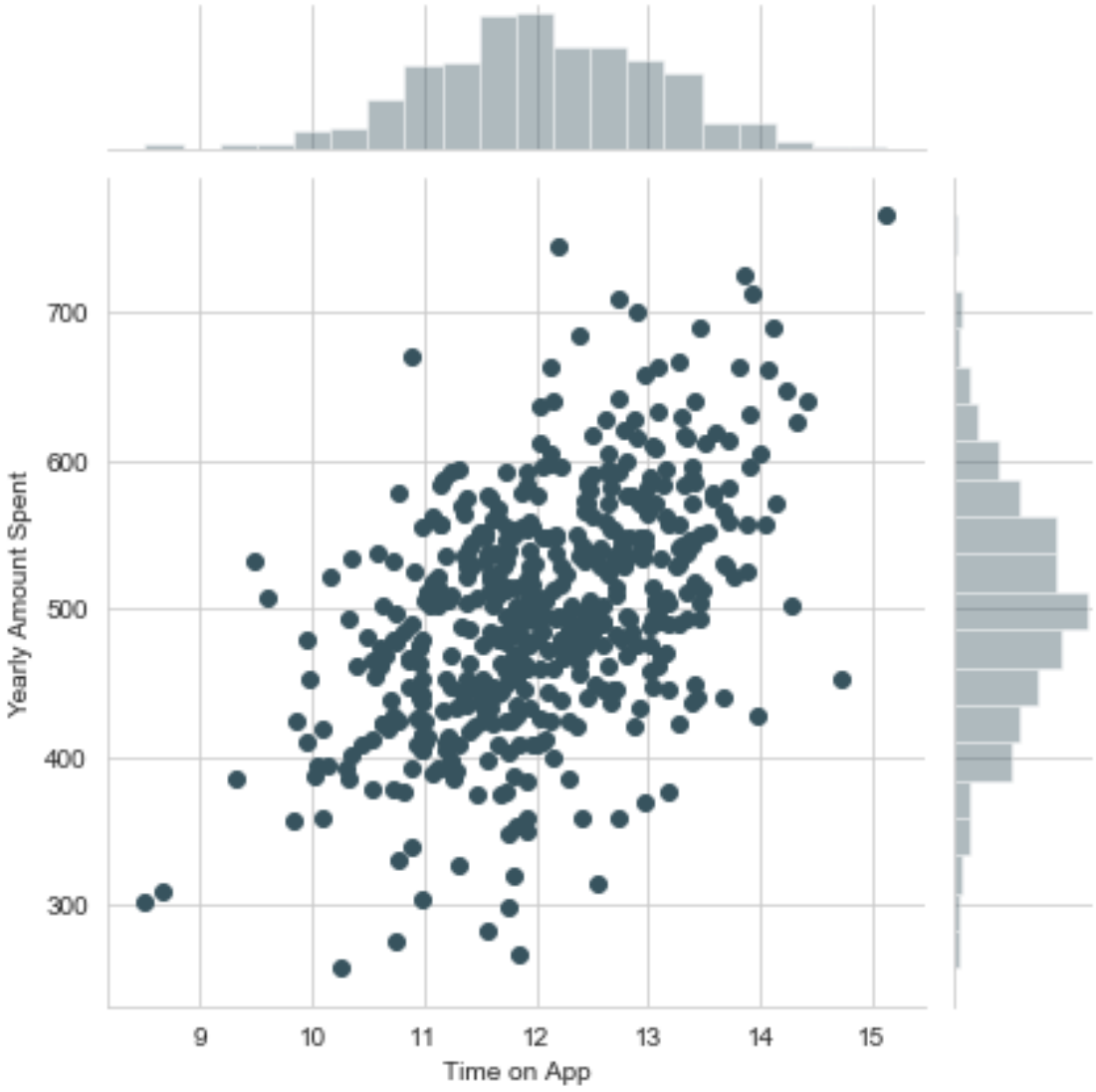
[11]: <seaborn.axisgrid.JointGrid at 0x2ccbd3cec08>



Jointplot to compare the Time on App and Yearly Amount Spent columns using seaborn.

[10]:
```
sns.jointplot(x='Time on App',y='Yearly Amount Spent',data=customers)
```

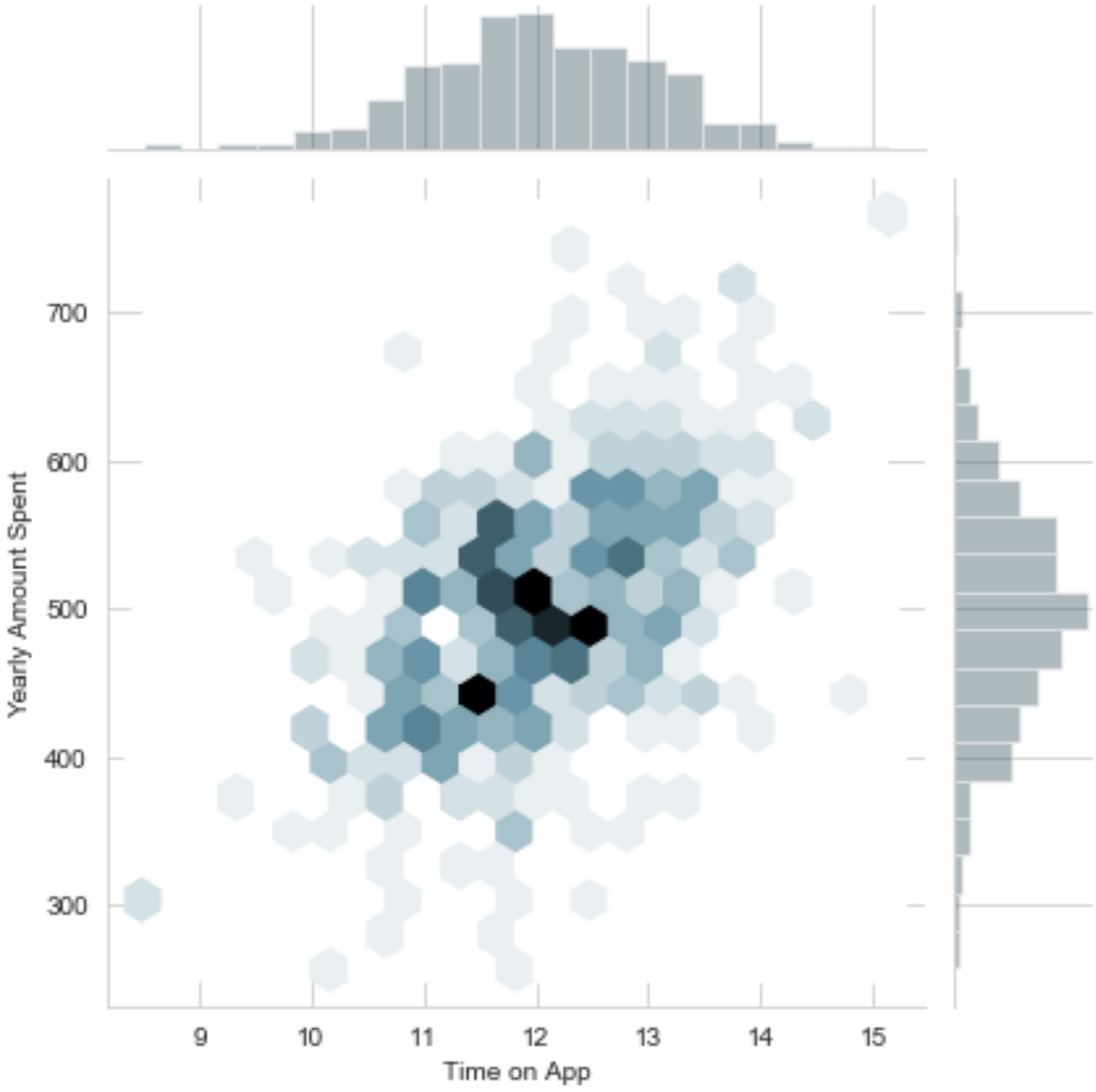
[10]: <seaborn.axisgrid.JointGrid at 0x2ccbd25bf48>



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

[12]:
```
sns.jointplot(x='Time on App',y='Yearly Amount Spent',kind='hex',data=customers)
```

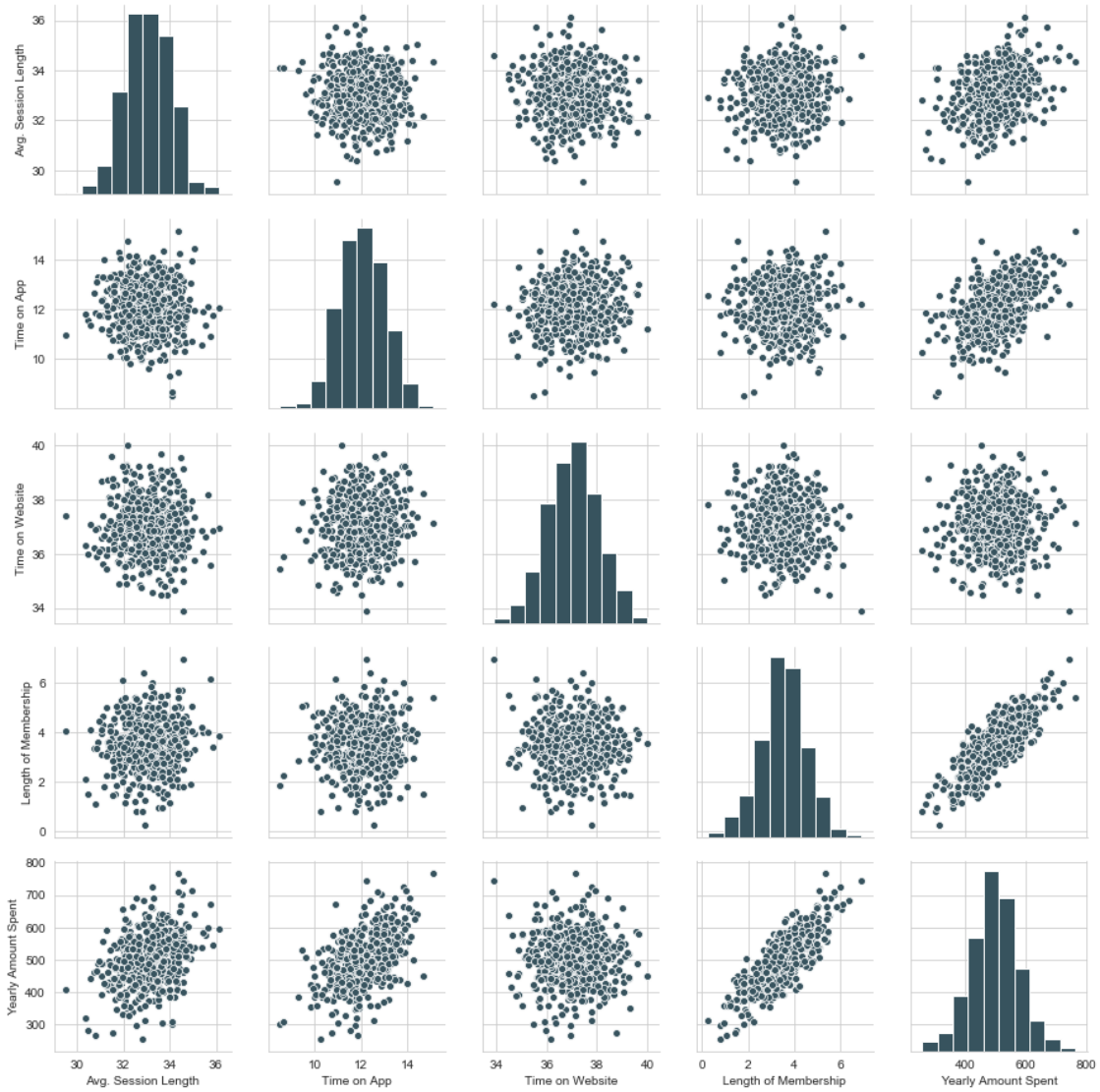
[12]:
```
<seaborn.axisgrid.JointGrid at 0x2ccbd527088>
```



Pairplot using seaborn to explore the type of relationship against the entire dataset

[13]:
```
sns.pairplot(customers)
```

[13]:
```
<seaborn.axisgrid.PairGrid at 0x2ccbd68aa88>
```



Linear model plot of Yearly Amount Spent vs. Length of Membership.

1.0.2 Training and Testing Data

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

[14]:
```
customers.columns
```

[14]:
```
Index(['Email', 'Address', 'Avatar', 'Avg. Session Length', 'Time on App',
       'Time on Website', 'Length of Membership', 'Yearly Amount Spent'],
      dtype='object')
```

[15]:
```
X=customers[['Avg. Session Length', 'Time on App','Time on Website', 'Length of␣
↪Membership']]
```

[16]:
```
y=customers['Yearly Amount Spent']
```

Use model_selection.train_test_split from sklearn to split the data into training and testing sets

[17]:
```
from sklearn.model_selection import train_test_split
```

[18]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

Training the Model

[19]:
```
from sklearn.linear_model import LinearRegression
```

[20]:
```
lm = LinearRegression()
```

[21]:
```
lm.fit(X_train,y_train)
```

[21]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

[22]:
```
print('Coefficients: \n', lm.coef_)
```

```
Coefficients:
 [25.49273883 38.21247657  0.69186749 61.64674188]
```

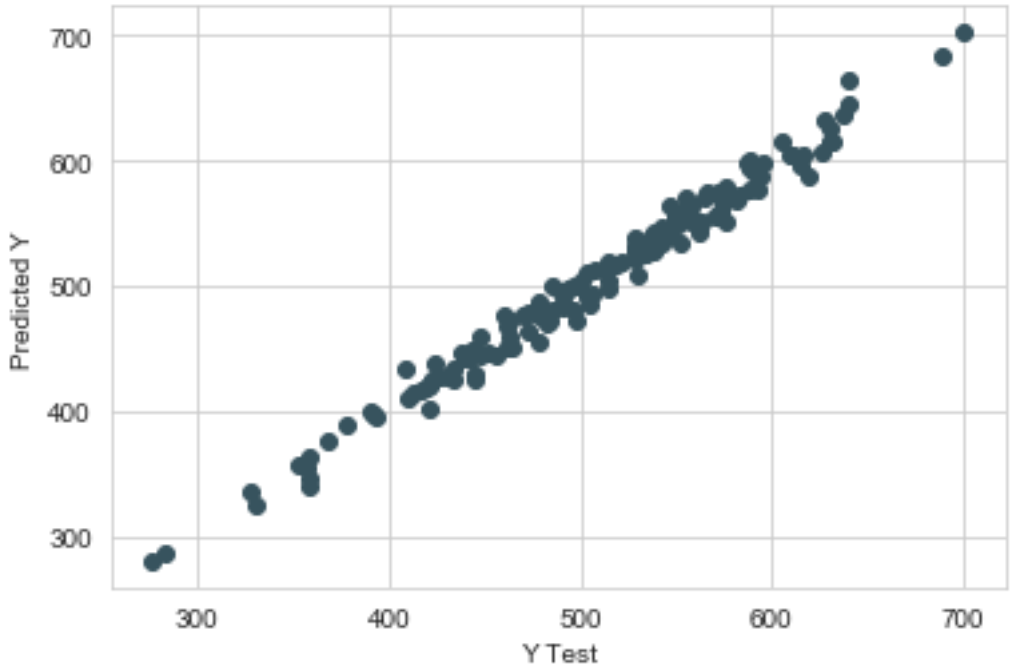
Predicting Test Data

[23]:
```
predictions = lm.predict(X_test)
```

Scatterplot of the real test values versus the predicted values.

[24]:
```
plt.scatter(y_test,predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

[24]:
```
Text(0, 0.5, 'Predicted Y')
```



Evaluating the Model

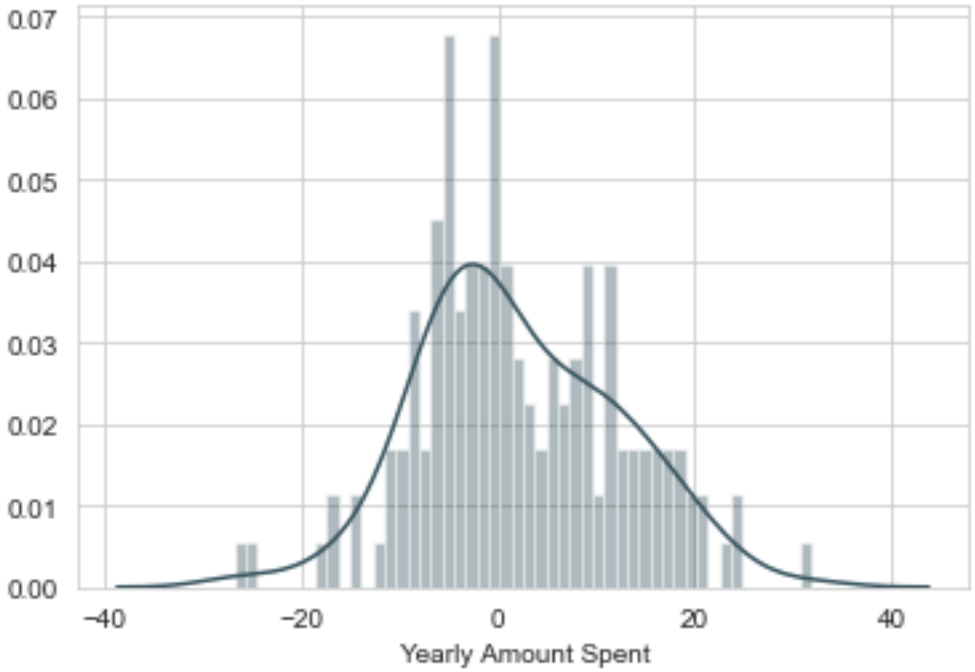
[25]:
```
from sklearn import metrics
```

[26]:
```
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)))
```

```
MAE: 8.116675048097452
MSE: 107.20910289034107
RMSE: 10.35418286927274
```

Residuals Plot a histogram of the residuals and make sure it looks normally distributed

[27]:
```
sns.distplot((y_test-predictions),bins=50);
```


0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
−40
−20
0
20
40
Yearly Amount Spent