Data Analysis Year2- Week9 Lab Exercies An Ecommerce company based in New York City that sells clothing online but they also have 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 for the clothes they want.

The company is trying to decide whether to focus their efforts on their mobile app experience or their website.

Just follow the steps below to analyze the customer data .

Imports

** Import pandas, numpy, matplotlib,and seaborn. (You'll import sklearn as you need it.)**

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

Get the Data

The dataset 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.

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

```
file_path = "EcomCust.csv"
Customers = pd.read_csv(file_path)
```

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

```
Customers.head()
```

	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

```
Customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Email                 500 non-null    object 
 1   Address               500 non-null    object 
 2   Avatar                500 non-null    object 
 3   Avg. Session Length   500 non-null    float64
 4   Time on App           500 non-null    float64
 5   Time on Website       500 non-null    float64
 6   Length of Membership  500 non-null    float64
 7   Yearly Amount Spent   500 non-null    float64
dtypes: float64(5), object(3)
memory usage: 31.4+ KB
```

```
Customers.describe()
```

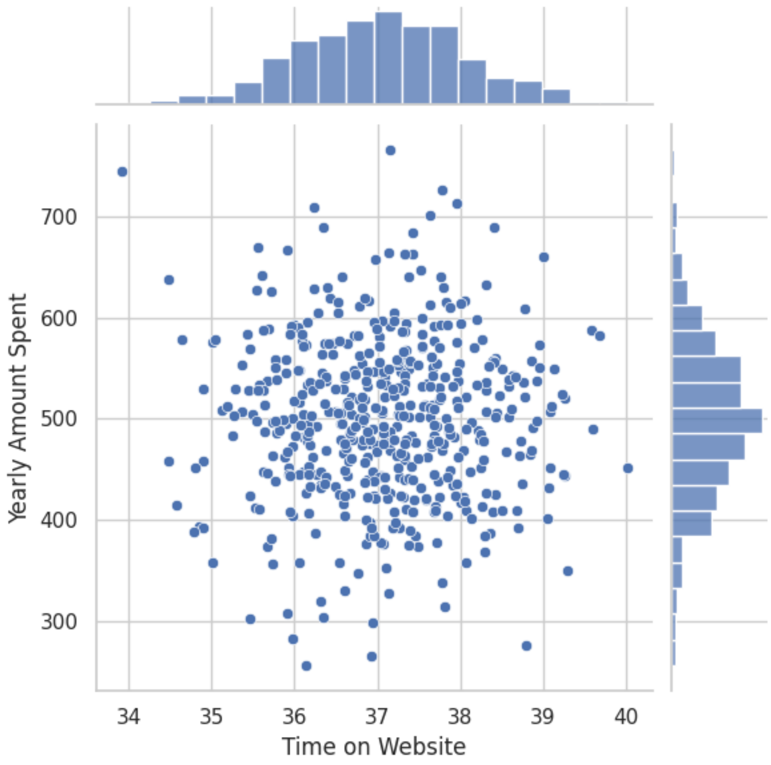
```
       Avg. Session Length  Time on App  Time on Website  Length of Membership  Yearly Amount Spent
count           500.000000   500.000000       500.000000            500.000000           500.000000
mean             33.053194    12.052488        37.060445              3.533462           499.314038
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
```

Exploratory Data Analysis

Use the the numerical data of the csv file.

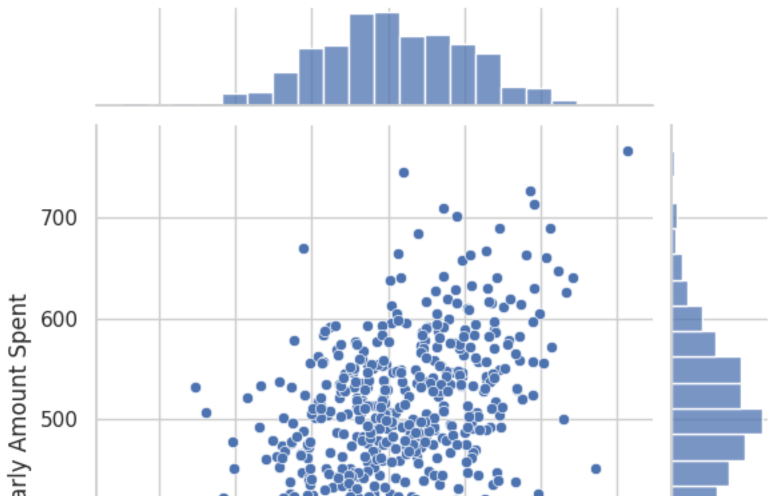
3. Use seaborn to create a jointplot to compare the Time on Website and Yearly Amount Spent columns. Does the correlation make sense?

```
#Create JoinPLot
sns.set(style="whitegrid")
sns.jointplot(x='Time on Website', y='Yearly Amount Spent', data=Customers)
#Show Plot
plt.show()
```

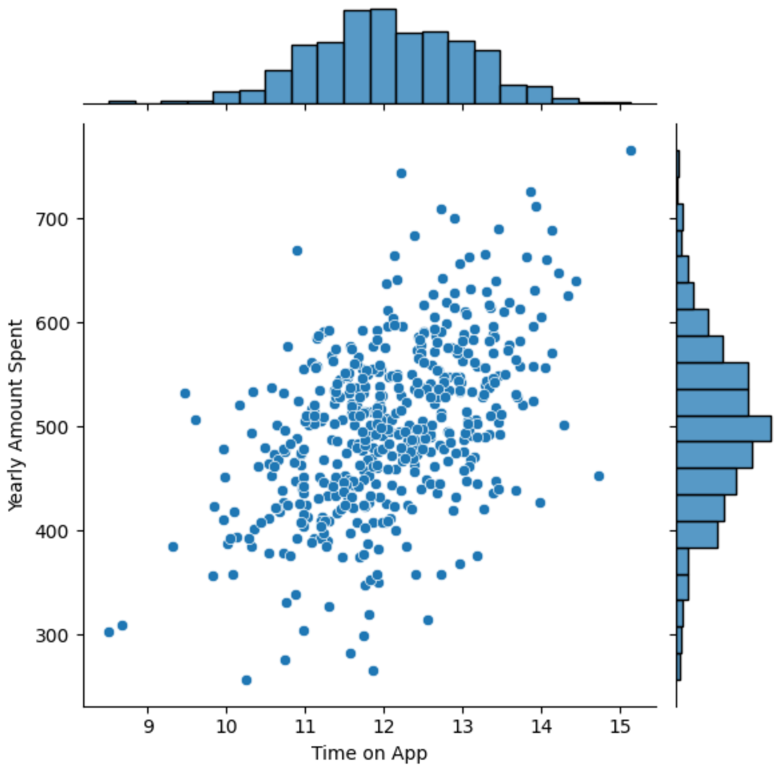


4. Do the same but with the Time on App column instead.

```
#Create the JointPlot
sns.set(style="whitegrid")  # Optional: Set the style for the plot
sns.jointplot(x='Time on App', y='Yearly Amount Spent', data=Customers)

#Show Plot
plt.show()
```

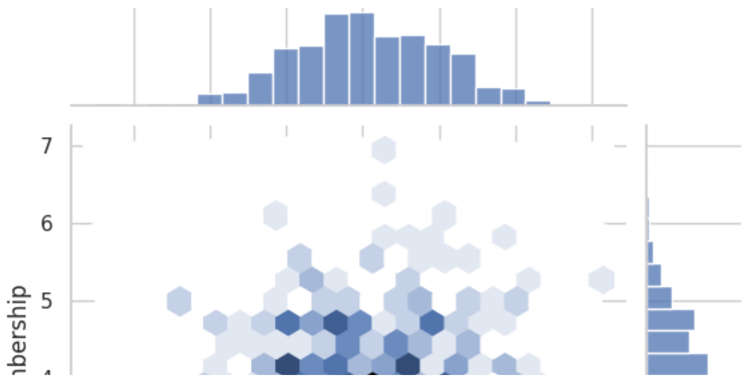


```
<seaborn.axisgrid.JointGrid at 0x7f9ebd9c84c0>
```

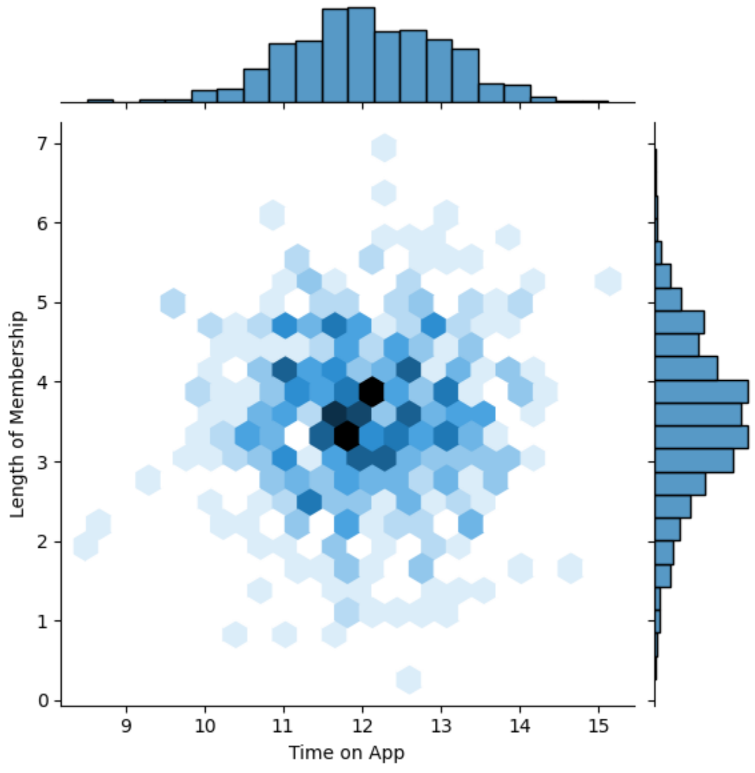


5. Use jointplot to create a 2D hex bin plot comparing Time on App and Length of Membership

```
#Create 2D Hex
sns.set(style="whitegrid")  # Optional: Set the style for the plot
sns.jointplot(x='Time on App', y='Length of Membership', kind='hex', data=Customers)

#Show Plot
plt.show()
```



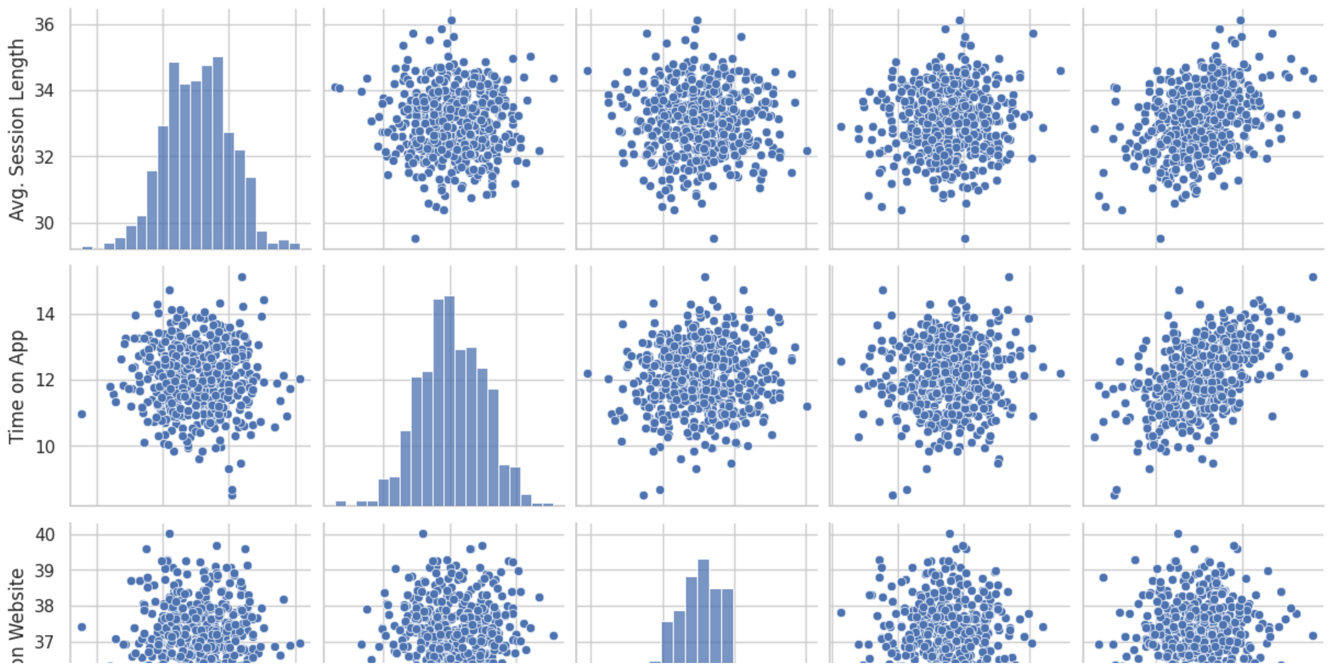
```
<seaborn.axisgrid.JointGrid at 0x7f9ebdd5e7d0>
```



6. Usepairplot to recreate the plot below.(Don't worry about the the colors)

```
#Create AirPlot
sns.pairplot(Customers)
```

```
<seaborn.axisgrid.PairGrid at 0x7f33ea949000>
```


Avg. Session Length
36
34
32
30
Time on App
14
12
10
on Website
40
39
38
37

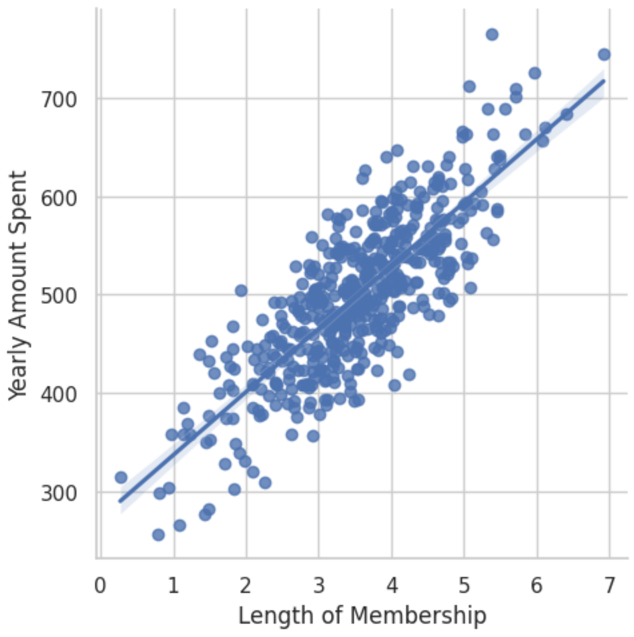
7. Based off this plot what looks to be the most correlated feature with Yearly Amount Spent?

The Length of Membership seems to have the most tightly packed scatter in relation to the Yearly Amount Spent.

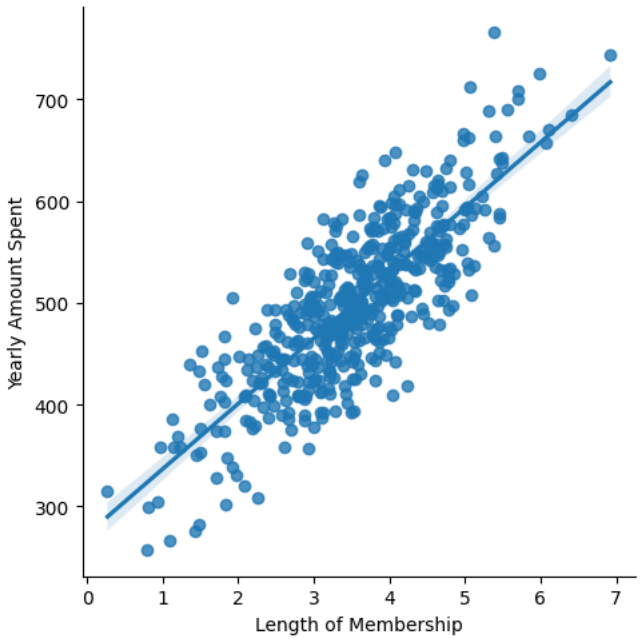
8. Create a linear model plot (using seaborn's lmplot) of Yearly Amount Spent vs. Length of Membership.

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

```
<seaborn.axisgrid.FacetGrid at 0x7f33e7ad9f00>
```



```
<seaborn.axisgrid.FacetGrid at 0x7f9ebbfc0160>
```



Training and Testing Data

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

```
#Select Numericals for X
X = Customers[["Avg. Session Length", "Time on App", "Time on Website", "Length of Membership"]]

#Select Target Value for Y
Y = Customers["Yearly Amount Spent"]
```

10. 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).

```
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=101)
```

Training the Model

11. Import LinearRegression from sklearn.linear_model

```
from sklearn.linear_model import LinearRegression
```

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

```
#Creating Instance of Regression
lm = LinearRegression()
```

13. Train/fit lm on the training data.

```
# Fit the model on the training data
lm.fit(X_train, Y_train)
```

```
▾ LinearRegression
LinearRegression()
```

12. Print out the coefficients of the model

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

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

Predicting Test Data

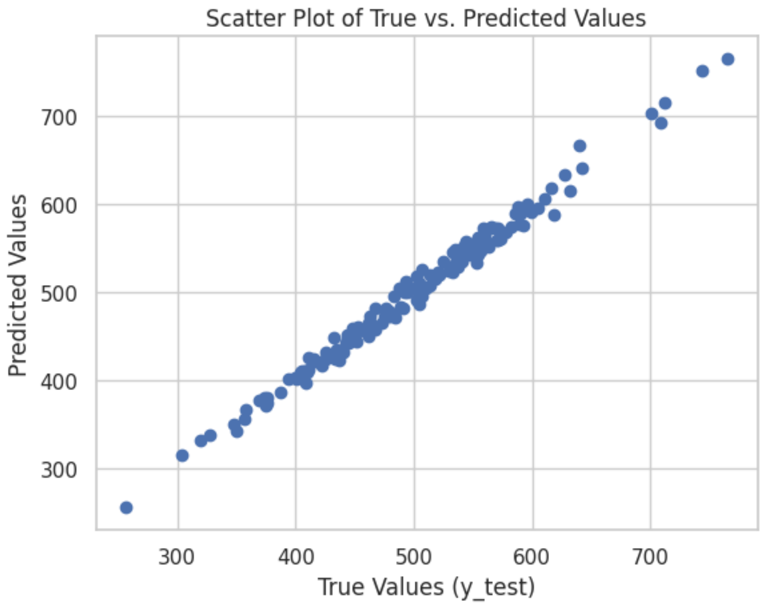
13. Use lm.predict() to predict off the X_test set of the data.

```
from sklearn.metrics import mean_squared_error, r2_score
predictions = lm.predict(X_test)

# Calculate Mean Squared Error
mse = mean_squared_error(y_test, predictions)
print(f"Mean Squared Error: {mse}")

# Calculate R-squared
r2 = r2_score(y_test, predictions)
print(f"R-squared: {r2}")
```

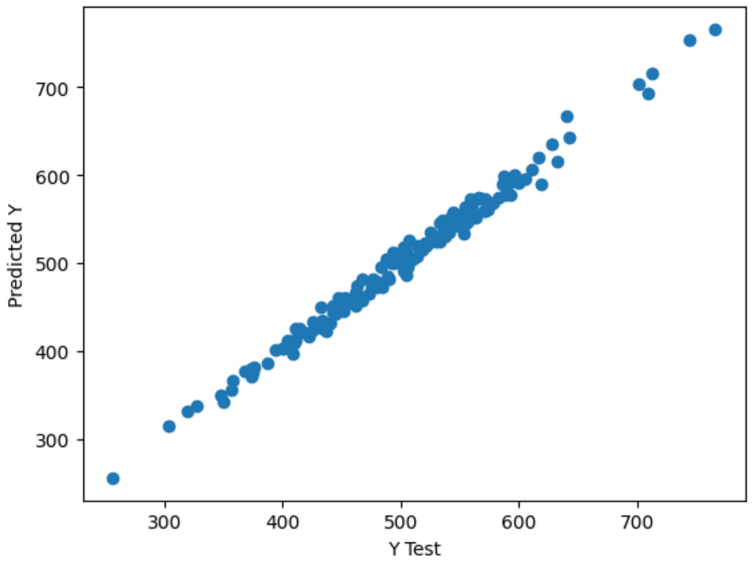
Show hidden output

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

```
plt.scatter(y_test, predictions)
plt.xlabel("True Values (y_test)")
plt.ylabel("Predicted Values")
plt.title("Scatter Plot of True vs. Predicted Values")
plt.show()
```



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



Evaluating the Model

15. Calculate the Mean Absolute Error, Mean Squared Error, and the Root Mean Squared Error.

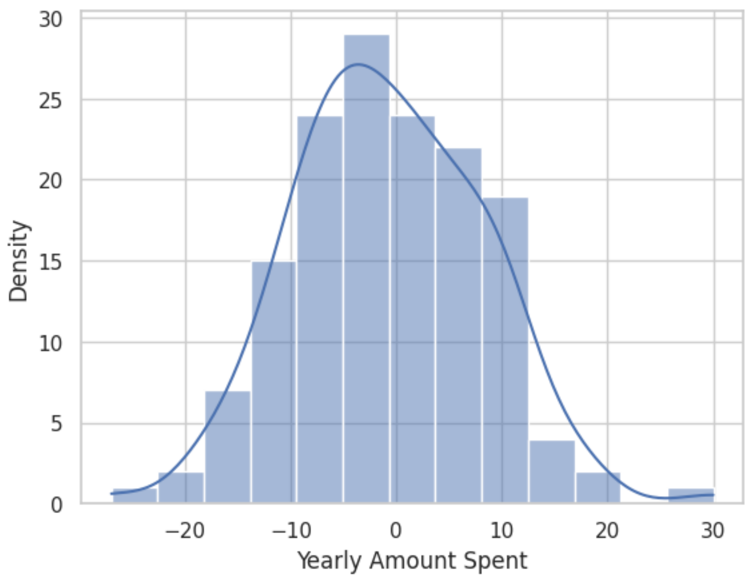
```
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Calculate Mean Absolute Error
mae = mean_absolute_error(y_test, predictions)
print(f"Mean Absolute Error: {mae}")

# Calculate Mean Squared Error
mse = mean_squared_error(y_test, predictions)
print(f"Mean Squared Error: {mse}")

# Calculate Root Mean Squared Error
rmse = mse**0.5
print(f"Root Mean Squared Error: {rmse}")
```

```
Mean Absolute Error: 7.228148653430826
Mean Squared Error: 79.81305165097427
Root Mean Squared Error: 8.933815066978624
```

```
MAE: 7.228148653430826
MSE: 79.81305165097427
RMSE: 8.933815066978624
```

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

```
# Calculate residuals
residuals = y_test - predictions

# Plot histogram of residuals using Seaborn
sns.histplot(residuals, kde=True)
plt.xlabel("Yearly Amount Spent")
plt.ylabel("Density")
plt.show()
```



```
<ipython-input-26-5f2bc21c0ef7>:1: UserWarning: 

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

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