Multiple Linear Regression - Ecommerce

July 12, 2023

[1]:
```
#importing the libraries:

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

[2]:
```
#read the data

data = pd.read_csv('Ecommerce Customers.csv')
data.head()
```

[2]:

	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	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	4.446308	599.406092

```
3           581.852344
4           599.406092
```

[3]:
```
data.describe()
```

[3]:
```
       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
```

[4]:
```
data.shape
```

[4]:
```
(500, 8)
```

[5]:
```
data.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
```

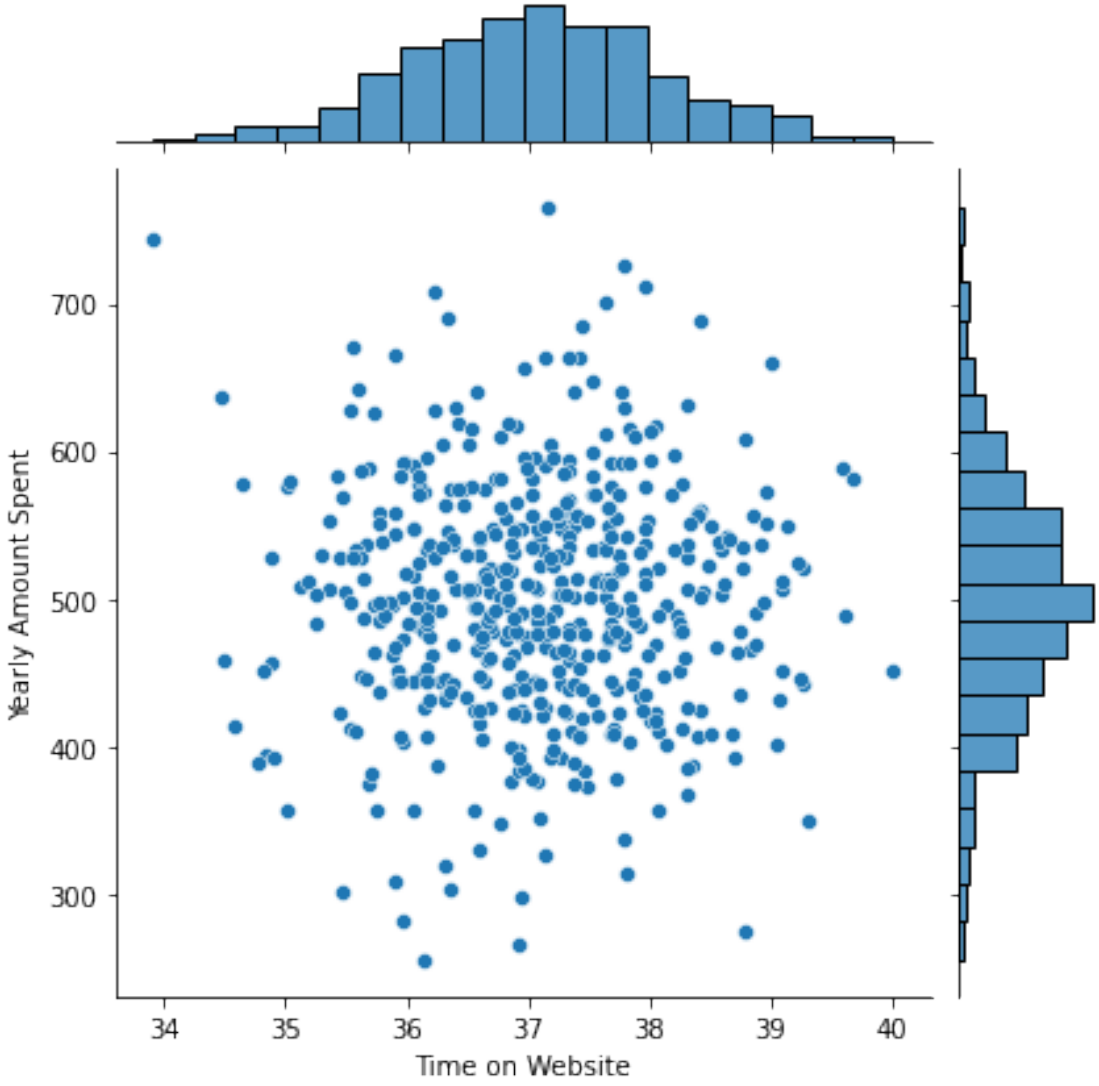
avoid string variables

0.1 EDA

[6]:
```
# testing which variable has maximum impact on dependent variable
# Bivariate analysis
# jointplot to help understand each variable individually and with other␣
 ↪variables

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

[6]: <seaborn.axisgrid.JointGrid at 0x7fd5d14a7690>

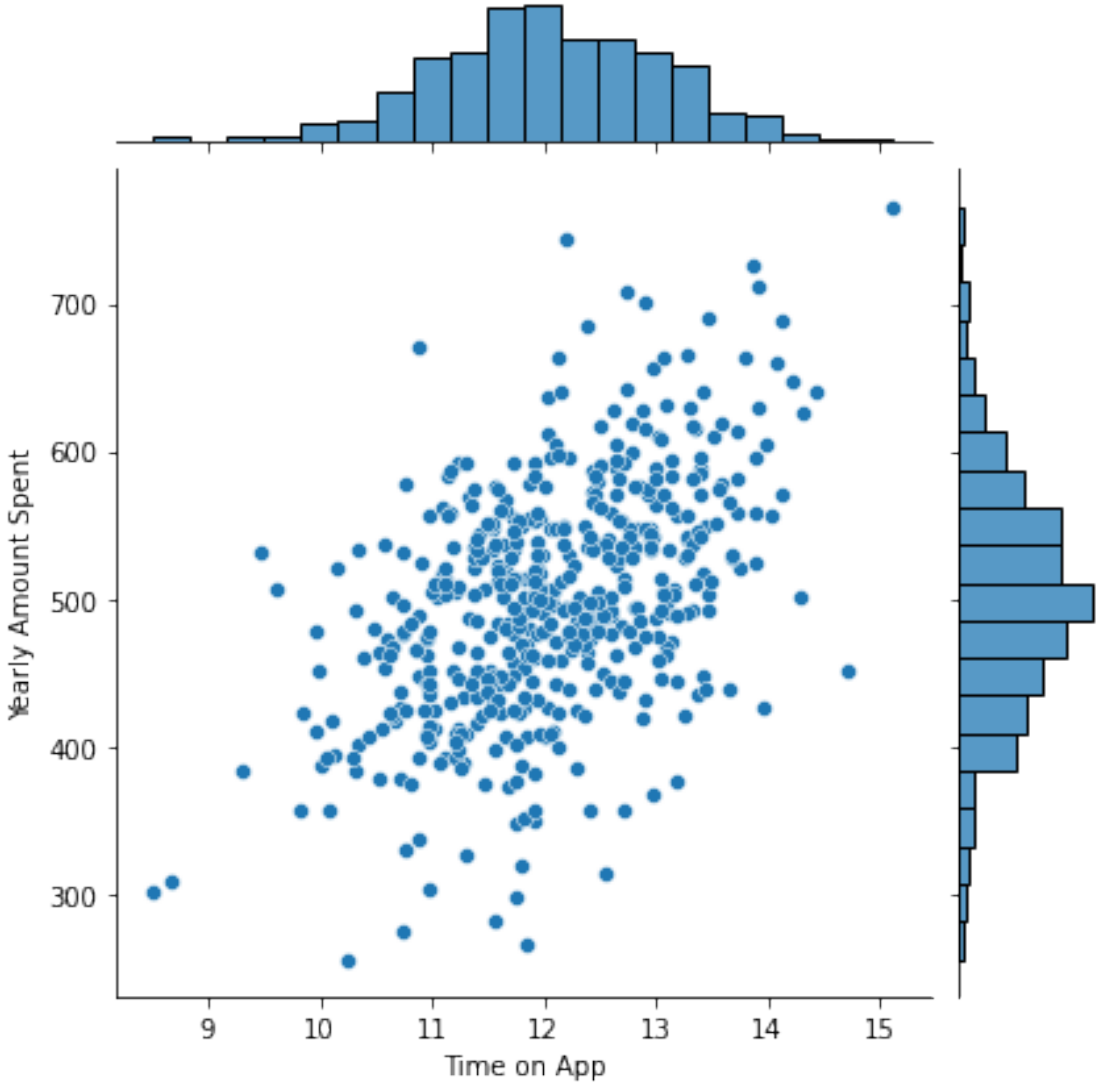


it shows small positive correlation

Impact of Time spent on website is negligible

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

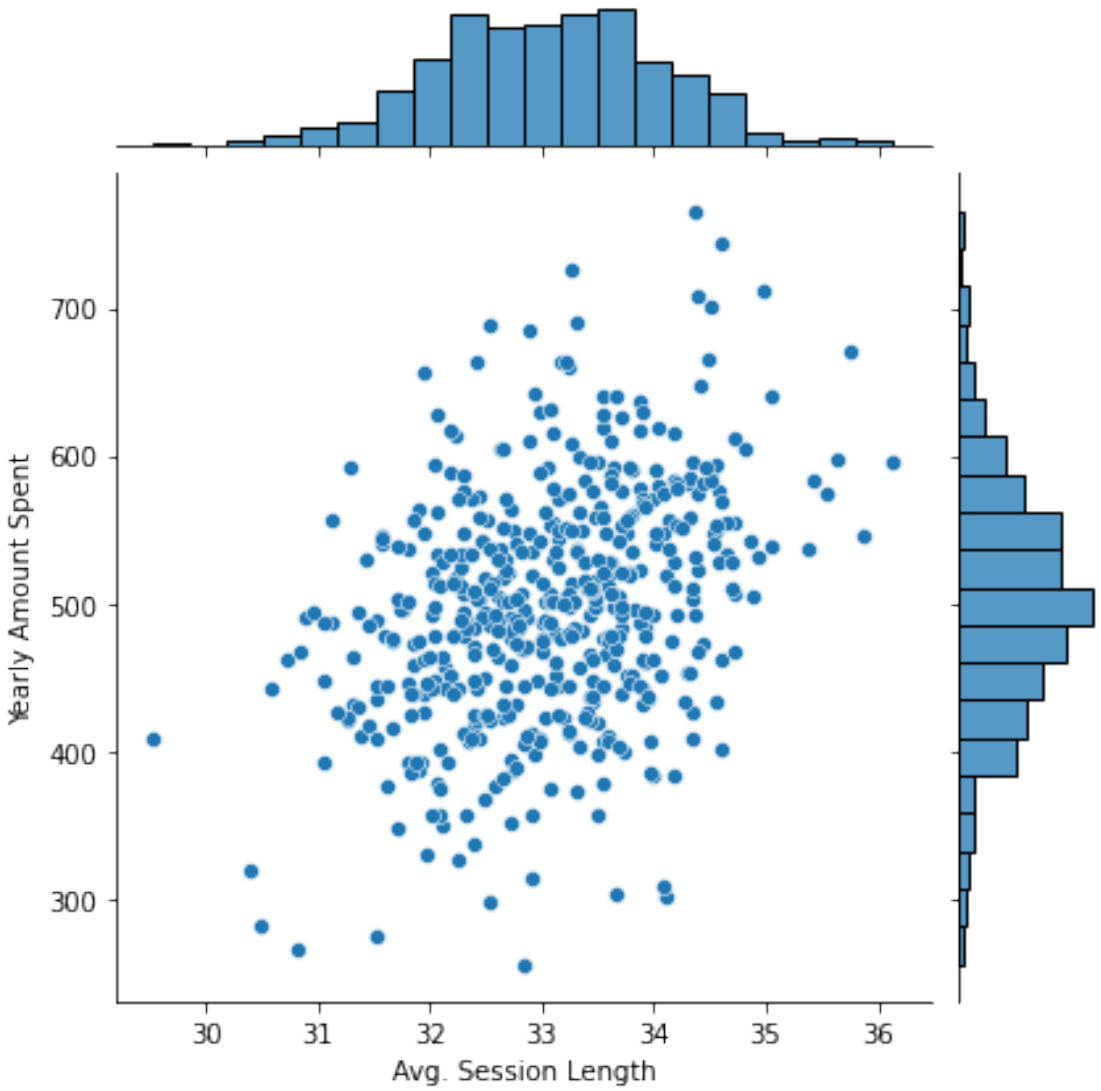
[7]:
```
<seaborn.axisgrid.JointGrid at 0x7fd5ad189450>
```



it shows a clear positive trend

[9]:
```
sns.jointplot(x='Avg. Session Length', y= 'Yearly Amount Spent', data = data)
```

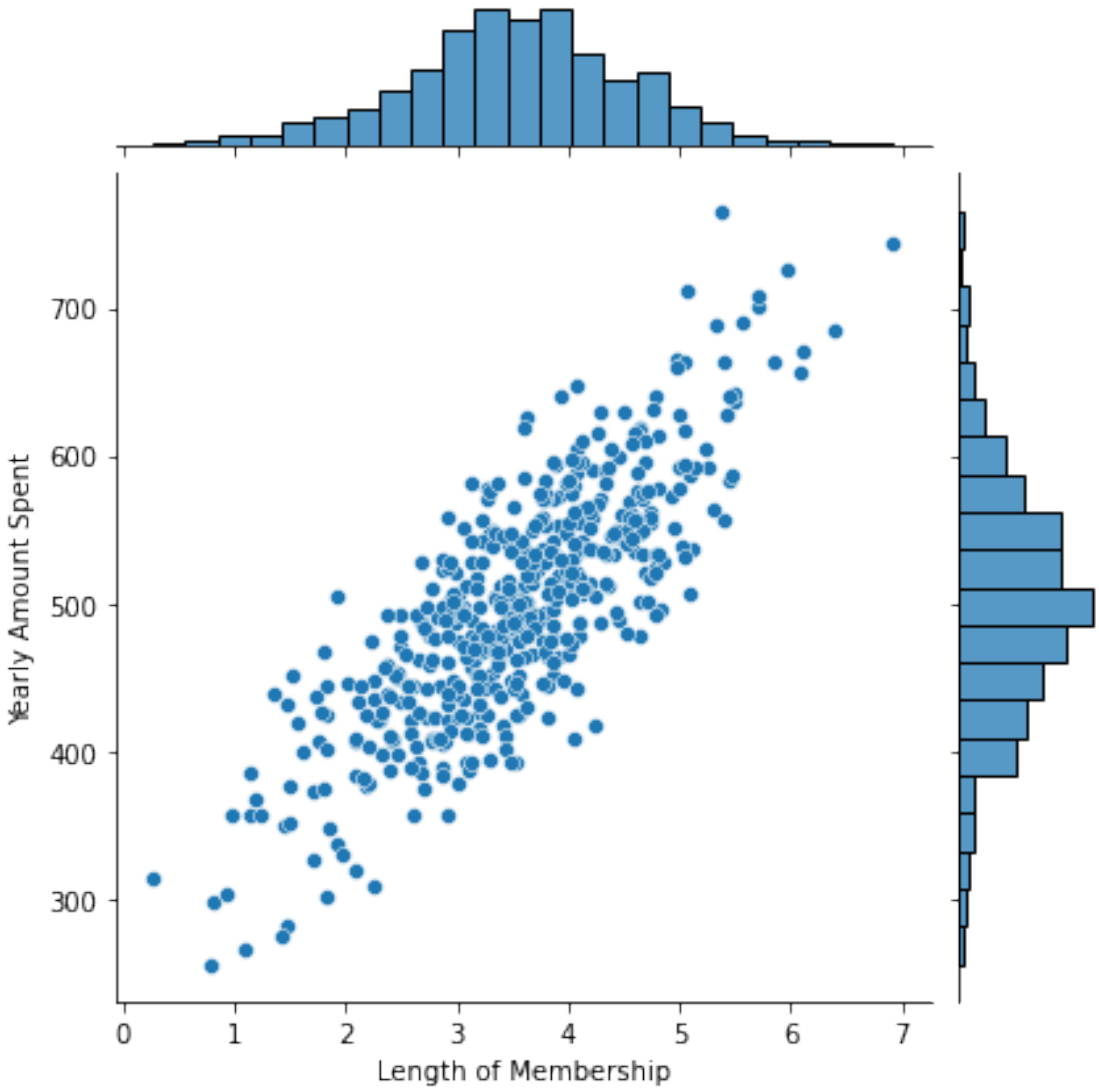
[9]:
```
<seaborn.axisgrid.JointGrid at 0x7fd5ad1db350>
```



it shows a feeble relationship between 2 variables. logic : just to kill time you're spending time on website and not with the intention of buying.

[10]:
```
sns.jointplot(x='Length of Membership', y= 'Yearly Amount Spent', data = data)
```

[10]:
```
<seaborn.axisgrid.JointGrid at 0x7fd5ace11cd0>
```



0.1.1 Length of Membership has maximum impact on Yearly amount spent due to brand loyalty

we will verify this with beta coeff once we build the model

here, we can do heatmap analysis too.

Correlation can be used if more no of numerical variables are present

[12]:
```
data.columns
```

[12]:
```
Index(['Email', 'Address', 'Avatar', 'Avg. Session Length', 'Time on App',
       'Time on Website', 'Length of Membership', 'Yearly Amount Spent'],
```

```
      dtype='object')
```

[13]:
```
# creating the feature set and the dependent variable set
X = data[['Avg. Session Length', 'Time on App', 'Time on Website', 'Length of␣
↪Membership']]

# we have 4 variables : for subsetting we need to combine them in a container␣
↪here that is a list. One [] is for slicing data ; another [] is for␣
↪combining features together
```

[18]:
```
y = data['Yearly Amount Spent']
```

[ ]:
```

```

[20]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.30,␣
↪random_state =42)
```

[21]:
```
from sklearn.linear_model import LinearRegression

lm = LinearRegression()
```

[22]:
```
# fitting the training data to this model

lm.fit(X_train, y_train)
```

[22]:
```
LinearRegression()
```

[25]:
```
# Checking the coefficient (beta value) of the most important predictor i.e.␣
↪length of membership
lm.coef_
```

[25]:
```
array([25.72425621, 38.59713548,  0.45914788, 61.67473243])
```

check that 61 coef is for length of membership by looking at the order of variables in X

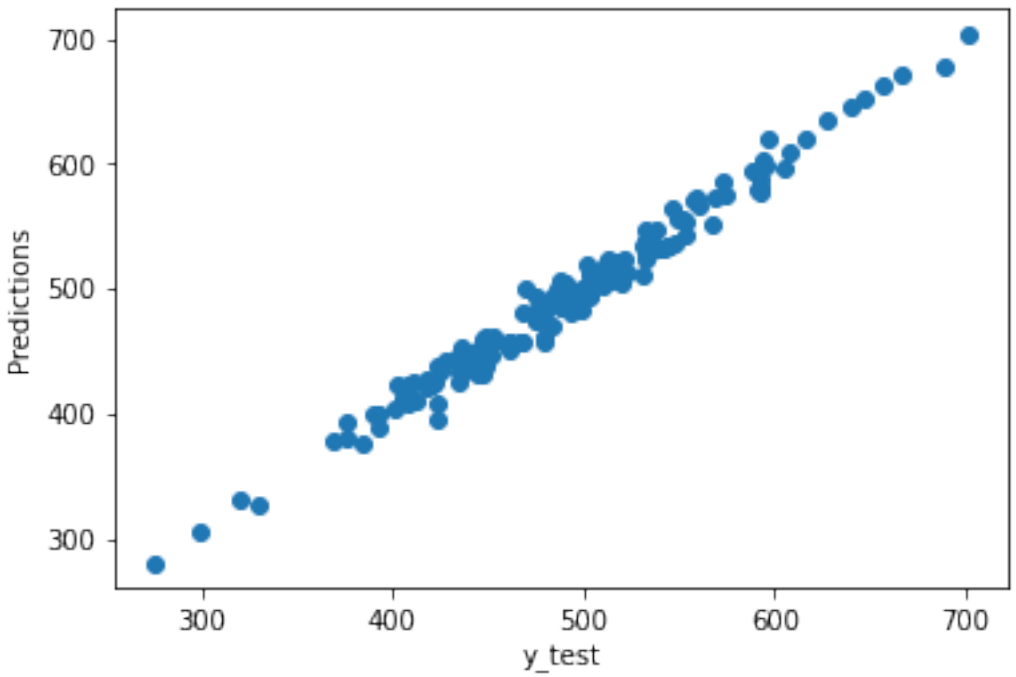
[27]:
```
pred = lm.predict(X_test)

# Creating a basic scatter plot to test the actuals with the predictions

plt.scatter(y_test, pred)

plt.xlabel('y_test')
plt.ylabel('Predictions')
```

[27]:
```
Text(0, 0.5, 'Predictions')
```



we see that points are very close knit, we do not see any major deviations. Ideally the model is a very good fit.

we do not use Mean sq error (MSE) OR root mean sq error(RMSE) as evaluation metric because there is no standard threshold. we just know that it's good if it's low as it's an error component but we don't know how low

like MAPE has a threshold, below 20 it's a good model.

We use "r squared" to evaluate accuracy as we know the closer it is to 100 or 1 the better

[28]:
```
# To evaluate the accuracy

from sklearn.metrics import r2_score

r2 = r2_score(y_test, pred)
print(r2)
```

```
0.9808757641125855
```

[ ]:
```

```