Linear Regression Training Project: Ecommerce Clients

The data includes information about customers of an e-commerce website, including the following:

- 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.

In this project, we suppose that the company is trying to decide whether to focus their efforts on their mobile app experience or their website. We are here to help them make a data-driven decision.

In [29]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading the Data

In [2]:
```
customers = pd.read_csv('Ecommerce Customers')
```

In [3]:
```
customers.head()
```

Out[3]:

	Email	Address	Avatar	Avg Sessior Lengtl
0	mstephenson@fernandez.com	835 Frank Tunnel\nWrightmouth, MI 82180-9605	Violet	34.49726ε
1	hduke@hotmail.com	4547 Archer Common\nDiazchester, CA 06566-8576	DarkGreen	31.92627
2	pallen@yahoo.com	24645 Valerie Unions Suite 582\nCobbborough, D...	Bisque	33.00091
3	riverarebecca@gmail.com	1414 David Throughway\nPort Jason, OH 22070-1220	SaddleBrown	34.30555
4	mstephens@davidson-herman.com	14023 Rodriguez Passage\nPort Jacobville, PR 3...	MediumAquaMarine	33.33067

In [4]:
```
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
```

In [12]:
```
customers.describe()
```

Out[12]:

	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

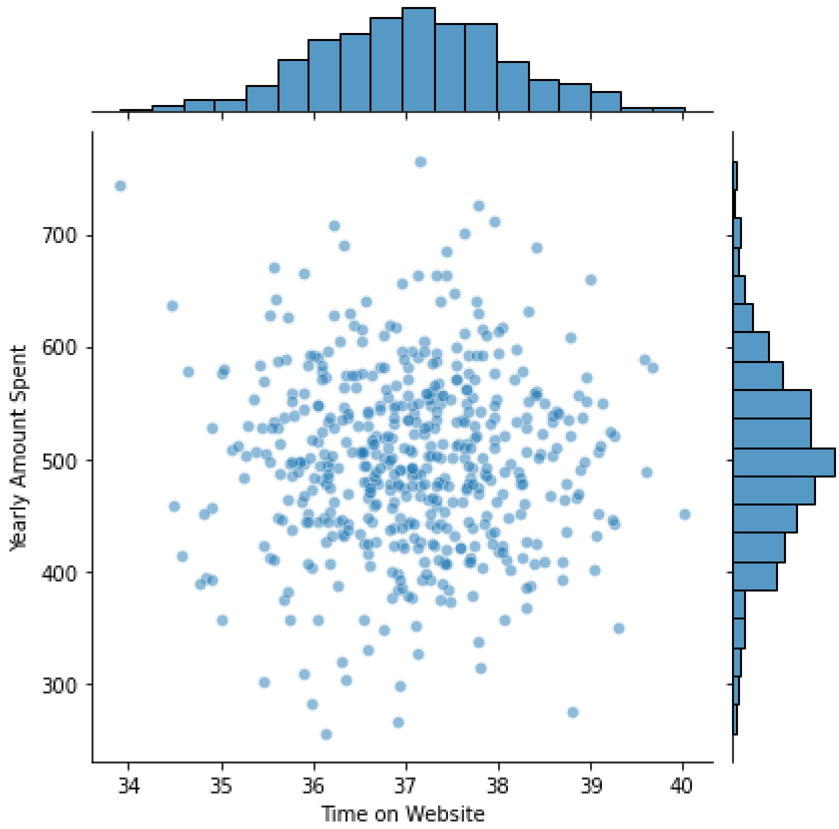
Exploratory Data Analysis

There doesn't seem to be much correlation between the time on the desktop website with the amount that clients spend per year. On the other side, the second graph shows that there seems to be a small correlation between the time spent on the app and the yearly spending. This is probably because these clients tend to spend less time browsing on the phone.

After analysing the pairplot, there does seem to be one big positive correlation between two variables: the length of membership and the yearly expenditure. In the end we recrate this plot to visualise the regression line.

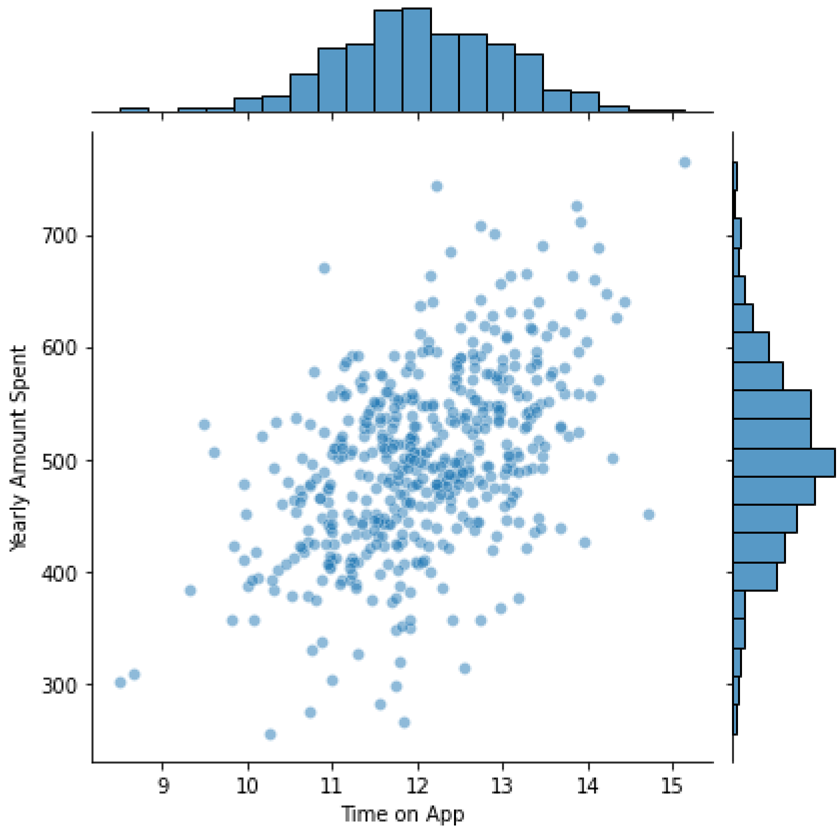
In [5]:
```
sns.jointplot(x='Time on Website', y='Yearly Amount Spent', data=customer
```

Out[5]: <seaborn.axisgrid.JointGrid at 0x14d8bb640>



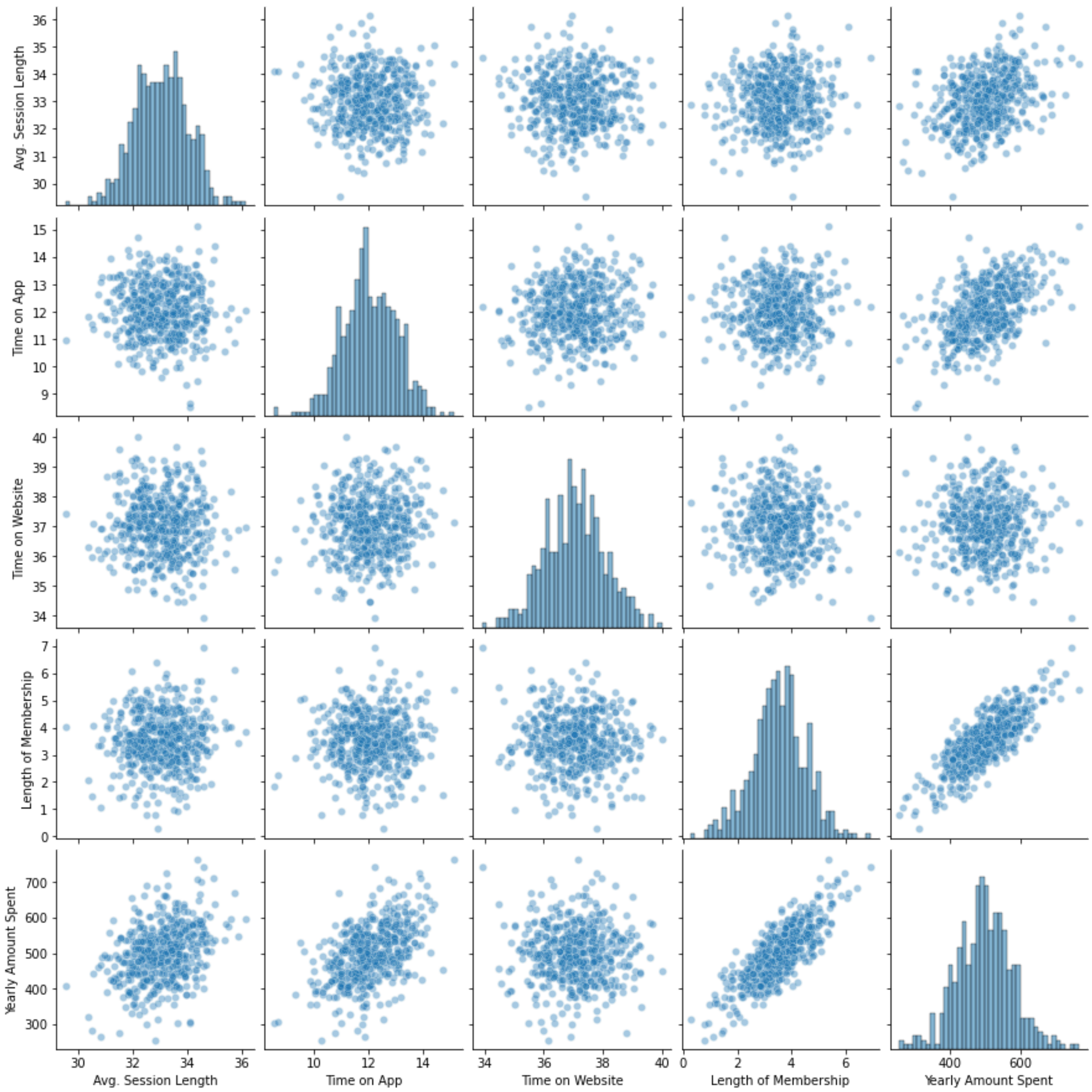
In [6]:
```
sns.jointplot(x='Time on App', y='Yearly Amount Spent', data=customers, a
```

Out[6]: <seaborn.axisgrid.JointGrid at 0x14d9ff910>



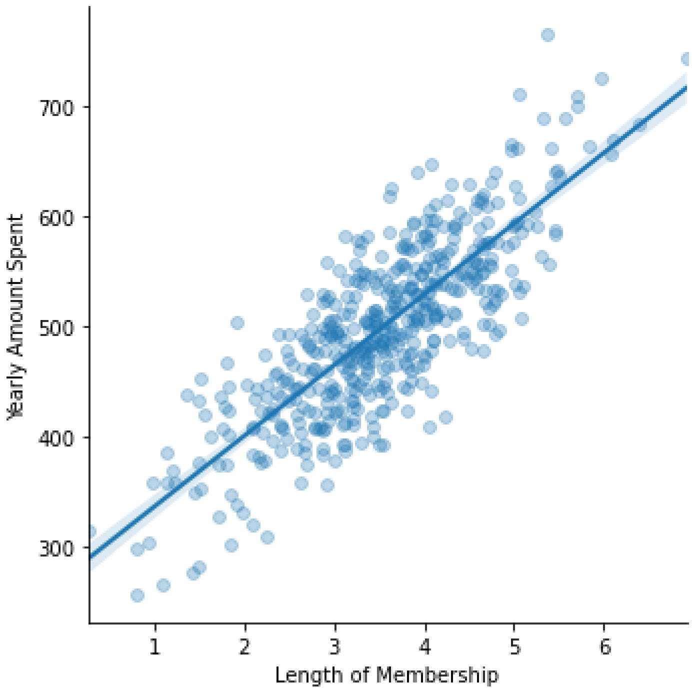
In [7]:
```
sns.pairplot(customers,
             kind='scatter',
             plot_kws={'alpha':0.4},
             diag_kws={'alpha':0.55, 'bins':40})
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x14dc67f10>



In [36]:
```
sns.lmplot(x='Length of Membership',
           y='Yearly Amount Spent',
           data=customers,
           scatter_kws={'alpha':0.3})
```

Out[36]: <seaborn.axisgrid.FacetGrid at 0x7fdd76fc4970>



Splitting the data

X are the predictores, and y is the output. We want to do is create a model that will take in the values in the X variable and predict y with a linear regression algorithm. We will use the SciKit Learn library to create the model.

In [8]:
```
from sklearn.model_selection import train_test_split
```

In [9]:
```
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
```

In [11]:
```
X = customers[['Avg. Session Length', 'Time on App', 'Time on Website', '
y = customers['Yearly Amount Spent']
```

In [14]:
```
X.head()
y.head()
```

Out[14]:
```
0    587.951054
1    392.204933
2    487.547505
3    581.852344
4    599.406092
Name: Yearly Amount Spent, dtype: float64
```

In [17]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, ra
```

Training the Model with multivariable regression using Scikit Learn

We create the model and feed the training data to it. This model will tell us which input has the biggest impact in the output (yearly expenditure). As the plots suggested, we find that the most important coefficient is that of the "Length of Membership" predictor, followed by the 'Time on App' and the 'Avg. Session Length'. The time on website does not seem to be an important factor to the amount a customer spends per year.

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

In [22]:
```
lm = LinearRegression()
```

In [23]:
```
lm.fit(X_train, y_train)
```

Out[23]:
```
LinearRegression()
```

In [25]:
```
lm.coef_
```

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

In [34]:
```
lm.score(X, y)
```

Out[34]:
```
0.9842821675307221
```

In [24]:
```
cdf = pd.DataFrame(lm.coef_,X.columns,columns=['Coef'])
print(cdf)
```

```
                           Coef
Avg. Session Length   25.724256
Time on App           38.597135
Time on Website        0.459148
Length of Membership  61.674732
```

Training the model with multivariable regression using OLS

Allows us to get more details about the moel

In [47]:
```
X = sm.add_constant(X_train)
model = sm.OLS(y_train, X)
```

```
model_fit = model.fit()
print(model_fit.summary())
```

```
                            OLS Regression Results
==============================================================================
=====
Dep. Variable:     Yearly Amount Spent   R-squared:
0.985
Model:                             OLS   Adj. R-squared:
0.985
Method:                  Least Squares   F-statistic:
5825.
Date:                 Fri, 09 Dec 2022   Prob (F-statistic):          3.46
e-315
Time:                         17:35:26   Log-Likelihood:                -1
296.2
No. Observations:                  350   AIC:
2602.
Df Residuals:                      345   BIC:
2622.
Df Model:                            4
Covariance Type:             nonrobust
==============================================================================
==============
                           coef    std err          t      P>|t|      [0.0
25      0.975]
------------------------------------------------------------------------------
--------------
const                -1050.6537     26.458    -39.710      0.000   -1102.6
94    -998.614
Avg. Session Length     25.7243      0.534     48.137      0.000      24.6
73      26.775
Time on App             38.5971      0.528     73.045      0.000      37.5
58      39.636
Time on Website          0.4591      0.520      0.884      0.377      -0.5
63       1.481
Length of Membership    61.6747      0.516    119.540      0.000      60.6
60      62.690
==============================================================================
====
Omnibus:                         1.523   Durbin-Watson:
2.024
Prob(Omnibus):                   0.467   Jarque-Bera (JB):
1.262
Skew:                           -0.108   Prob(JB):
0.532
Kurtosis:                        3.199   Cond. No.                    2.56
e+03
==============================================================================
====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
[2] The condition number is large, 2.56e+03. This might indicate that ther
e are
strong multicollinearity or other numerical problems.
```

Predicting Test Data

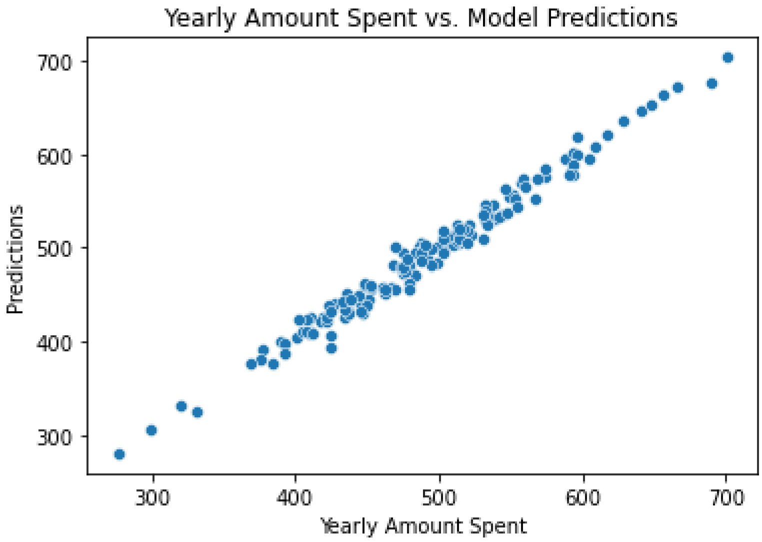
The scatter plot below plots the actual y values to the model's predictions. The model seems to behave accurately.

In [26]:
```
predictions = lm.predict(X_test)
```

In [31]:
```
sns.scatterplot(y_test, predictions)
plt.ylabel('Predictions')
plt.title('Yearly Amount Spent vs. Model Predictions')
```

```
/opt/homebrew/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.p
y:36: FutureWarning: Pass the following variables as keyword args: x, y. F
rom version 0.12, the only valid positional argument will be `data`, and p
assing other arguments without an explicit keyword will result in an error
or misinterpretation.
  warnings.warn(
```

Out[31]: Text(0.5, 1.0, 'Yearly Amount Spent vs. Model Predictions')



Evaluation of the model

In [32]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error
import math
```

In [64]:
```
print('Mean Absolute Error:',mean_absolute_error(y_test, predictions))
print('Mean Squared Error:',mean_squared_error(y_test, predictions))
print('Root Mean Squared Error:',math.sqrt(mean_squared_error(y_test, pre
```

```
Mean Absolute Error: 7.228148653430853
Mean Squared Error: 79.81305165097487
Root Mean Squared Error: 8.933815066978656
```

Residuals

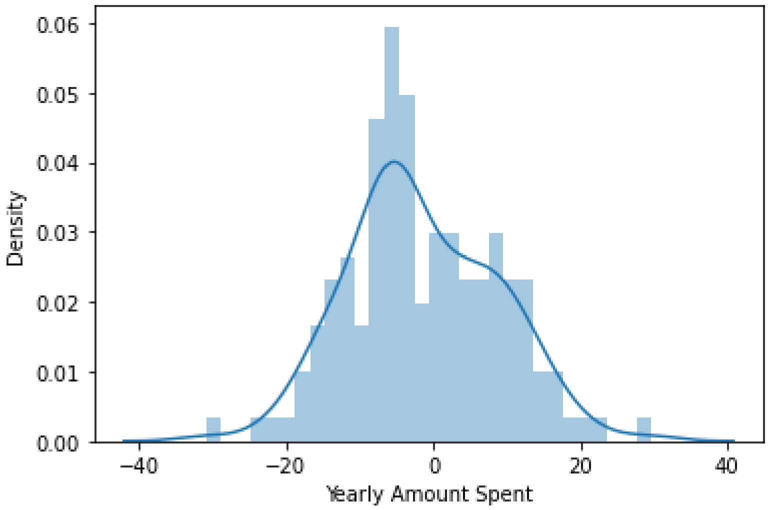
Distribution plot of the residuals of the model's predictions. They should be normally distributed.

In [37]:
```
residuals = y_test-predictions
sns.distplot(residuals, bins=30)
```

```
/opt/homebrew/anaconda3/lib/python3.9/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[37]:

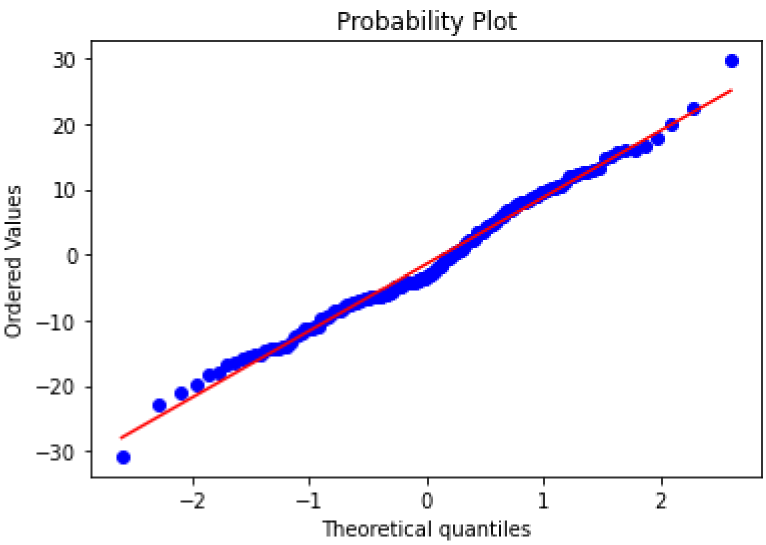
```
<AxesSubplot:xlabel='Yearly Amount Spent', ylabel='Density'>
```



In [43]:

```python
import pylab
import scipy.stats as stats

stats.probplot(residuals, dist="norm", plot=pylab)
pylab.show()
```



Conclusion

According to the model, the most significant factor for clients is not the time spent on the app or website, but their length of membership. However, of the two predictors, the app has the strongest influence by far. In fact, the time spent on the desktop website does not seem to have any correlation at all! In other words, according to the data, the amount of time that the customer spends on the desktop website has almost nothing to do with the amount of money they will spend.

This could mean that the desktop website needs more work to make its visitors buy more. Secondly, it could mean that people tend to be more influenced by mobile applications of online stores than by desktop websites. So maybe efforts should be directed towards taking advantage of this fact. Indeed, the interpretation of this information requires expertise in the online marketing sphere. Our analysis and our model, however, does a very good job in weighting the predictors importance.

In [ ]: