

**MSDA Capstone Project:**

**Predictive Modeling for Marketing Effectiveness with Multiple Linear Regression**

David Harvell

Master of Science, Data Analytics Program

Data Analytics Graduate Capstone – D214

January 2022

## Section A – Research

### A1. Summary of the Research Question

Our base research question is “How much does the frequency of ads impact the probability of purchase”.

We will investigate digital/online advertising for this research. Digital advertising encompasses any type of paid communication on the internet. There are many advantages to digital advertising, including user targeting, retargeting, and the ability to have a more accurate estimate of the return on investment (The Neeva Team, 2021).

User targeting is the act of using information about the consumer such as purchase history, websites visited, or recent searches when serving advertising content. Retargeting is the act of targeting consumers that have previously interacted with ads or shown any other type of interest. Both tactics increase the value of the advertising by focusing on consumers that are most likely to purchase. When a business takes advantage of one or both, it can reduce the cost of advertising and either provide a savings or an opportunity to push out more ads across more avenues.

The amount of money involved in online advertising is staggering. The current value of Google is listed at over \$1 trillion dollars, and 95% of their revenue is from advertising spend (Rudolph, 2014).

However, even with the massive amount of money involved in online advertising, there is still doubt to its effectiveness. Even if online advertising works, it is difficult to say that it works better than the more traditional route of television commercials. The overwhelming

majority of memorable advertising campaigns still come from television. While the internet has “viral” videos and memes that reach peak popularity, rarely (if ever) do these originate from an ad campaign.

## A2. Justification and Context for the Research Question

A common question shared by companies that sell products and services is “does marketing work, and if so, how much should be spent on it?”. This project offers a chance to create a model that answer those questions. An answer to this question can help grow products and plan budgets.

Digital advertising current accounts for about half of all global advertising spend, totaling \$398 billion globally. The largest sector in digital advertising is paid search. This is classified as the search results that appear at the top of the search engine results, above the organic results (The Neeva Team, 2021).

Although paid search is the largest sector fiscally, there are many more streams of digital advertising that can account for more ads at a lower cost. These include banner ads on the top and sides of pages, as well as email advertising.

Banner ads come with many risks. An estimated 72% of advertising campaigns had at least some impressions that were delivered alongside inappropriate content (Rudolph, 2014). This can create an association that devalues a brand or ties unintentional ideologies with their company.

In addition to delivering messages adjacent to inappropriate content, there are many banners that are never seen by humans. This can occur because there are bots that are

scanning pages, and sometimes even creating interaction (“clicks”) with the ads (Imseng, 2018). Of an average 1700 ads shown per person per month, only an estimated half of those is viewed by end-users.

Yet another issue with banner advertising is fraud that is perpetrated by the website selling the advertising. There are instances of websites stacking dozens or even hundreds of banners on top of one another to inflate the view and click count (Imseng, 2018). This can be achieved through the CSS and HTML code so that the end-user only sees a single banner on their end, but there are others stacked underneath it.

Other avenues of digital advertising include email campaigns and social media. Email is a technology that has been around for decades, whereas social media advertising just started seeing a boom in the last 5 years. As of 2018, email was a clear winner between the two routes, acquiring customers at nearly 40 times the rate of Facebook and Twitter. This can be attributed in part to the wide-spread adoption of email (Aufreiter et al., 2018).

There were many reasons why email is considered more successful. Over 90% of all U.S. consumers still use email daily, and the prompted purchases triple that of social media. Even when social media is successful, the value of the order is almost 20% less than the average email order (Aufreiter et al., 2018).

Due to the success of email campaigns and the percentage of emails that are viewed on mobile devices, it can benefit a company to optimize their website for mobile viewing. Almost 45% of advertising emails are viewed on mobile devices. According to Google, over 60% of users are unlikely to return to a website if they encounter issues browsing, and almost half will

visit a competing website instead (Aufreiter et al., 2018). Not surprisingly, this can affect the success of an email marketing campaign greatly.

Even though there is a substantial portion of advertising budget currently drives digital campaigns, there is still some doubt as to the effectiveness when compared to traditional campaigns such as television. Until recently, it was difficult to think of any brands that were truly built using digital advertising. With the growth and maturing state of advertising on social media, there are starting to be some instances of brands going “viral” – but they don’t appear to hit the heights that non-advertising memes and videos can achieve.

Television is still a staple throughout households in the U.S. The penetration is currently 96.5% and has continued to see increases in recent years. U.S. social media is estimated under 80%, although the exact number could be as low as 70%. The mindset of consumers during television viewing could also be of substance. People are primarily watching television while at home in a relaxed state. This gives the messaging a higher likelihood of being received. Social media is commonly consumed throughout the day and could be competing for attention with work, studies, or other duties. Most traditional companies are adding social media as an avenue to expand advertising, as opposed to replacing their traditional television commercials (Consumer Tech, 2021).

### A3. Hypothesis

Our assumption is that a higher frequency of ads will impact the probability of purchase with statistical significance. Because of this, our null hypothesis will be that total ad count has

NO impact on the probability of conversion. We will review a marketing A/B dataset that has measured counts of ads along with conversions (getting a consumer to purchase).

There are some inherent issues that we can expect with the model. To begin with, the click/interaction rate for online marketing is extremely low. Only an estimated one in one thousand ads gets clicked. With this low of a rate, it is somewhat expected that the correlation is very low (Rudolph, 2014).

Determining the “success” of an online marketing campaign isn’t always directly measured as clicks/purchases. It isn’t uncommon to have a goal of building the brand recognition or identity. This doesn’t seek immediate purchases, but rather works to build a foundation for future sales.

Campaigns that seek immediate purchases (calls to action) are commonly discounted and rejected when presented in too obvious of a manner. Rather, using ads that create memories or experiences are more likely to influence purchasing behavior in the long-term (Hollis, 2011). Advertising companies work to entertain or inform with “storytelling”, and the focus usually drifts from directly promoting the brand. This is also working to seed ideas and favorability for long-term growth (Duffy, 2016).

## Section B – Data Collection

### B1. Data Source

Our dataset is from Kaggle. Kaggle is a website that allows users to publish and search datasets, and then share models or other finding pertaining to data analytics and data science. Datasets on Kaggle are free for download and open to public use. The dataset can be found at <https://www.kaggle.com/favio vaz/marketing-ab-testing> and was uploaded in October of 2021 by the user FavioVazquez. It is provided in the form of a cleansed comma separated value (CSV) file and is approximately 22MB in size.

The data contains records that track a marketing A/B experiment. An A/B experiment, when using the medium of a website, is conducted by having two versions of the website and route a portion of the traffic to the alternative version. User interaction can then be tracked to determine the success of the alternate page.

Our dataset is using a slightly alternative version of the standard website A/B test – instead of tracking which version of the webpage performs better, we are using an alternate version that will show a version without an ad to create a control group. The dataset is split into results that showed a banner ad, and results that instead showed a public service announcement. Because alternate versions of a website can jeopardize the search engine rankings for your website, Google provides a method for web developers to indicate an A/B test is in progress (A/B Testing, n.d.).

The goal listed for this data is analyzing groups of users, determining if ads were successful, calculating how much a company could make from ads, and if the difference between the groups is statistically significant.

## B2. Possible Issues with Data Source

Because this dataset is provided by a user on the internet that we can't confidently contact, there is no guaranteeing that the data has been properly collected. We also will not be able to request supplementary information, such as the following –

- A. Are there any demographic attributes we can attach to the data?
- B. During what date range was this data collected?
- C. What type of product(s) are being promoted by the ads?
- D. Was the ad always located in the same position on the webpage?
- E. What time range was considered for the conversion? If we looked out further, we might see residual impact of the campaign or weeks or even months afterwards.

The data is listed under the license of CC0: Public Domain. This means there is no copyright, and anyone is allowed to copy, modify, distribute, or perform work on the dataset (even for commercial purposes) without asking permission.

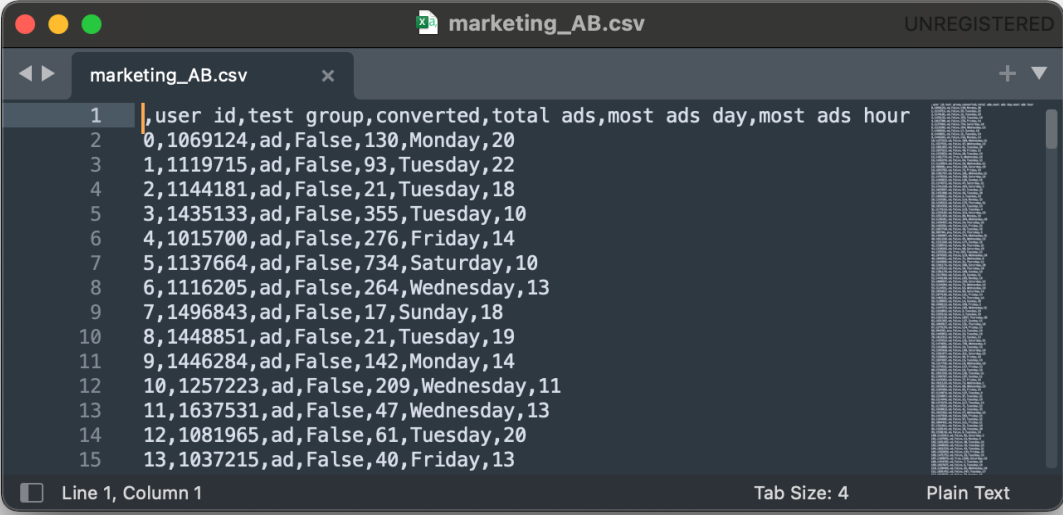


## Section C – Data Extraction and Preparation

### C1. Basic Data Loading/Review

The dataset is downloaded as a 22MB zip (archive) file that contains a single comma separated value (CSV) file. The CSV was extracted from the zip file using the built-in functionality in Mac OSX 12.1. This is accessed via the right-click “Extract” option.

CSV files can be accessed in a variety of ways. The simplest, straight forward is by opening with a text editor. Below is a screenshot of the first few lines of the file being viewed with Sublime text editor. Sublime was chosen because it has several positive reviews and is a free application.



```
1 ,user id,test group,converted,total ads,most ads day,most ads hour
2 0,1069124,ad,False,130,Monday,20
3 1,1119715,ad,False,93,Tuesday,22
4 2,1144181,ad,False,21,Tuesday,18
5 3,1435133,ad,False,355,Tuesday,10
6 4,1015700,ad,False,276,Friday,14
7 5,1137664,ad,False,734,Saturday,10
8 6,1116205,ad,False,264,Wednesday,13
9 7,1496843,ad,False,17,Sunday,18
10 8,1448851,ad,False,21,Tuesday,19
11 9,1446284,ad,False,142,Monday,14
12 10,1257223,ad,False,209,Wednesday,11
13 11,1637531,ad,False,47,Wednesday,13
14 12,1081965,ad,False,61,Tuesday,20
15 13,1037215,ad,False,40,Friday,13
```

The second most accessible way to view CSV files is with a spreadsheet software such as Excel or Google Sheets. If the CSV is properly formed, these types of software will open the file

as a table, with each comma indicating a new column and the first row corresponding to column headers. On a system with Microsoft Excel, it is commonly set to the default application for CSVs.

Possible Data Loss Some features might be lost if you save this workbook in the comma-d...

	A	B	C	D	E	F	G	H	I	J	K
1		user id	test group	converted	total ads	most ads day	most ads hour				
2	0	1069124	ad	FALSE	130	Monday	20				
3	1	1119715	ad	FALSE	93	Tuesday	22				
4	2	1144181	ad	FALSE	21	Tuesday	18				
5	3	1435133	ad	FALSE	355	Tuesday	10				
6	4	1015700	ad	FALSE	276	Friday	14				
7	5	1137664	ad	FALSE	734	Saturday	10				
8	6	1116205	ad	FALSE	264	Wednesday	13				
9	7	1496843	ad	FALSE	17	Sunday	18				
10	8	1448851	ad	FALSE	21	Tuesday	19				
11	9	1446284	ad	FALSE	142	Monday	14				
12	10	1257223	ad	FALSE	209	Wednesday	11				
13	11	1637531	ad	FALSE	47	Wednesday	13				
14	12	1081965	ad	FALSE	61	Tuesday	20				
15	13	1037215	ad	FALSE	40	Friday	13				
16	14	1535652	ad	FALSE	20	Tuesday	19				
17	15	1461774	ad	TRUE	9	Wednesday	18				

One item of note – If your data is over 1 million records, Excel cannot handle the file properly and can result in a corrupt file. In these instances, it is preferred to use a text editor or a more data analysis-oriented approach (covered in the next section).

Although these tools don't always allow for the most in-depth analysis, they do provide an extremely accessible way to quickly review the data. A user with a few basic tools and little

technical knowledge can open the data and perform actions such as validate, check record counts, visually check for empty values, etc.

## C2. Detailed Data Loading

For in-depth analysis we will be using Python, and more specifically, the Pandas library. Pandas allow for easy loading, manipulation, and exploration of tabular data. Instead of installing these tools on our computer, we are utilizing the Colab website from Google Labs. It provides a fully featured installation of Python and commonly used libraries through a web interface. To enable the use of Colab, we will feed the data in via Google Drive. The following code links to Google Drive, imports some basic libraries for exploration, and loads the data.

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

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

```
df = pd.read_csv('/content/drive/My Drive/School/Datasets/marketing_AB.csv')
```

Numpy is useful for series/array analysis. Our specific use-case will be transforming data into Pandas dataframes. Pandas allows quick work with tabular data with the dataframe type, which stores a table of data. Matplotlib will allow graphing and charting for data exploration, and Seaborn compliments it with more aesthetically pleasing visuals.

One inherent advantage of these tools is the speed at which you can begin analysis.

With very basic Python knowledge, you can perform checks such as checking for empty values within a few seconds – even when working with millions of records. Quick transformations are also possible. Trying to manipulate data over 100K records inside of Excel can quickly lead to a processor that is pushed to the limit for many minutes during processing.

Although Python and pandas are well-suited to this task, there are still a few things that could be considered disadvantages. The first is learning curve. Although most people are familiar with spreadsheet applications, not many have learned to work with a programming or scripting language. Just the idea can be intimidating for many. Beyond the learning curve of the language itself is the setup. Getting libraries and compilers installed can sometimes turn into multi-hour tasks as you search the internet for answers to vague errors. As mentioned earlier, we are avoiding this problem by using an environment that is served through Google Colab.

Another disadvantage of Python is sometimes speed. Our analysis isn't too heavy of a task, with 22MB of data, but Python can be slow when using bigger datasets. This is because Python is an interpreted language – meaning that specialized code is generated (compiled) for the program. Instead, the code is processed line by line and ran through an interpreter.

### C3. Detailed Data Review

Our initial review after loading will check the columns, datatypes, null values, record count, distribution of values, and unique values. We will now run through each of these checks

and include the findings. If we discover values that need to be corrected or cleansed, this would be the stage where that happens.

- A. Check the first few records and visually assess the data. There are 7 columns (or variables) per record. The first two seem to be id columns and are inconsequential for our analysis.

We will remove them in a later step

```
df.head()
```

	Unnamed: 0	user id	test group	converted	total ads	most ads day	most ads hour
0	0	1069124	ad	False	130	Monday	20
1	1	1119715	ad	False	93	Tuesday	22
2	2	1144181	ad	False	21	Tuesday	18
3	3	1435133	ad	False	355	Tuesday	10
4	4	1015700	ad	False	276	Friday	14

- B. Use the info() command in Pandas to view non-null data counts and datatypes. We are working with 1 Boolean value (True/False), 4 integers (whole numbers), and 2 objects (in this case, strings of characters or words).

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 588101 entries, 0 to 588100
Data columns (total 7 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Unnamed: 0      588101 non-null int64
1   user id         588101 non-null int64
2   test group      588101 non-null object
3   converted        588101 non-null bool
4   total ads       588101 non-null int64
5   most ads day    588101 non-null object
6   most ads hour   588101 non-null int64
dtypes: bool(1), int64(4), object(2)
memory usage: 27.5+ MB
```

- C. Since nothing new was discovered that indicates we will need the first two columns, we will now remove them to reduce the dataset size. This can benefit us in multiple ways – both allowing faster process of the dataset and making it simpler to visually comprehend.

```
df.drop(columns = ['Unnamed: 0', 'user id'], inplace = True)
```

- D. Now we utilize the describe() command in Pandas. It will provide statistical information about the numeric/continuous values in the dataset, such as min, max, mean, and standard deviation.

```
df.describe()
```

	total ads	most ads hour
<b>count</b>	588101.000000	588101.000000
<b>mean</b>	24.820876	14.469061
<b>std</b>	43.715181	4.834634
<b>min</b>	1.000000	0.000000
<b>25%</b>	4.000000	11.000000
<b>50%</b>	13.000000	14.000000
<b>75%</b>	27.000000	18.000000
<b>max</b>	2065.000000	23.000000

- E. Next we will use the nunique() command. This will show us the distinct value counts for each of the columns. This shows us some expected values – only 2 groups, 2 values for the Boolean, and 7 values for the day – but also some new information. Apparently, ads were shown at all hours of the day (indicated by the 24 values in the most ads hour column), and there are over 800 different values for the total number of ads.

```
df.nunique()
```

```
test group      2
converted       2
total ads      807
most ads day     7
most ads hour   24
dtype: int64
```

- F. Although the `info()` results indicated that the values were non-null, we can confirm by using a combination of functions – `isna()` and `sum()`. This confirms the earlier results and shows there are no columns with null values.

```
df.isna().sum()
```

```
test group      0
converted       0
total ads       0
most ads day     0
most ads hour    0
dtype: int64
```

- G. Next, to get a “feel” for the data, we can display a grouping of the categorical values to see how many records fall into each group of categories. The results are generally what we would expect – there are fewer conversions the not, and a higher number of conversions when an ad was shown instead of a public service announcement. This result is produced by using `value_counts()` on a subset of the dataframe, containing only the categorical columns.

```
df[['test group', 'converted', 'most ads day']].value_counts()
```

test group	converted	most ads day	
ad	False	Friday	86810
		Monday	80793
		Sunday	80305
		Thursday	77366
		Saturday	77123
		Wednesday	75455
		Tuesday	72302
psa	False	Thursday	3826
		Friday	3741
		Wednesday	3435
		Monday	3423
		Sunday	2996
		Tuesday	2865
		Saturday	2818
ad	True	Monday	2778
		Tuesday	2270
		Sunday	2027
		Friday	1995
		Wednesday	1963
		Thursday	1711
		Saturday	1679
psa	True	Monday	79
		Thursday	79
		Sunday	63
		Friday	62
		Wednesday	55
		Tuesday	42
		Saturday	40

dtype: int64

H. Finally, we review the `value_counts()` for the individual columns. Again, this will give us and idea of distributions that might be helpful in later analysis.

```
df['test group'].value_counts()
```

```
ad      564577
psa     23524
Name: test group, dtype: int64
```

```
df['converted'].value_counts()
```

```
False    573258
True      14843
Name: converted, dtype: int64
```

```
df['most ads day'].value_counts()
```

```
Friday      92608
Monday      87073
Sunday      85391
Thursday    82982
Saturday    81660
Wednesday  80908
Tuesday     77479
Name: most ads day, dtype: int64
```



This dataset appears to be cleansed already. Although we performed some basic analysis, it was just enough to explore the data for potential issues. In the next section we will perform the more in-depth analysis and predictive modeling. For that purpose, we will also transform some of the values to make them more adaptable to analysis.

It is not uncommon for a dataset to need additional cleansing. Extra cleansing steps can involve removing blank values or filling with a substitution value, such as the mean or median. Other cleansing might involve removing outliers.

## Section D – Data Analysis

### D1. Overview

We will transform and perform basic analysis, followed by the creation of our predictive model and review of its performance. We will employ Multiple Linear Regression (MLR) models for predictive purposes, and then measure the success with R-squared and P-values. We will perform the analysis in Jupyter Notebooks. This tool allows for easy ad-hoc coding and quick pivoting when necessary.

### D2. Basic Analysis and Transformations

To begin, we will create a new value to represent the rate of conversions. This can be achieved using the `groupby()` function in Pandas. Below is a small example of the function, returning a record for each distinct “most ads hour” value and totaling other fields.

```
df.groupby('total ads').sum().head()
```

	converted	most ads hour
total ads		
1	89	815721
2	94	571967
3	81	415246
4	84	341284
5	101	425789

Using this logic, we can create an aggregated dataset that contains sums and counts of each numeric value, and then use those values to calculate a new column that contains the percentage of records that were converted for each distinct count of total ads shown.

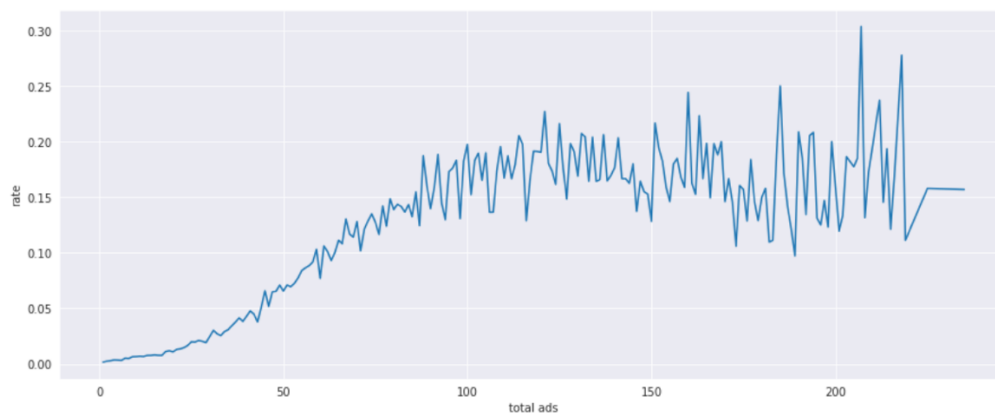
```
dfAgg = df.groupby('total ads')['converted'].agg(['sum', 'size'])
dfAgg['rate'] = dfAgg['sum'] / dfAgg['size']
dfAgg = dfAgg[dfAgg['size'] >= 50]
dfAgg.tail(100)
```

	sum	size	rate
<b>total ads</b>			
117	44	266	0.165414
118	54	282	0.191489
119	48	251	0.191235
120	48	252	0.190476
121	62	273	0.227106
...	...	...	...
216	11	66	0.166667
218	15	54	0.277778
219	6	54	0.111111
225	9	57	0.157895
235	8	51	0.156863

100 rows × 3 columns

Common sense would imply that the conversion rate should go up as people are shown more ads. We can quickly check with a chart of the rate vs total ads though the Matplotlib and Seaborn libraries. We are also setting the size and style of the grids in this block of code.

```
sns.set_style('darkgrid')
plt.figure(figsize=(15, 6))
sns.lineplot(data=dfAgg, x="total ads", y="rate");
```



Next, we can compare this data to the conversion rate when just PSAs were shown.

Ideally, it will show less consistent growth as the number of PSAs increase.

```
dfAgg = df.groupby(['total ads', 'test group'])['converted'].agg(['sum', 'size'])
dfAgg['rate'] = dfAgg['sum'] / dfAgg['size']
dfAgg = dfAgg[dfAgg['size'] >= 20]
dfAgg.tail(100)
```

		sum	size	rate
total ads	test group			
197	ad	10	68	0.147059
198	ad	8	59	0.135593
199	ad	12	57	0.210526
200	ad	7	46	0.152174
201	ad	8	63	0.126984
...	...	...	...	...
323	ad	2	23	0.086957
326	ad	6	23	0.260870
332	ad	4	21	0.190476
334	ad	5	20	0.250000
341	ad	5	21	0.238095

100 rows × 5 columns

```
plt.figure(figsize=(15, 6))
sns.lineplot(data=dfAgg, x="total ads", y="rate", hue='test group');
```



Indeed, there appears to be less consistent growth when just looking at the records that included a PSA. While there is some growth, it is very inconsistent.

For Multiple Linear Regression models, we need to convert all qualitative values into quantitative. In other words, we will replace text descriptions with numeric representations – such as 1 through 7 to represent Sunday through Saturday. There are numerous ways to accomplish this inside of Python, but we will begin by manually replacing with the NumPy `select()` function. Another common method would be `get_dummies()`.

```
conditions = [(df['test group'] == 'psa')
              , (df['test group'] == 'ad')]
values = [0, 1]
df['group_d'] = np.select(conditions, values)

conditions = [(df['converted'] == False)
              , (df['converted'] == True)]
values = [0, 1]
df['converted_d'] = np.select(conditions, values)

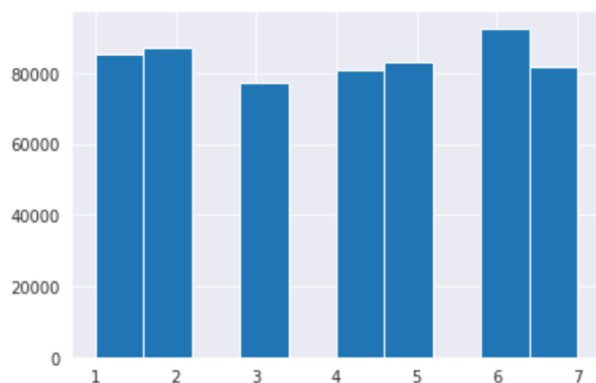
conditions = [(df['most ads day'] == 'Sunday')
              , (df['most ads day'] == 'Monday')
              , (df['most ads day'] == 'Tuesday')
              , (df['most ads day'] == 'Wednesday')
              , (df['most ads day'] == 'Thursday')
              , (df['most ads day'] == 'Friday')
              , (df['most ads day'] == 'Saturday')]
values = [1, 2, 3, 4, 5, 6, 7]
df['day_d'] = np.select(conditions, values)
```

Now that the numeric representations have been created, we can remove the original columns.

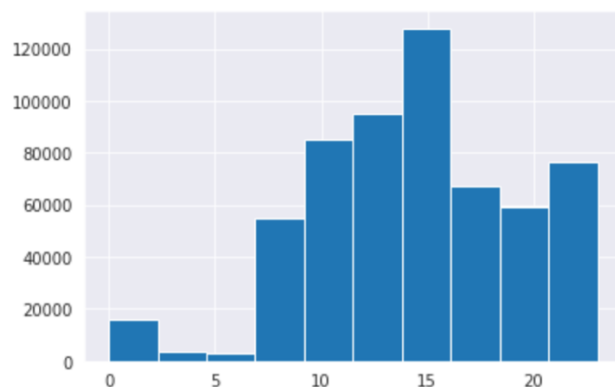
```
df.drop(columns = ['test group', 'converted', 'most ads day'], inplace = True)
```

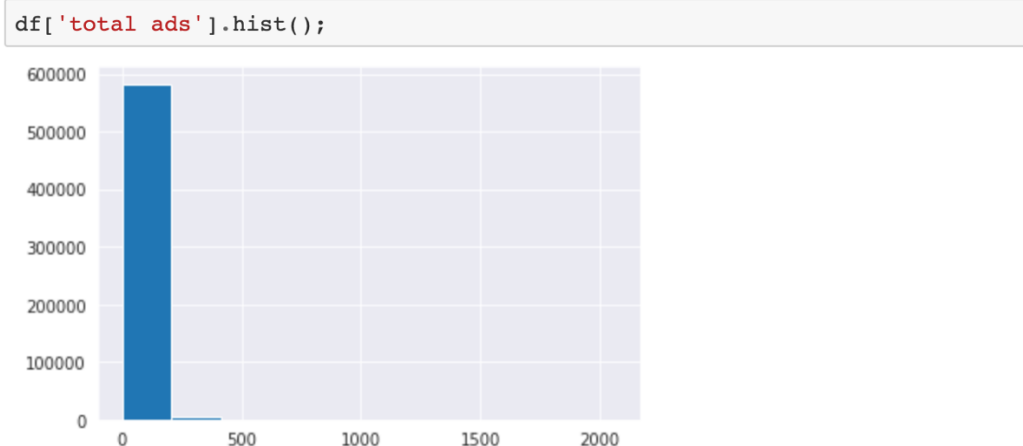
Next, we can review histograms for the variables to analyze the distributions. Part of this was done early with `value_counts()`, but this will put a graphical representation on top for added insights.

```
df['day_d'].hist();
```



```
df['most ads hour'].hist();
```





There are no surprises revealed by the histograms, but it does reinforce some findings.

The day of week is an even distribution, indicating that ads were shown an equal amount on every day of the week.

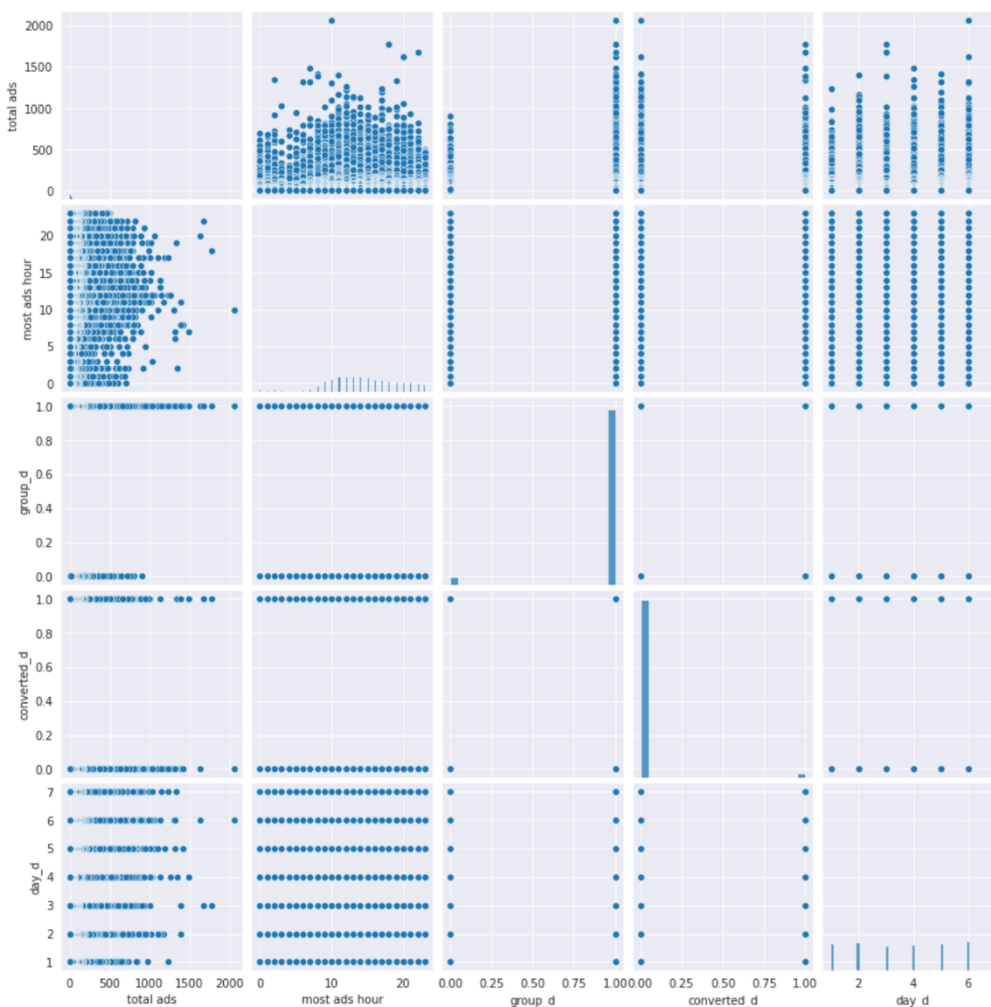
The distribution for “most ads hour” indicates that users were online less during the early morning hours. This is likely due to most of the population being asleep during these hours.

Finally, the number of ads shown is skewed heavily towards values between 0 and 250. The large range on the X-axis means there is still data with as many as 2K ads shown, but it is a much smaller portion of the records.

Next, we can run a pair plot. This will create a two-axis chart for every combination of numeric field in the dataset. The type of chart used will be dynamically chosen based on the data in the columns. In some instances, pair plots can quickly demonstrate correlation in a visual manner.

```
plt.figure(figsize=(20, 12))
sns.pairplot(df);
```

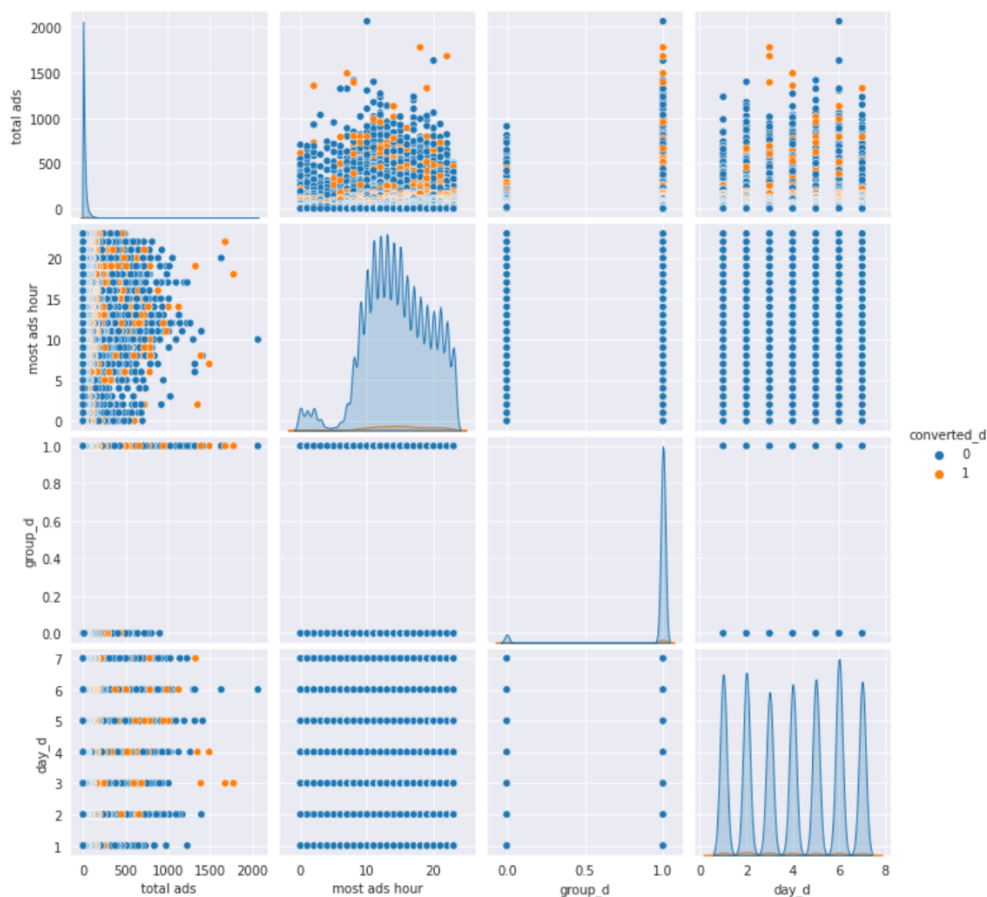
<Figure size 1440x864 with 0 Axes>



Unfortunately, this pair plot doesn't reveal any new insights. Another optional parameter on the pair plot function is for hue. This will allow the report to color points/lines/areas/etc. based on values of a specified column. We will run another pair plot with the hue set to vary based on conversion. This will allow any trends specific to conversion (or non-conversion) to be quickly viewed.



```
sns.pairplot(df, hue='converted_d');
```

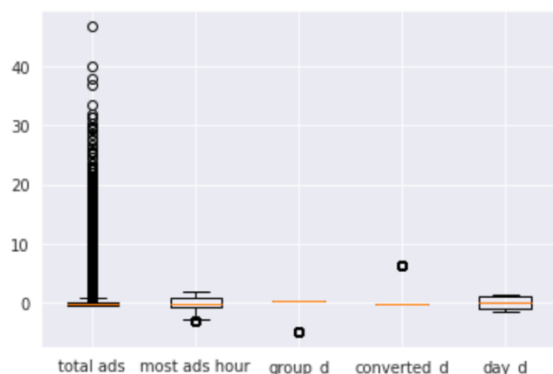


This does indicate a few visual trends –

- There were rarely any conversions on the day corresponding to 1 (Sunday – indicated by the “day\_d” vs “total ads” charts).
- Most of the conversions were present on the group that were presented ads (indicated by the “group\_d” vs “total ads” charts).

Finally, we will run a box plot to check for extreme outliers. Box plots will show the min, max, 25<sup>th</sup> percentile, 75<sup>th</sup> percentile, and outliers.

```
boxplot = plt.boxplot(dfs, labels=df.columns)
```



There are many values listed at outliers for “total ads”. This could indicate data that needs to be removed, and we will need to remember this in case it is necessary later in analysis. Although some of the other columns show outliers, it is only because the vast majority of values are marked as non-conversion or PSA. We cannot remove the converted or ad data and still test for prediction of conversions. Note: the data was standardized before running our box plot. The steps for standardization will be covered in the next step.

### D3. Predictive Model

Most of the data preparation has been handled, but we will standardize the data for model creation. This will prevent variable that have larger ranges from inherently receiving a higher correlation. We will use the SciKitLearn library for standardization. This library has numerous functions that aid in AI/ML procedures.

```
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
```

```
x = df.values
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
dfs = pd.DataFrame(x_scaled, columns=df.columns)
dfs.head()
```

	total ads	most ads hour	group_d	converted_d	day_d
0	0.062500	0.869565	1.0	0.0	0.166667
1	0.044574	0.956522	1.0	0.0	0.333333
2	0.009690	0.782609	1.0	0.0	0.333333
3	0.171512	0.434783	1.0	0.0	0.333333
4	0.133236	0.608696	1.0	0.0	0.833333

We are using the min/max scaler. It uses the minimum and maximum values for each column to scale the values between a range of 0 and 1. Now we can use the `corr()` function to test the base correlation between each variable.

```
dfs.corr()
```

	total ads	most ads hour	group_d	converted_d	day_d
<b>total ads</b>	1.000000	-0.010837	0.000279	0.217419	0.007803
<b>most ads hour</b>	-0.010837	1.000000	0.006930	0.019674	0.023186
<b>group_d</b>	0.000279	0.006930	1.000000	0.009611	-0.003370
<b>converted_d</b>	0.217419	0.019674	0.009611	1.000000	-0.018155
<b>day_d</b>	0.007803	0.023186	-0.003370	-0.018155	1.000000

We can also use a heatmap plot from Seaborn to visually represent the correlation matrix.

```
plt.figure(figsize=(16,10))
sns.heatmap(dfs.corr(), annot=True, cmap='YlGnBu');
```



This indicates a low correlation between the total number of ads and conversions.

These results are in line with the earlier chart that showed an upward trend of conversions as the total ads increased.

Now we will create the model using the StatsModel library. The model will take the array of dependent values and the dataframe of independent variables as parameters.

```
import statsmodels.api as sm

# Move the dependent variable to the end for ease of handling
cols = list(dfs)
cols.insert(0, cols.pop(cols.index('converted_d')))
dfs = dfs.loc[:, cols]

y = dfs['converted_d']
X = dfs.iloc[:, 1:]
variables = X.columns

Xc = sm.add_constant(X)
linear_regression = sm.OLS(y, Xc)
fitted_model = linear_regression.fit()
fitted_model.summary()
```

Fitting the model is relatively simple, and now we can review the summary information.

#### OLS Regression Results

<b>Dep. Variable:</b>	converted_d	<b>R-squared:</b>	0.048
<b>Model:</b>	OLS	<b>Adj. R-squared:</b>	0.048
<b>Method:</b>	Least Squares	<b>F-statistic:</b>	7455.
<b>Date:</b>	Thu, 06 Jan 2022	<b>Prob (F-statistic):</b>	0.00
<b>Time:</b>	12:34:54	<b>Log-Likelihood:</b>	2.6950e+05
<b>No. Observations:</b>	588101	<b>AIC:</b>	-5.390e+05
<b>Df Residuals:</b>	588096	<b>BIC:</b>	-5.389e+05
<b>Df Model:</b>	4		
<b>Covariance Type:</b>	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
<b>const</b>	-0.0063	0.001	-5.257	0.000	-0.009	-0.004
<b>total ads</b>	1.6131	0.009	171.206	0.000	1.595	1.632
<b>most ads hour</b>	0.0167	0.001	17.634	0.000	0.015	0.019
<b>group_d</b>	0.0075	0.001	7.330	0.000	0.005	0.009
<b>day_d</b>	-0.0095	0.001	-15.987	0.000	-0.011	-0.008

<b>Omnibus:</b>	621013.207	<b>Durbin-Watson:</b>	1.994
<b>Prob(Omnibus):</b>	0.000	<b>Jarque-Bera (JB):</b>	28597333.235
<b>Skew:</b>	5.614	<b>Prob(JB):</b>	0.00
<b>Kurtosis:</b>	35.264	<b>Cond. No.</b>	76.2

The summary of this model provides in-depth information describing the model. We have the R-squared variable to indicate the amount of variance explained by the independent variables and P-values that let us know if the variables are statistically significant. The results will be discussed in greater detail in the next section.

Before wrapping up, we can do a quick check of the eigenvectors and corresponding eigenvalues in hopes that we can tighten up the model. First, we will check the eigenvectors for the model.

```
corr = np.corrcoef(X, rowvar=0)
eigenvalues, eigenvectors = np.linalg.eig(corr)
print(eigenvalues)

[0.96932056  1.02374209  1.00677064  1.00016672]
```

Although none of these are extremely out of line, the outlier appears to be the first set at index 0. Investigating it might provide some insight.

```
print(eigenvectors[:, 0])

[-0.38828327 -0.65099726  0.21811312  0.61470752]
```

```
variables

Index(['total ads', 'most ads hour', 'group_d', 'day_d'], dtype='object')
```

Looking at the eigenvalues in index 0 tells us that the second and fourth variables are outliers when compared to the set as a whole. These values correspond to the “most ads hour” and “day\_d” variables. This makes sense, that the day and hour would have less impact when compared to the group (PSA vs ads) or the total number of ads. Now we can remove those variables and fit a new model.

```
dfs_rev = dfs.drop(columns=['most ads hour', 'day_d'])

y = dfs_rev['converted_d']
X = dfs_rev.iloc[:, 1:]
variables = X.columns
Xc = sm.add_constant(X)
linear_regression = sm.OLS(y, Xc)
fitted_model = linear_regression.fit()
fitted_model.summary()
```

Finally, we can fit the new model and produce summary information.

#### OLS Regression Results

<b>Dep. Variable:</b>	converted_d	<b>R-squared:</b>	0.047
<b>Model:</b>	OLS	<b>Adj. R-squared:</b>	0.047
<b>Method:</b>	Least Squares	<b>F-statistic:</b>	1.462e+04
<b>Date:</b>	Thu, 06 Jan 2022	<b>Prob (F-statistic):</b>	0.00
<b>Time:</b>	12:34:54	<b>Log-Likelihood:</b>	2.6923e+05
<b>No. Observations:</b>	588101	<b>AIC:</b>	-5.384e+05
<b>Df Residuals:</b>	588098	<b>BIC:</b>	-5.384e+05
<b>Df Model:</b>	2		
<b>Covariance Type:</b>	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
<b>const</b>	-0.0007	0.001	-0.679	0.497	-0.003	0.001
<b>total ads</b>	1.6101	0.009	170.826	0.000	1.592	1.629
<b>group_d</b>	0.0076	0.001	7.503	0.000	0.006	0.010

<b>Omnibus:</b>	621599.509	<b>Durbin-Watson:</b>	1.994
<b>Prob(Omnibus):</b>	0.000	<b>Jarque-Bera (JB):</b>	28701568.426
<b>Skew:</b>	5.622	<b>Prob(JB):</b>	0.00
<b>Kurtosis:</b>	35.324	<b>Cond. No.</b>	65.8

## Section E – Data Summary and Implications

The summary results from the OLS model provide us with all the values we need to conclude our analysis.

The R-Squared value of 0.047 means that the independent variables only explain 4.7% of the variance in the dependent variable. Unfortunately, this is a very low value for  $R^2$  and would not indicate a model that will produce very accurate results.

Plugging our degrees of freedom and the F value of 1.462e+04 into a probability calculator arrive at a P-value of less than 0.001 that the F value exceeds 1.462e+04. This infers that we should reject our null hypothesis and conclude that the variables do impact the dependent variable of conversion in a statistically significant manner. This is further reinforced with the P-values of 0 (or essentially 0) for the independent variables of “total ads” and “group\_d”.

Since we are rejecting our null hypothesis, that means we should accept the alternative hypothesis that the number of ads and being shown ads vs PSAs impact the success of conversion in a statistically significant manner.

These findings align with the early investigation of the data. We could see an upward trend in conversion rate when more ads were shown, but the conversion rate is so small that accurate prediction seems unlikely.



## Section F – Sources

### F1. Citations / References

*A/B testing*. (n.d.). Optimizely. Retrieved January 15, 2022, from

<https://www.optimizely.com/optimization-glossary/ab-testing/>

Consumer Tech. (2021, June 30). *Why Do Big Companies Still Advertise On TV Instead Of Social*

*Media?* Forbes. <https://www.forbes.com/sites/quora/2019/03/01/why-do-big-companies-still-advertise-on-tv-instead-of-social-media/?sh=14a9ef3cdd41>

Duffy, M. (2016, December 22). *Does advertising even work any more?* Digiday.

<https://digiday.com/marketing/advertising-even-work-anymore/>

Hollis, N. (2011, August 31). *Why Good Advertising Works (Even When You Think It Doesn't)*.

The Atlantic. <https://www.theatlantic.com/business/archive/2011/08/why-good-advertising-works-even-when-you-think-it-doesnt/244252/>

Imseng, D. (2018, April 27). *"The effectiveness of online advertising is a delusion."* Medium.

<https://medium.com/stronger-content/the-effectiveness-of-online-advertising-is-a-delusion-75b3ea7027e9>

Rudolph, S. (2014, September 4). *How Effective Is Online Advertising (Infographic)*. Business 2

Community. <https://www.business2community.com/infographics/effective-online-advertising-0996804>

The Neeva Team. (2021, April 14). *How Does Online Advertising Work?* Neeva.

<https://neeva.com/learn/how-does-online-advertising-work>

Aufreiter, N., Boudet, J., & Weng, V. (2018, February 5). *Why marketers should keep sending you e-mails*. McKinsey & Company. <https://www.mckinsey.com/business-functions/marketing-and-sales/our-insights/why-marketers-should-keep-sending-you-emails>

## F2. Code References

*Controlling figure aesthetics — seaborn 0.11.2 documentation*. (n.d.). Seaborn Documentation.

Retrieved January 9, 2022, from <https://seaborn.pydata.org/tutorial/aesthetics.html>

GeeksforGeeks. (2019, February 5). *Drop rows from the dataframe based on certain condition applied on a column*. <https://www.geeksforgeeks.org/drop-rows-from-the-dataframe-based-on-certain-condition-applied-on-a-column/>

GeeksforGeeks. (2020, July 15). *Python - seaborn.pairplot() method*.

<https://www.geeksforgeeks.org/python-seaborn-pairplot-method/>

*How to get the P Value in a Variable from OLSResults in Python?* (2016, December 10). Stack Overflow. <https://stackoverflow.com/questions/41075098/how-to-get-the-p-value-in-a-variable-from-olsresults-in-python>

Luvsandorj, Z. (2021, December 16). *6 simple tips for prettier and customised plots in Seaborn (Python)*. Medium. <https://towardsdatascience.com/6-simple-tips-for-prettier-and-customised-plots-in-seaborn-python-22f02ecc2393>

Melih, A. (2019, November 23). *MultipleLinear Regression - Fish Weight Estimation*. Kaggle. <https://www.kaggle.com/akdagmelih/multiplelinear-regression-fish-weight-estimation>

*Normalize columns of pandas data frame.* (2014, October 16). Stack Overflow.

<https://stackoverflow.com/questions/26414913/normalize-columns-of-pandas-data-frame>

*Pandas DataFrame Groupby two columns and get counts.* (2013, July 16). Stack Overflow.

<https://stackoverflow.com/questions/17679089/pandas-dataframe-groupby-two-columns-and-get-counts>

*pandas.DataFrame.rename* — *pandas 1.3.5 documentation.* (n.d.). Pandas Documentation.

Retrieved January 9, 2022, from

<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.rename.html>

*pandas.get\_dummies* — *pandas 1.3.5 documentation.* (n.d.). Pandas Documentation. Retrieved

January 9, 2022, from

[https://pandas.pydata.org/docs/reference/api/pandas.get\\_dummies.html](https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html)

*seaborn.lineplot* — *seaborn 0.11.2 documentation.* (n.d.). Seaborn Documentation. Retrieved

January 9, 2022, from <https://seaborn.pydata.org/generated/seaborn.lineplot.html>

Statistics Solutions. (2021, August 11). *Assumptions of Multiple Linear Regression.*

<https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/assumptions-of-multiple-linear-regression/>

*Using categorical variables in statsmodels OLS class.* (2019, April 18). Stack Overflow.

<https://stackoverflow.com/questions/55738056/using-categorical-variables-in-statsmodels-ols-class>