Multiple Linear Regression Analysis on Superstore Sales Data

Nicole Reiswig

College of Information Technology, Western Governors University

Dr. William Sewell

April 24, 2024

Table of Contents

Research Question

This analysis aims to answer the research question: "Can a multiple linear regression model be constructed based solely on the research data?" The research question will be

answered utilizing multiple linear regression modeling. The goal of this analysis is to determine which variables in the given data set have a statistically significant impact on sales revenue. In determining which variables are most statistically significant in predicting sales revenue actionable insights can be made to increase sales revenue. The Null hypothesis is, "A predictive regression model cannot be constructed from the superstore sales data." The alternate hypothesis is, "A predictive regression model can be constructed from the superstore sales data."

Multiple linear regression is an appropriate technique to use for analyzing the research question because (Zach. 2020) multiple linear regression is the most commonly used technique in statistics and it is used to quantify the relationship between predictor variables and response variables. In this specific analysis, multiple linear analysis is used to quantify the relationship between the predictor variable Sales which is a continuous variable, and all the other variables in the superstore data set which are the independent variables. The multiple linear regression analysis provides insight into how strong a relationship is between the dependent and independent variables. Therefore the multiple linear regression analysis is a great technique for answering business questions with large data sets and finding the most significant factors affecting the business model.

Multiple linear regression has four assumptions according to (Zach, 2020) these assumptions are that there is a linear relationship between the x and y variables, residuals are independent, homoscedasticity, and normality are present. To meet these assumptions there are a couple of procedures that must take place. 1) A linear relationship between the x and y variables means there is a relationship in the form of a straight line. This is verified through bivariate graphs. 2) For the residuals to be independent, there must not be a correlation among the variables, they are independent and random. The residuals are verified through visualizations such as a correlation matrix and heatmap. 3) Homoscedasticity is when the error does not change across the values of the independent

variables (Middleton, 2023), homoscedasticity is verified through VIF the variance inflation factor. 4) Lastly, normality is when the residuals are normally distributed and this is checked through univariate graphs commonly a histogram.

The contribution of this study to the field of data analytics and the MSDA program is to create a predictive model that can estimate future sales of the superstore so that the company can be adequately prepared to meet future business needs. This study will utilize a multiple linear regression model to analyze the significant predictor variables for future sales. Dong, Chen, Y., Gu, A., Chen, J., Li, L., Chen, Q., Li, S., & Xun, Q., (2020) found that multiple linear regression analysis was successful in identifying positively correlated relationships in sales data specifically with the time and review variables. In previous studies utilizing multiple linear regression for sales revenue, there has been a positive correlation between sales revenue and review ratings (Dong et al., 2020). Analyzing the superstore sales data through multiple linear regression to answer the research question is crucial because it can be determined which factors have the greatest influence on sales revenue. In knowing which factors have the greatest impact both positive and negative smart business decisions can be made to maximize sales revenue in the upcoming years.

Data Collection

The data collected is owned by Bhanupratap Biswas, the author of the data is Bhanupratap Biswas a Kaggle expert. The dataset was collected from Kaggle.com and can be located at this web address:

https://www.kaggle.com/datasets/bhanupratapbiswas/superstore-sales.

This data set is updated on an annual basis. The superstore dataset has been licensed as ODC Public Domain Dedication and License which allows us to use this public data for this analysis (Open Knowledge, n.d.).

The data collection methodology was straightforward with no challenges once the dataset was selected. Finding the data set consisting of the minimum requirements for this study was a challenge and a disadvantage due to the time-consuming nature of the process. There are numerous amounts of data available to study. However, pages of data must be explored to find one that meets the criteria. The criteria for this analysis meant finding a dataset with a minimum of 7,000 rows of data and then also finding a dataset that could be utilized to answer the research question selected while also meeting the requirements of the chosen analysis technique. To overcome this challenge there were several topics of interest downloaded, cleaned, and explored before selecting this dataset to perform the analysis on. The advantage of this collection method was the ability to find a great dataset to work with for this analysis.

The superstore sales data consists of 9,800 rows, 18 columns, 4,922 unique Order IDs, 1,230 unique Order Dates, 1,320 unique Ship Dates, and 793 unique Customer IDs. The 18 columns in this data set are Row_ID, Order_ID, Order_Date, Ship_Date, Ship_Mode, Customer_ID, Customer_Name, Segment, Country, City, State, Postal_Code, Region, Product_ID, Category, Sub_Category, Product_Name and Sales.

Field	Data Type	Variable Type
Row_ID	Categorical	Independent
Order_ID	Categorical	Independent
Order_Date	Categorical	Independent
Ship_Date	Categorical	Independent
Ship_Mode	Categorical	Independent
Customer_ID	Categorical	Independent

Customer_Name	Categorical	Independent
Segment	Categorical	Independent
Country	Categorical	Independent
City	Categorical	Independent
State	Categorical	Independent
Postal_Code	Numeric	Independent
Region	Categorical	Independent
Product_ID	Categorical	Independent
Category	Categorical	Independent
Sub_Category	Categorical	Independent
Product_Name	Categorical	Independent
Sales	Numeric	Dependent

The study's limitations are that the dataset does not include certain factors that could have been valuable to the study such as review ratings. This data is not included because it was not collected or required in data collection. According to Chattopadhyay (2016) multiple linear regression analysis increases customer satisfaction and loyalty, predicts the number of purchases made, discovers the relationship between customer wait times and the number of complaints, and is used to estimate customers' needs. The delimitations of the study are that Order_ID, Customer_ID, Customer_Name, Country, and Product_ID will be removed from the dataset. These fields will be removed because they do not have statistically significant value in this analysis. Multiple Regression (n.d.) recommends that when using multiple linear regression the analysis should be limited to five or fewer independent variables to prevent multicollinearity.

Data Extraction and Preparation

The data analysis tools/techniques utilized are appropriate for this analysis. Python is an open-source programming language that in terms of usability is the better choice

because the syntax it uses is similar to other languages and therefore is more versatile (Ozgur, Colliau, Rogers, Hughes, & Myer-Tyson, 2017). Python is a famous data analysis language used by data scientists and is the preferred language because of its highly interactive nature and its scientific ecosystem libraries (Siddiqui, Alkadri, & Khan, 2017). According to Siddiqui (2017), SAS is an expensive solution and is between a programming language and a scripting language. R is an open-source programming language that is more difficult to learn but is great with data manipulation, statistics, and graphics functionality built-in (Siddiqui, 2017). Ozgur (2017) notes that R is geared more towards statistical roles, resembles SAS, does not rely on a computer science background or coding, and is open-sourced, while Python is geared more towards the coding aspects of jobs. Utilizing Python provides a few advantages one being ease of implementation through libraries such as pandas and scikit learn. Pandas can handle data manipulation and scalability. Scikit Learn is a machine learning library that can perform multiple linear regression analyses. Multiple linear regression can consider multiple factors and produce accurate predictions. One disadvantage of the data extraction and preparation method is that R is geared more towards statistical roles and potentially could have produced the same results with fewer lines of code.

The data was extracted from the publicly available CSV file on Kaggle.com which shows the superstore data set with data only in the United States. The data was imported into Jupyter Lab utilizing Python and Anaconda. The first step once in the lab environment was to import the proper libraries and packages to begin the analysis. Pandas, numpy, matplotlib, seaborn, os, patsy,statsmodels, scipy, missingno, warnings, datetime, tensorflow, sklearn, imports, and helpers were imported. These libraries provide these functions for this analysis, matplotlib provides visualizations, numpy provides arrays, Keras provides deep neural networks, TensorFlow is a machine learning library, scipy is used for mathematic and scientific computations, pandas is a software package, os is an operating system, seaborn is used for statistical graphs, patsy is used for describing statistical models,

statsmodels provides classes and functions for statistical models, missingno visualize missing data, warnings provide the ability to ignore warnings, datetime provides classes for manipulating dates and times, and sklearn is a machine learning library.

[1]:
```
#Install libraries and packages needed for analysis
import pandas as pd
from pandas.api.types import CategoricalDtype

import numpy as np

import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns

import os


from patsy import dmatrices

from statsmodels.stats.outliers_influence import variance_inflation_factor as vif
import statsmodels.api as sm
import statsmodels.formula.api as smf

from scipy import stats

import missingno as msno

import warnings
warnings.filterwarnings('ignore')

from datetime import datetime
from datetime import date

import tensorflow as tf
from tensorflow import keras
```

```
import statsmodels.api as sm
from statsmodels.formula.api import ols

from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn import pipeline

from helpers import *

from imports import *
```

The working directory was checked, the data was imported and then the data was viewed.

[2]:
```
#Check working directory
directory_path = os.getcwd()
print("My current directory is : "+ directory_path)
```

```
My current directory is : C:\Users\ntrei
```

Then, the CSV file is imported from the working directory into Jupyter Lab.

[3]:
```
#Importing the data
superstore = pd.read_csv('SuperStoreData.csv', encoding_errors= 'replace')
```

Next, the first 5 rows of imported data are viewed.

[4]:
```
#Viewing the first 5 rows of data
superstore.head()
```

[4]:

	Row_ID	Order_ID	Order_Date	Ship_Date	Ship_Mode	Customer_ID	Customer_Name	Segment	Country	City	State	Postal_Code	Region	Product_ID
0	1	CA-2017-152156	8/11/2017	11/11/2017	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South	FUR-BO-10001798
1	2	CA-2017-152156	8/11/2017	11/11/2017	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South	FUR-CH-10000454
2	3	CA-2017-138688	12/6/2017	16/06/2017	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	California	90036.0	West	OFF-LA-10000240
3	4	US-2016-108966	11/10/2016	18/10/2016	Standard Class	SO-20335	Sean O Donnel	Consumer	United States	Fort Lauderdale	Florida	33311.0	South	FUR-TA-10000577

The data was checked for missing, null, or duplicate values. The data quality is high as it has no missing, null, or duplicate values. Overall data sparsity is 0%. The data contains both qualitative and quantitative values as required for multiple linear regression analysis.

```
[6]: superstore.describe()
```

[6]:

	Row_ID	Postal_Code	Sales
count	9800.000000	9789.000000	9800.000000
mean	4900.500000	55273.322403	230.769059
std	2829.160653	32041.223413	626.651875
min	1.000000	1040.000000	0.444000
25%	2450.750000	23223.000000	17.248000
50%	4900.500000	58103.000000	54.490000
75%	7350.250000	90008.000000	210.605000
max	9800.000000	99301.000000	22638.480000

The data is searched for missing or null values.

```
[7]: superstore.iloc[0].isna().sum()
```

```
[7]: 0
```

```
[8]: superstore.isnull().sum(axis=1)
```

```
[8]: 0       0
     1       0
     2       0
     3       0
     4       0
            ..
     9795    0
     9796    0
```

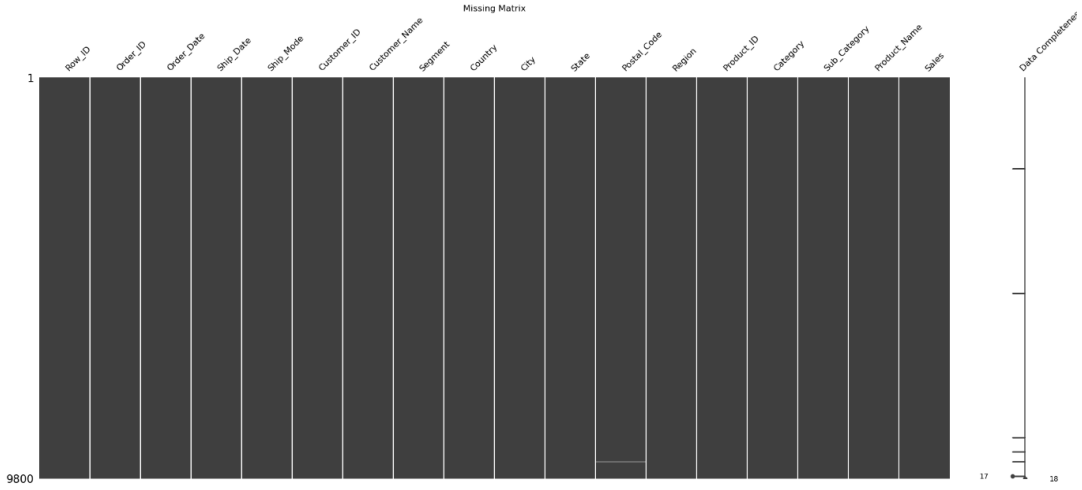
```
[9]: superstore.isna().sum
```

```
[9]: <bound method DataFrame.sum of       Row_ID  Order_ID  Order_Date  Ship_Date  Ship_Mode  Customer_ID  \
     0      False     False       False      False      False        False
     1      False     False       False      False      False        False
     2      False     False       False      False      False        False
     3      False     False       False      False      False        False
     4      False     False       False      False      False        False
     ...      ...       ...         ...        ...        ...          ...
     9795   False     False       False      False      False        False
     9796   False     False       False      False      False        False
     9797   False     False       False      False      False        False
     9798   False     False       False      False      False        False
     9799   False     False       False      False      False        False

           Customer_Name  Segment  Country   City  State  Postal_Code  Region  \
     0             False    False    False  False  False        False   False
     1             False    False    False  False  False        False   False
     2             False    False    False  False  False        False   False
     3             False    False    False  False  False        False   False
     4             False    False    False  False  False        False   False
     ...             ...      ...      ...    ...    ...          ...     ...
     9795          False    False    False  False  False        False   False
     9796          False    False    False  False  False        False   False
     9797          False    False    False  False  False        False   False
     9798          False    False    False  False  False        False   False
     9799          False    False    False  False  False        False   False

           Product_ID  Category  Sub_Category  Product_Name  Sales
     0          False     False         False         False  False
     1          False     False         False         False  False
     2          False     False         False         False  False
     3          False     False         False         False  False
     4          False     False         False         False  False
```

[10]:
```
superstore.duplicated().sum
```

[10]:
```
<bound method Series.sum of 0       False
1       False
2       False
3       False
4       False
        ...
9795    False
9796    False
9797    False
9798    False
9799    False
Length: 9800, dtype: bool>
```

[11]:
```
#use the missingno matrix,
msno.matrix(superstore, fontsize = 12, labels=True)
plt.title('Missing Matrix')
plt.show()
```



The data sparsity is calculated.

[12]:
```
#Checking sparsity
array = np.array(superstore)
sparsity = 1.0 - (np.count_nonzero(array) / float(array.size))
print(f"Sparsity: {sparsity:.6f}")
```

```
Sparsity: 0.000000
```

The unique values in the data frame are viewed.

[13]:
```
for col in superstore:
  print(superstore[col].unique())
```

```
[   1    2    3 ... 9798 9799 9800]
['CA-2017-152156' 'CA-2017-138688' 'US-2016-108966' ... 'CA-2015-127166'
 'CA-2017-125920' 'CA-2016-128608']
['8/11/2017' '12/6/2017' '11/10/2016' ... '18/06/2015' '28/02/2018'
 '9/5/2016']
['11/11/2017' '16/06/2017' '18/10/2016' ... '12/3/2015' '6/4/2018'
 '13/05/2016']
['Second Class' 'Standard Class' 'First Class' 'Same Day']
['CG-12520' 'DV-13045' 'SO-20335' 'BH-11710' 'AA-10480' 'IM-15070'
 'HP-14815' 'PK-19075' 'AG-10270' 'ZD-21925' 'KB-16585' 'SF-20065'
 'EB-13870' 'EH-13945' 'TB-21520' 'MA-17560' 'GH-14485' 'SN-20710'
 'LC-16930' 'RA-19885' 'ES-14080' 'ON-18715' 'PO-18865' 'LH-16900'
 'DP-13000' 'JM-15265' 'TB-21055' 'KM-16720' 'PS-18970' 'BS-11590'
 'KD-16270' 'HM-14980' 'JE-15745' 'KB-16600' 'SC-20770' 'DN-13690'
 'JC-16105' 'CS-12400' 'PG-18895' 'GM-14455' 'JS-15685' 'KB-16315'
 'RB-19705' 'PN-18775' 'KD-16345' 'ER-13855' 'RB-19465' 'GZ-14470'
 'LC-16870' 'JM-15250' 'PA-19060' 'CV-12805' 'CL-12565' 'RC-19960'
 'DK-13090' 'GG-14650' 'SC-20725' 'AD-10180' 'PF-19165' 'TS-21610'
 'LS-16975' 'DW-13585' 'LC-16885' 'JD-15895' 'SH-19975' 'SG-20080'
 'HA-14920' 'MG-17680' 'JE-16165' 'TW-21025' 'SP-20650' 'NK-18490'
 'DB-13060' 'NP-18670' 'TT-21070' 'EM-13960' 'RD-19900' 'MJ-17740'
 'BM-11140' 'CS-12130' 'JB-15400' 'SJ-20500' 'JK-15640' 'DK-13150'
 'RM-19675' 'SK-19990' 'FM-14290' 'AM-10360' 'MP-17470' 'MZ-17515'
```

The unique values for each column were calculated. The shape of encoding each unique value for each column in the data set was viewed. The cardinality is too great to use them all without proliferation.

```
[14]: pd.get_dummies(superstore, drop_first=True).shape

[14]: (9800, 13372)
```

Next, each variables' unique values and the count of those unique values are calculated.

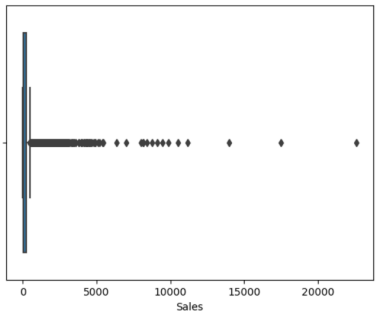
```
[15]: superstore.Sales.value_counts()

[15]: Sales
      12.960     55
      15.552     39
      19.440     39
      10.368     35
      25.920     34
                 ..
      339.136     1
      60.048      1
      5.022       1
      7.857       1
      10.384      1
      Name: count, Length: 5757, dtype: int64
```

```
[16]: superstore.Product_Name.value_counts()

[16]: Product_Name
      Staple envelope                                                   47
      Staples                                                           46
      Easy-staple paper                                                 44
      Avery Non-Stick Binders                                           20
      Staple remover                                                    18
                                                                        ..
      Park Ridge Embossed Executive Business Envelopes                   1
      Canon imageCLASS MF7460 Monochrome Digital Laser Multifunction Copier  1
```

Before manipulating the data it's important to first have visualizations to examine outliers, distribution shape and spread, relationships between variables, and correlation (Walker, 2020).

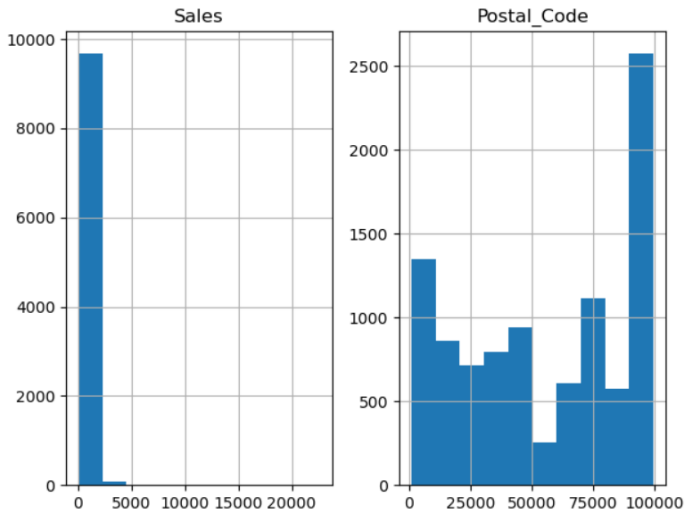
The data exploration continues by visualizing the data with boxplots, histograms, and scatter plots. Boxplots identify outliers, histograms visualize the distribution and scatterplots visualize the spread.

[29]:
```
#Create box plot for each variable prior to cleaning
boxplot=sns.boxplot(x='Sales',data=superstore)
plt.show()
```

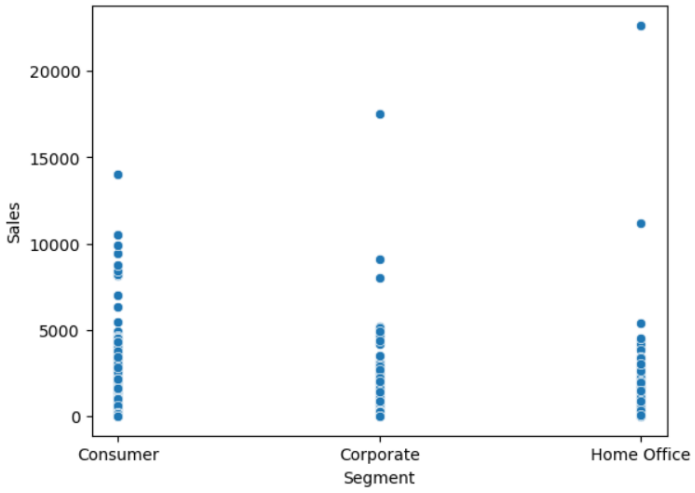


[30]:
```
boxplot=sns.boxplot(x='Postal_Code',data=superstore)
plt.show()
```

[31]:
```
# Create histograms of variables prior to cleaning
superstore[['Sales', 'Postal_Code']].hist()
plt.savefig('superstore_pyplot.jpg')
plt.tight_layout()
```



[32]:
```
##Add scatter plot for each variable before cleaning
sns.scatterplot(x="Segment",
                y="Sales",
                data=superstore)
plt.show()
```



Univariate and bivariate visualizations were created to view the distribution and spread of the data and to check for patterns. All categorical variables were converted using encoding to binary variables so that they could be utilized in the multiple linear regression model.

Order_Date

The categorical variables need to be one-hot encoded for analysis.

[42]:
```
# define categorical data
categorical_features = superstore.select_dtypes(include="object").columns
print(categorical_features)
```

```
Index(['Order_ID', 'Order_Date', 'Ship_Date', 'Ship_Mode', 'Customer_ID',
       'Customer_Name', 'Segment', 'Country', 'City', 'State', 'Region',
       'Product_ID', 'Category', 'Sub_Category', 'Product_Name'],
      dtype='object')
```

[43]:
```
# define bool features
bool_features = superstore.select_dtypes(include="bool").columns
print(bool_features)
```

```
Index([], dtype='object')
```

[44]:
```
# convert array to list then append lists
cat_or_bool_features = categorical_features.tolist() + bool_features.tolist()
print(cat_or_bool_features)
```

```
['Order_ID', 'Order_Date', 'Ship_Date', 'Ship_Mode', 'Customer_ID', 'Customer_Name', 'Segment', 'Country', 'City', 'State', 'Region', 'Product_ID', 'Cate
gory', 'Sub_Category', 'Product_Name']
```

[45]:
```
superstore = superstore.drop(['Order_ID', 'Customer_ID', 'Customer_Name', 'Country', 'City', 'State', 'Product_ID', 'Sub_Category', 'Product_Name'], axis
```

[46]:
```
ohe = OneHotEncoder(drop='first')
feature_arr = ohe.fit_transform(superstore[['Ship_Mode', 'Segment', 'Region', 'Category']]).toarray()
feature_labels = ohe.get_feature_names_out()
encoded_df = pd.DataFrame(feature_arr, columns=feature_labels)
```

[47]:
```
df = pd.concat([superstore.drop(['Ship_Mode', 'Segment', 'Region', 'Category'], axis=1), encoded_df], axis=1)
```

[48]:
```
print(df)
```

```
      Row_ID  Order_Date   Ship_Date  Postal_Code     Sales  \
0          1   8/11/2017  11/11/2017      42420.0  261.9600
1          2   8/11/2017  11/11/2017      42420.0  731.9400
2          3   12/6/2017  16/06/2017      90036.0   14.6200
3          4  11/10/2016  18/10/2016      33311.0  957.5775
4          5  11/10/2016  18/10/2016      33311.0   22.3680
...      ...         ...         ...          ...       ...
9795    9796  21/05/2017  28/05/2017      60610.0    3.7980
9796    9797   12/1/2016  17/01/2016      43615.0   10.3680
9797    9798   12/1/2016  17/01/2016      43615.0  235.1880
9798    9799   12/1/2016  17/01/2016      43615.0   26.3760
9799    9800   12/1/2016  17/01/2016      43615.0   10.3840

      Ship_Mode_Same Day  Ship_Mode_Second Class  Ship_Mode_Standard Class  \
0                    0.0                     1.0                       0.0
1                    0.0                     1.0                       0.0
2                    0.0                     1.0                       0.0
3                    0.0                     0.0                       1.0
4                    0.0                     0.0                       1.0
...                  ...                     ...                       ...
9795                 0.0                     0.0                       1.0
9796                 0.0                     0.0                       1.0
9797                 0.0                     0.0                       1.0
9798                 0.0                     0.0                       1.0
```

Toggle output scrolling mn is rounded to 2 decimal places because it is in reference to dollar amount and that is the standard format for money.

```
[49]: superstore['Sales'] = superstore['Sales'].round(2)
```

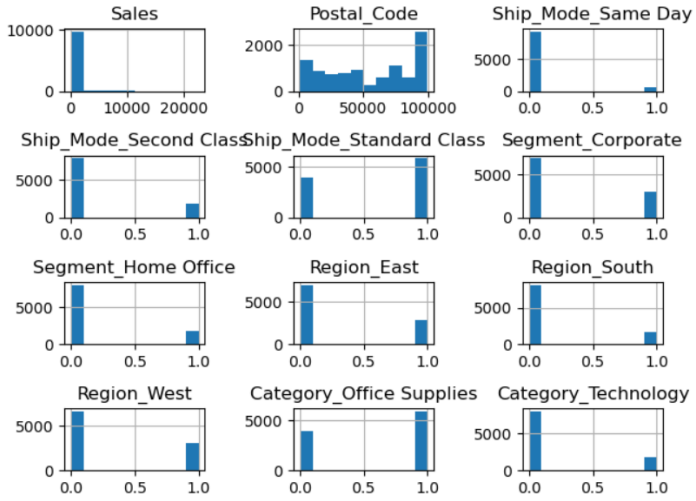
The first five rows of data are viewed to examine our changes have successfully occurred and look for other areas that need cleaned.

```
[50]: df.head()
```

[50]:

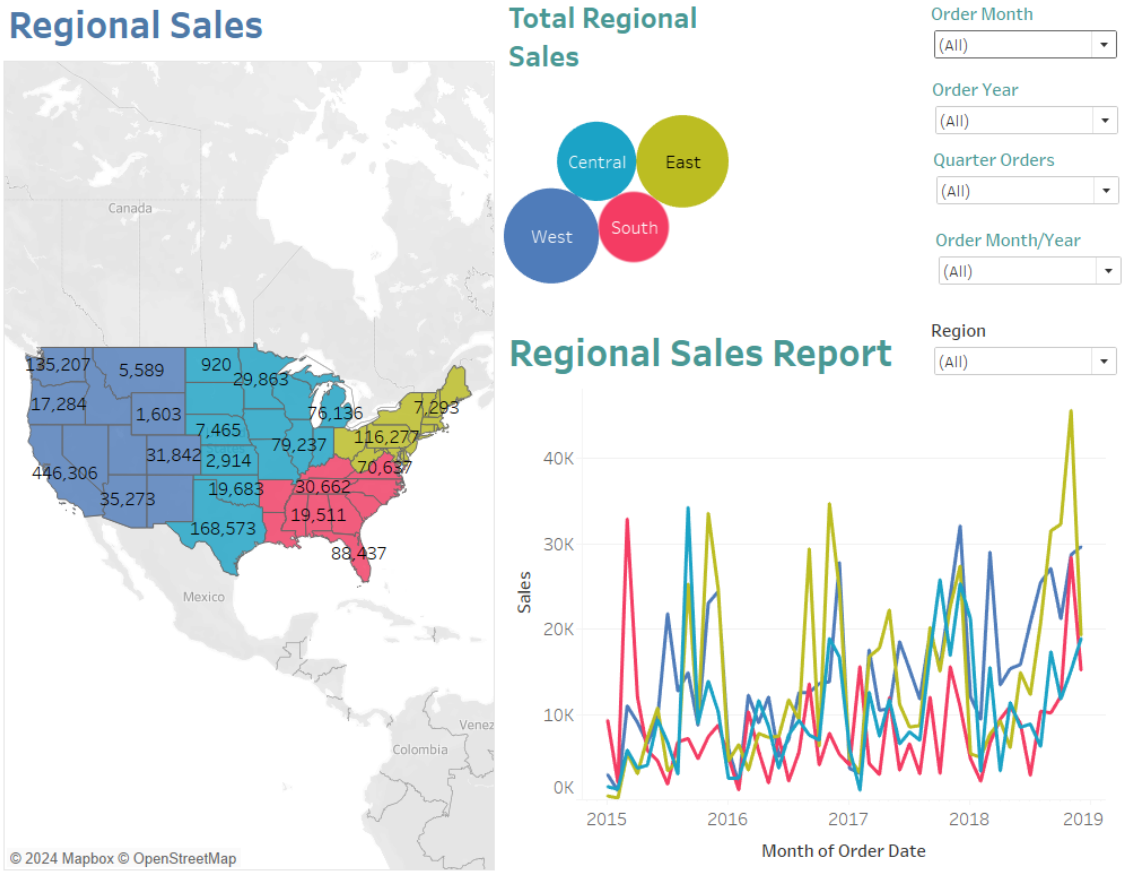
	Row_ID	Order_Date	Ship_Date	Postal_Code	Sales	Ship_Mode_Same Day	Ship_Mode_Second Class	Ship_Mode_Standard Class	Segment_Corporate	Segment_Home Office	Region_East
0	1	8/11/2017	11/11/2017	42420.0	261.9600	0.0	1.0	0.0	0.0	0.0	0.0
1	2	8/11/2017	11/11/2017	42420.0	731.9400	0.0	1.0	0.0	0.0	0.0	0.0
2	3	12/6/2017	16/06/2017	90036.0	14.6200	0.0	1.0	0.0	1.0	0.0	0.0
3	4	11/10/2016	18/10/2016	33311.0	957.5775	0.0	0.0	1.0	0.0	0.0	0.0
4	5	11/10/2016	18/10/2016	33311.0	22.3680	0.0	0.0	1.0	0.0	0.0	0.0

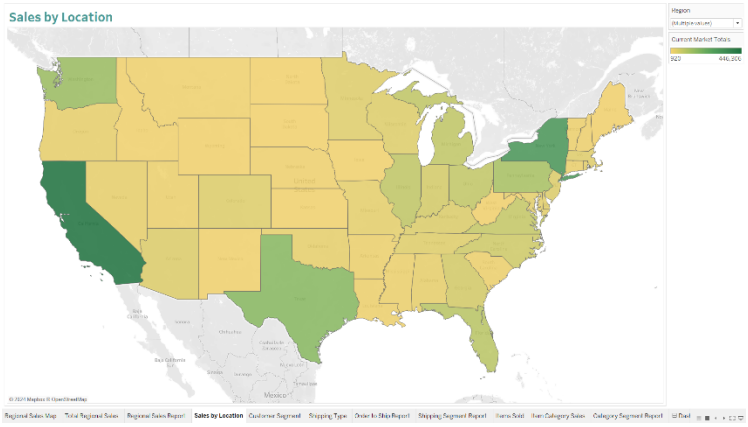
```
[51]: # Create histograms of variables
df[['Sales', 'Order_Date', 'Ship_Date', 'Postal_Code', 'Ship_Mode_Same Day', 'Ship_Mode_Second Class', 'Ship_Mode_Standard Class', 'Segment_Corporate',
plt.savefig('superstore_pyplot.jpg')
plt.tight_layout()
```

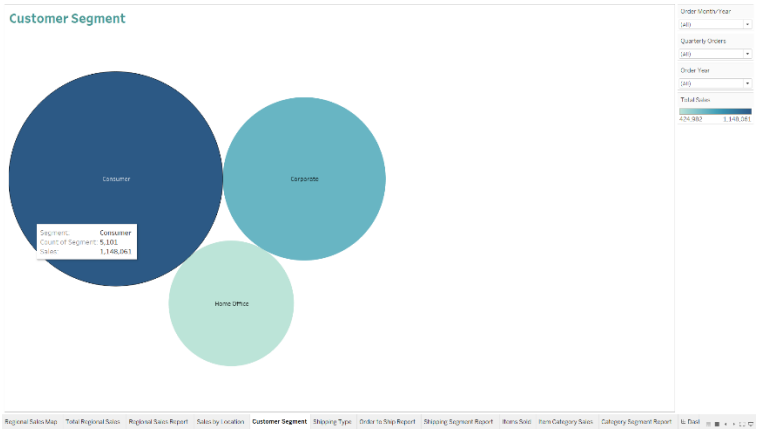


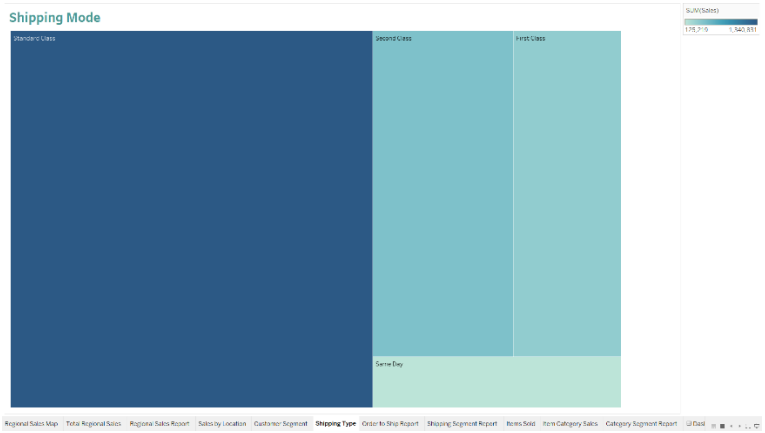
The original dataset was uploaded to Tableau Desktop and the data was explored. Reports were created looking into regional sales. The image below shows which states are part of which region. Each state's sales are shown on the map. There is an interactive feature that allows for the month, year, month/year combination, or quarterly sales to be viewed individually. The report shows the total regional sales with the West having the highest profits and the South with the least. The report also shows a line graph displaying this same information but over time. There is a report that visualizes which states have the most and least sales. These reports were also created to visualize shipping trends, segment

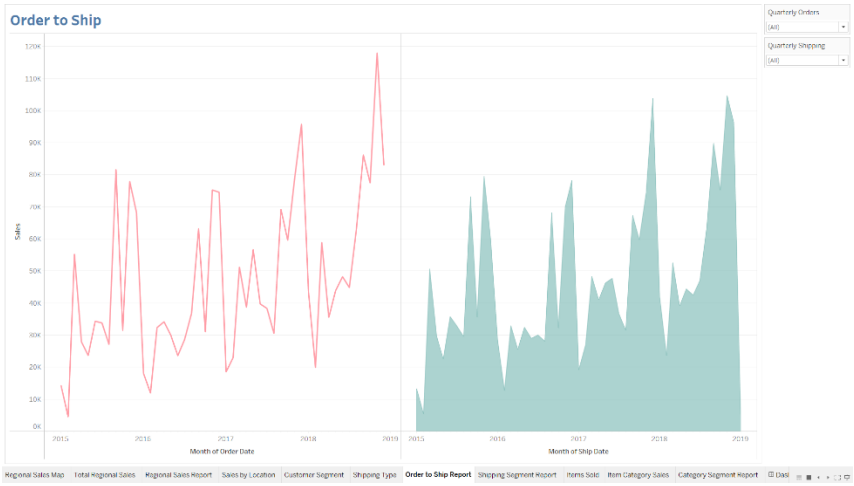
trends, and product trends.

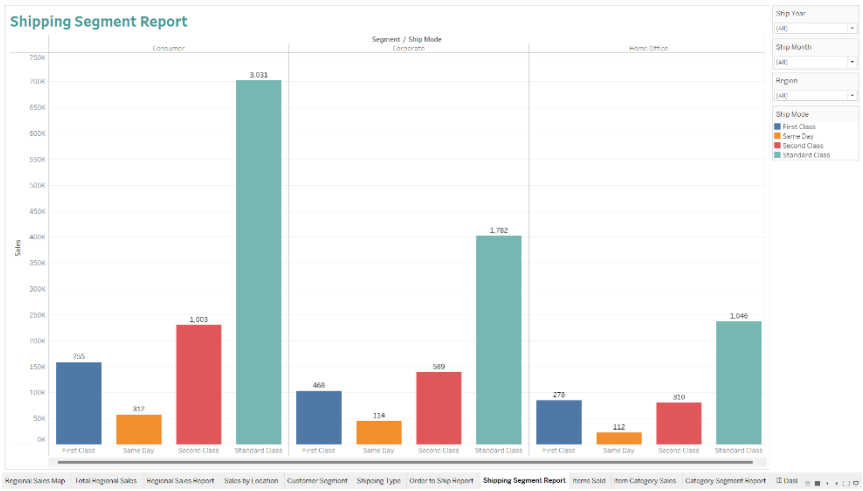


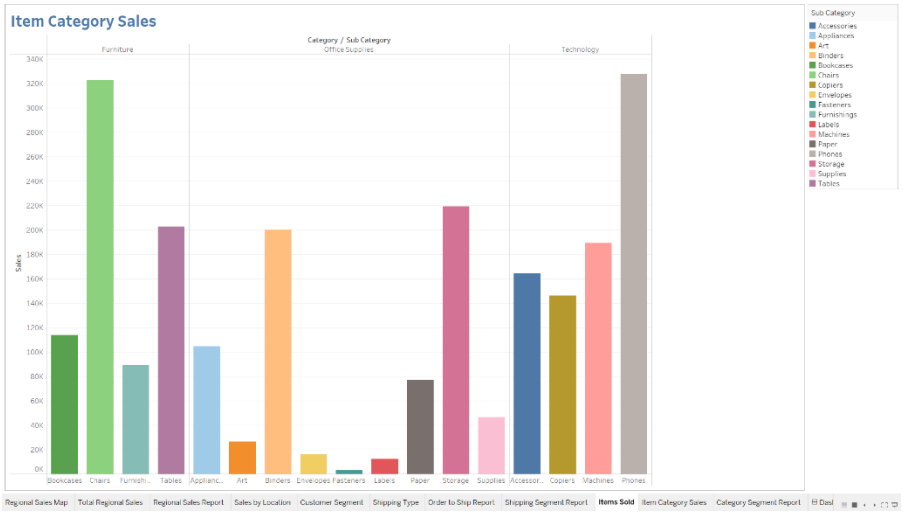

Sales by Location
Region
Current Market Totals
Regional Sales Map
Total Regional Sales
Regional Sales Report
Sales by Location
Customer Segment
Shipping Type
Order to Ship Report
Shipping Segment Report
Items Sold
Item Category Sales
Category Segment Report

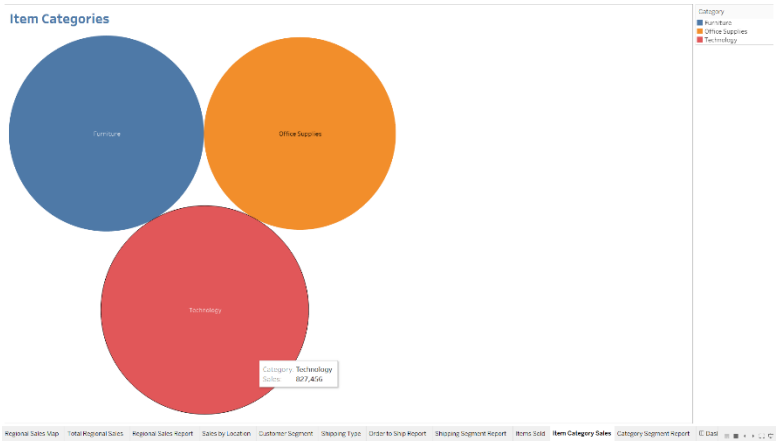

Customer Segment
Order Month/Year
Quarterly Orders
Order Year
Total Sales
Consumer
Corporate
Home Office
Segment: Consumer
Count of Segment: 5,101
Sales: 1,148,061
Regional Sales Map
Total Regional Sales
Regional Sales Report
Sales by Location
Customer Segment
Shipping Type
Order to Ship Report
Shipping Segment Report
Items Sold
Item Category Sales
Category Segment Report

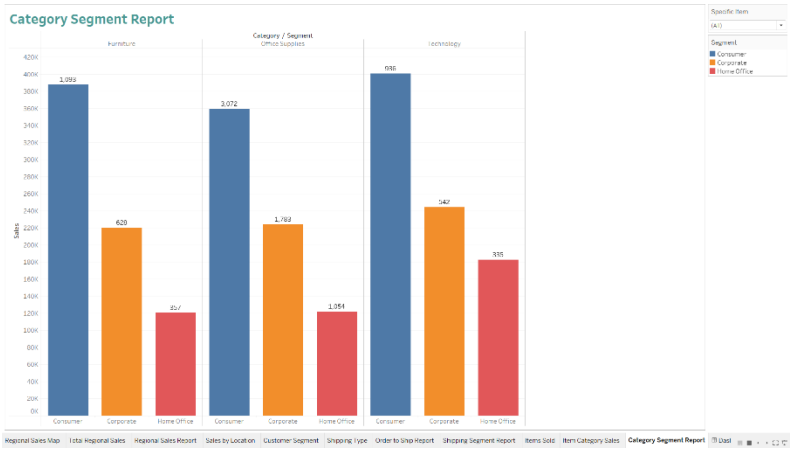

Shipping Mode
SUM(Sales)
Standard Class
Second Class
First Class
Same Day
Regional Sales Map
Total Regional Sales
Regional Sales Report
Sales by Location
Customer Segment
Shipping Type
Order to Ship Report
Shipping Segment Report
Items Sold
Item Category Sales
Category Segment Report


Order to Ship
Quarterly Orders
Quarterly Shipping
Sales
Month of Order Date
Month of Ship Date
Regional Sales Map
Total Regional Sales
Regional Sales Report
Sales by Location
Customer Segment
Shipping Type
Order to Ship Report
Shipping Segment Report
Items Sold
Item Category Sales
Category Segment Report

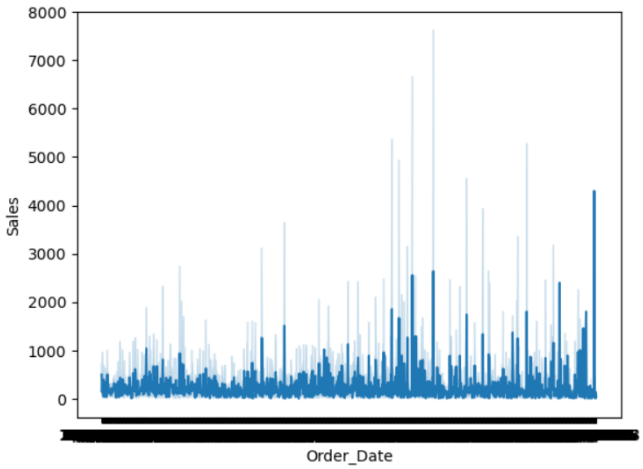

Shipping Segment Report
Segment / Ship Mode
Consumer
Corporate
Home Office
Ship Year
Ship Month
Region
Ship Mode
First Class
Same Day
Second Class
Standard Class
Sales
755
317
1,003
3,031
468
114
589
1,782
278
112
310
1,046
Regional Sales Map
Total Regional Sales
Regional Sales Report
Sales by Location
Customer Segment
Shipping Type
Order to Ship Report
Shipping Segment Report
Items Sold
Item Category Sales
Category Segment Report


Item Category Sales
Category / Sub Category
Furniture
Office Supplies
Technology
Sub Category
Accessories
Appliances
Art
Binders
Bookcases
Chairs
Copiers
Envelopes
Fasteners
Furnishings
Labels
Machines
Paper
Phones
Storage
Supplies
Tables
Sales
Bookcases
Chairs
Furnish..
Tables
Applianc..
Art
Binders
Envelopes
Fasteners
Labels
Paper
Storage
Supplies
Accessor..
Copiers
Machines
Phones
Regional Sales Map
Total Regional Sales
Regional Sales Report
Sales by Location
Customer Segment
Shipping Type
Order to Ship Report
Shipping Segment Report
Items Sold
Item Category Sales
Category Segment Report





[76]:
```
sns.lineplot(x="Order_Date",
             y="Sales",
             data=df)
plt.show()
```



Analysis

Pandas has many tools that were used to clean the data including calculating summary statistics, changing series values conditionally, evaluating and cleaning strings, working with dates, and missing value imputation (Walker, 2020). For ease of reading and interpreting columns were renamed. All data types were converted to numeric so that the multiple linear analysis could be performed. The k-1 method was utilized when one-hot encoding. Columns that served no purpose in the analysis such as Customer_ID, Product_ID, etc. were excluded from the data frame. Datetime data was reformatted into month, year, and day columns for workability. A correlation matrix was calculated with a heat mat to visualize the correlation and satisfy the assumptions on multiple linear regression analysis. The distribution of the sales data was visualized with a histogram. A Q-Q plot and Shapiro-Wilk were calculated to determine normality and to view the test statistic. Although it was not required for multiple regression analysis it will provide visualization of the data. ANOVA was performed on all variables, although not required it provides insight into the data. The variance inflation factor was calculated to satisfy the assumptions of multiple linear regression analysis. A VIF of 1 means there is no correlation between the predictor and remaining predictor variables, a VIF greater than 5 warrants an investigation, and a VIF of 10 or greater is a sign of multicollinearity requiring correction. Based on the VIF scores it is advisable that in the reduced model variables that should be removed from the multiple regression model are OrderDateYear, OrderDateMonth, ShipDateYear, and ShipDateMonth. StandardClassShipping is very close to the benchmark at 4.94 and may also be excluded. The data was split into training and test sets and the initial model was calculated using the kitchen sink approach, where all variables are included. The initial model was then reduced based on the multiple linear regression model assumptions and tests performed previously such as VIF. The model was reduced until all assumptions were met and only statistically significant variables remained in the final model.

Toggle output scrolling

Renaming columns for ease of coding and reading.

[80]:
```
df.rename(columns={'Ship_Mode_Same Day': 'SameDayShipping', 'Ship_Mode_Second Class': 'SecondClassShipping', 'Ship_Mode_Standard Class': 'StandardClassSh
```

The cleaned data is exported onto a CSV file.

[82]:
```
# export file to csv
df.to_csv('D214superstoreclean.csv', index = False)
```

The describe() and info() functions are utilized again to see the details of the data after cleaning.

[83]:
```
df.describe()
```

[83]:

	Row_ID	Postal_Code	Sales	SameDayShipping	SecondClassShipping	StandardClassShipping	Corporate	HomeOffice	Region_East	Region_South
count	9800.000000	9789.000000	9800.000000	9800.000000	9800.000000	9800.000000	9800.000000	9800.000000	9800.000000	9800.000000
mean	4900.500000	55273.322403	230.769059	0.054898	0.194082	0.597857	0.301327	0.178163	0.284184	0.163061
std	2829.160653	32041.223413	626.651875	0.227793	0.395512	0.490355	0.458858	0.382670	0.451048	0.369440
min	1.000000	1040.000000	0.444000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	2450.750000	23223.000000	17.248000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Columns that cannot be utilized in the multiple linear regression analysis are excluded as they provide no statistical significance.

Data types are changed from object to numeric values so that they can be used in the multiple linear regression analysis.

[87]:
```
df['Postal_Code'] = df['Postal_Code'].astype(float)
df['Postal_Code'] = df['Postal_Code'].astype('Int64')
```

The columns that contain dates are changed to a uniform and standard format month-day-year.

[88]:
```
df['Order_Date'] = pd.to_datetime(df['Order_Date'], format='mixed')
df['Order_Date'] = df['Order_Date'].dt.strftime('%m-%d-%Y')
print (df)
```

```
      Row_ID  Order_Date   Ship_Date  Postal_Code     Sales  SameDayShipping  \
0          1  08-11-2017  11/11/2017        42420  261.9600              0.0
1          2  08-11-2017  11/11/2017        42420  731.9400              0.0
2          3  12-06-2017  16/06/2017        90036   14.6200              0.0
3          4  11-10-2016  18/10/2016        33311  957.5775              0.0
4          5  11-10-2016  18/10/2016        33311   22.3680              0.0
...      ...         ...         ...          ...       ...              ...
9795    9796  05-21-2017  28/05/2017        60610    3.7980              0.0
9796    9797  12-01-2016  17/01/2016        43615   10.3680              0.0
9797    9798  12-01-2016  17/01/2016        43615  235.1880              0.0
9798    9799  12-01-2016  17/01/2016        43615   26.3760              0.0
9799    9800  12-01-2016  17/01/2016        43615   10.3840              0.0

      SecondClassShipping  StandardClassShipping  Corporate  HomeOffice  \
0                     1.0                    0.0        0.0         0.0
1                     1.0                    0.0        0.0         0.0
2                     1.0                    0.0        1.0         0.0
```

[89]:
```
df['Ship_Date'] = pd.to_datetime(df['Ship_Date'], format= "mixed")
df['Ship_Date'] = df['Ship_Date'].dt.strftime('%m-%d-%Y')
print(df)
```

```
      Row_ID  Order_Date   Ship_Date  Postal_Code     Sales  SameDayShipping  \
0          1  08-11-2017  11-11-2017        42420  261.9600              0.0
1          2  08-11-2017  11-11-2017        42420  731.9400              0.0
2          3  12-06-2017  06-16-2017        90036   14.6200              0.0
3          4  11-10-2016  10-18-2016        33311  957.5775              0.0
4          5  11-10-2016  10-18-2016        33311   22.3680              0.0
...      ...         ...         ...          ...       ...              ...
9795    9796  05-21-2017  05-28-2017        60610    3.7980              0.0
9796    9797  12-01-2016  01-17-2016        43615   10.3680              0.0
9797    9798  12-01-2016  01-17-2016        43615  235.1880              0.0
9798    9799  12-01-2016  01-17-2016        43615   26.3760              0.0
9799    9800  12-01-2016  01-17-2016        43615   10.3840              0.0

      SecondClassShipping  StandardClassShipping  Corporate  HomeOffice  \
0                     1.0                    0.0        0.0         0.0
1                     1.0                    0.0        0.0         0.0
2                     1.0                    0.0        1.0         0.0
3                     0.0                    1.0        0.0         0.0
4                     0.0                    1.0        0.0         0.0
...                   ...                    ...        ...         ...
9795                  0.0                    1.0        1.0         0.0
9796                  0.0                    1.0        1.0         0.0
9797                  0.0                    1.0        1.0         0.0
9798                  0.0                    1.0        1.0         0.0
9799                  0.0                    1.0        1.0         0.0

      Region_East  Region_South  Region_West  OfficeSupplies  Technology
0             0.0           1.0          0.0             0.0         0.0
1             0.0           1.0          0.0             0.0         0.0
2             0.0           0.0          1.0             1.0         0.0
```

The date columns are then broken down into month, day and year columns.

[90]:
```
df['Sales'] = df['Sales'].astype(int)

# Change the type of 'Order_Date' and 'Ship_Date'
df['Order_Date'] = pd.to_datetime(df['Order_Date'])
df['Ship_Date'] = pd.to_datetime(df['Ship_Date'])

df['year_order_date'] = df['Order_Date'].dt.year
df['month_order_date'] = df['Order_Date'].dt.month
df['weekday_order_date'] = df['Order_Date'].dt.weekday

df['year_ship_date'] = df['Ship_Date'].dt.year
df['month_ship_date'] = df['Ship_Date'].dt.month
df['weekday_ship_date'] = df['Ship_Date'].dt.weekday
```

Changes made to the data are viewed again.

[91]:
```
df.head()
```

[91]:

	Row_ID	Order_Date	Ship_Date	Postal_Code	Sales	SameDayShipping	SecondClass
0	1	2017-08-11	2017-11-11	42420	261	0.0	
1	2	2017-08-11	2017-11-11	42420	731	0.0	
2	3	2017-12-06	2017-06-16	90036	14	0.0	
3	4	2016-11-10	2016-10-18	33311	957	0.0	

[94]:
```
df.rename(columns={'Postal_Code': 'PostCode', 'Region_East': 'EastRegion', 'Region_South': 'SouthRegion', 'Region_West': 'WestRegion', 'year_order_date':
df.drop('Order_Date', axis=1, inplace=True)
df.drop('Ship_Date', axis=1, inplace=True)
```

[95]:
```
df.head()
```

[95]:

	Row_ID	PostCode	Sales	SameDayShipping	SecondClassShipping	StandardClassShipping	Corporate	HomeOffice	EastRegion	SouthRegion	WestRegion	OfficeSup
0	1	42420	261	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	
1	2	42420	731	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	
2	3	90036	14	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	
3	4	33311	957	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	
4	5	33311	22	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	

The cleaned dataset is exported to a CSV file.

[96]:
```
# export file to csv
df.to_csv('D214superstoreclean1.csv', index = False)
```

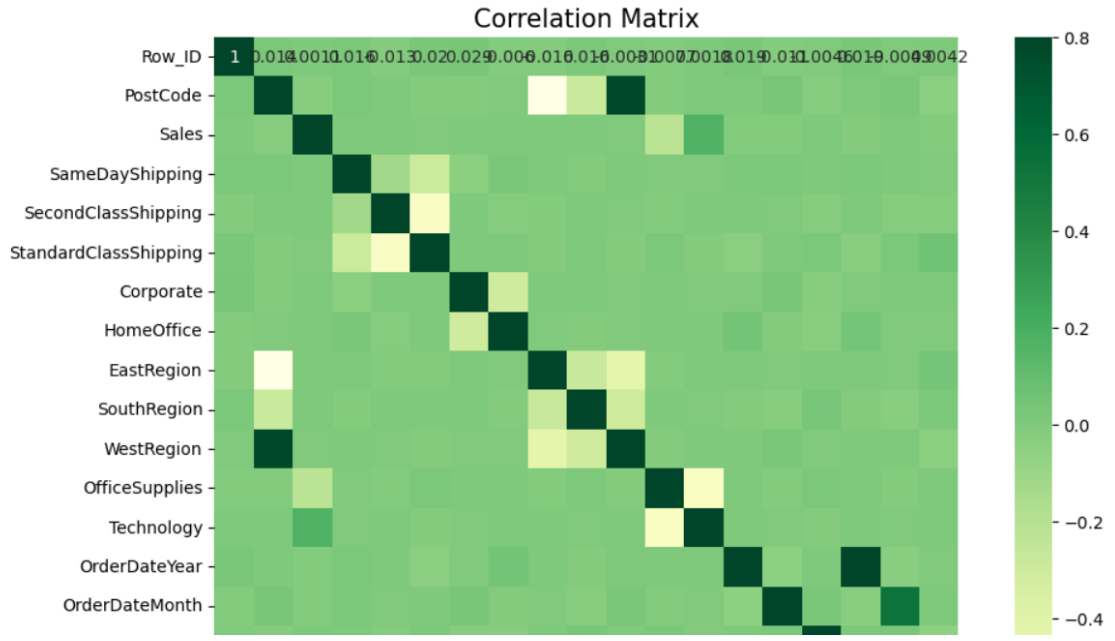
A correlation matrix is created to check for correlation among variables.

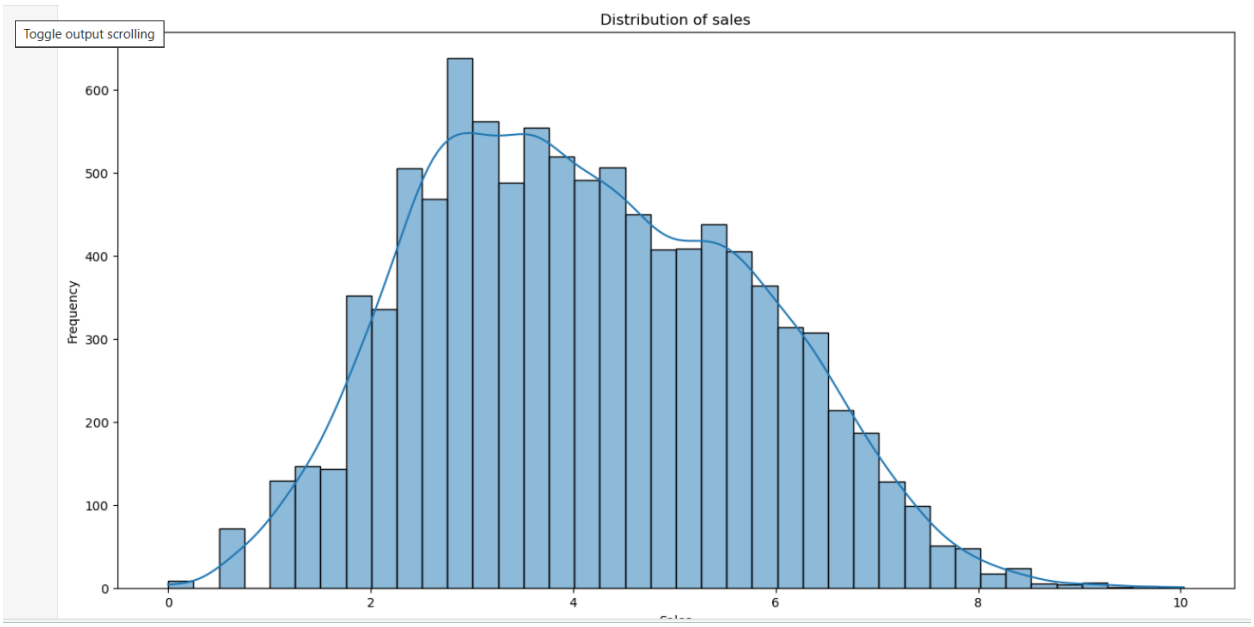
[98]:
```
numeric_columns = df.select_dtypes(include=[np.number])
correlation_matrix = numeric_columns.corr()
correlation_matrix
```

[98]:

	Row_ID	PostCode	Sales	SameDayShipping	SecondClassShipping	StandardClassShipping	Corporate	HomeOffice	EastRegion	SouthRegio
Row_ID	1.000000	0.013645	0.001149	0.015671	-0.013183	0.019757	0.029110	-0.006028	-0.015802	0.01610
PostCode	0.013645	1.000000	-0.024056	0.016458	0.005512	-0.008820	-0.010940	-0.002930	-0.738841	-0.28649
Sales	0.001149	-0.024056	1.000000	0.000766	0.004517	-0.003724	0.002495	0.009391	0.009679	0.00898
SameDayShipping	0.015671	0.016458	0.000766	1.000000	-0.118273	-0.293865	-0.046975	0.018905	0.001102	-0.00573
SecondClassShipping	-0.013183	0.005512	0.004517	-0.118273	1.000000	-0.598350	0.008928	-0.019464	-0.011165	0.01037
StandardClassShipping	0.019757	-0.008820	-0.003724	-0.293865	-0.598350	1.000000	0.007496	0.001165	-0.011550	0.00147
Corporate	0.029110	-0.010940	0.002495	-0.046975	0.008928	0.007496	1.000000	-0.305772	0.005328	0.01112
HomeOffice	-0.006028	-0.002930	0.009391	0.018905	-0.019464	0.001165	-0.305772	1.000000	-0.003065	-0.01278
EastRegion	-0.015802	-0.738841	0.009679	0.001102	-0.011165	-0.011550	0.005328	-0.003065	1.000000	-0.27811
SouthRegion	0.016103	-0.286491	0.008984	-0.005732	0.010377	0.001478	0.011125	-0.012780	-0.278117	1.00000
WestRegion	-0.003069	0.781519	-0.005022	0.010196	-0.000230	-0.016176	-0.001985	0.000324	-0.432640	-0.30307
OfficeSupplies	-0.007696	-0.009276	-0.219000	-0.001275	-0.009401	0.016274	0.001119	0.000672	-0.005660	0.01099
Technology	0.001825	0.005200	0.171459	-0.002919	0.004738	-0.012817	-0.002466	0.008234	0.006861	-0.00471
OrderDateYear	0.019065	0.006418	-0.010622	0.017693	0.002893	-0.037216	-0.005246	0.043082	0.007679	-0.01578
OrderDateMonth	-0.011146	0.025064	-0.009928	0.016860	-0.004446	0.001672	0.027499	-0.014796	-0.002138	-0.02576
OrderDateDay	-0.004628	-0.021842	0.002128	0.009602	-0.022678	0.017255	-0.024822	-0.029563	0.010584	0.03121

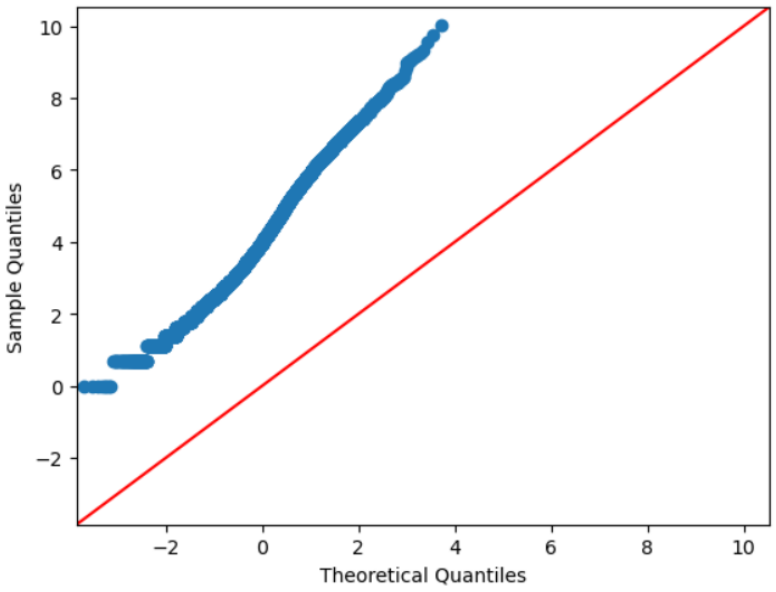
[99]:
```
fig, ax = plt.subplots()
fig.set_size_inches(12,8)
sns.heatmap(correlation_matrix, vmax =.8, square = True, annot = True,cmap='YlGn' )
plt.title('Correlation Matrix',fontsize=15);
```





A QQ plot was created, although not required for multiple linear regression analysis it is a good way to visualize the data.

[101]:
```
# Create the Q-Q plot with a 45-degree line
fig = sm.qqplot(data=np.log1p(df['Sales']), line='45')
plt.show()
```



A Shapiro-Wilk was run to see the test statistic.

[102]:
```
#Shapiro-Wilk
statistic, p_value = stats.shapiro(df['Sales'])
print(f"Shapiro-Wilk test statistic: {statistic:.4f}")
print(f"P-value: {p_value:.4f}")
```

```
Shapiro-Wilk test statistic: 0.3226
P-value: 0.0000
```

ANOVA was run to view the statistic.

[105]:
```
#Run ANOVA
# Grouping the data by SameDayShipping
grouped_data = [df[df['SameDayShipping'] == mode]['Sales'] for mode in df['SameDayShipping'].unique()]

# Performing ANOVA test
anova_result = stats.f_oneway(*grouped_data)

print("ANOVA F-value:", anova_result.statistic)
print("ANOVA p-value:", anova_result.pvalue)
```

```
ANOVA F-value: 0.005743756404279135
ANOVA p-value: 0.9395896493394683
```

VIF was checked.

[112]:
```
#Calculate Variance Inflation Factor
from statsmodels.stats.outliers_influence import variance_inflation_factor

# the independent variables set
X = df[['SameDayShipping', 'SecondClassShipping', 'StandardClassShipping', 'Corporate', 'HomeOffice', 'EastRegion', 'WestRegion', 'SouthRegion', 'OfficeS

# VIF dataframe
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns

# calculating VIF for each feature
vif_data["VIF"] = [variance_inflation_factor(X.values, i)
                   for i in range(len(X.columns))]

print(vif_data)
```

	feature	VIF
0	SameDayShipping	1.363118e+00
1	SecondClassShipping	2.273063e+00
2	StandardClassShipping	4.943143e+00
3	Corporate	1.585985e+00
4	HomeOffice	1.345858e+00
5	EastRegion	2.233553e+00
6	WestRegion	2.387902e+00
7	SouthRegion	1.710106e+00
8	OfficeSupplies	3.846229e+00
9	Technology	1.873322e+00
10	OrderDateYear	3.144436e+08
11	OrderDateMonth	8.400819e+00
12	OrderDateDay	2.954805e+00
13	ShipDateYear	3.144429e+08
14	ShipDateMonth	8.775129e+00
15	ShipDateDay	3.479309e+00

A VIF of 1 means there is no correlation between the predictor and remaining predictor variables, a VIF greater than 5 warrants an investigation, a VIF of 10 or greater is a sign of multicollinearity requiring correction. Based on the VIF scores it is advisable that in the reduced model variables that should be removed from the multiple regression model are OrderDateYear, OrderDateMonth, ShipDateYear, and ShipDateMonth. StandardClassShipping is very close to the benchmark at 4.94 and may also be excluded.

The data is split into training and test sets.

[113]:
```
#Splitting the data into train and test sets
rmse_results = {}
mae_results = {}
r2score_results = {}

X = df.drop(columns=['Sales'])
y = df['Sales']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

[114]:
```
# Preprocessing steps for numeric and categorical features
numeric_features = X_train.select_dtypes(include=[np.number]).columns.tolist()
categorical_features = X_train.select_dtypes(include=['object']).columns.tolist()
```

[115]:
```
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder(handle_unknown='ignore'))
])
```

The data analysis technique that was employed was Multiple Linear Regression Analysis. Multiple linear regression analysis is an appropriate technique because the data consists of one continuous dependent variable and several independent variables (Mishra, Pandey, Singh, Keshri, 2019). The analysis was performed using Python in the Jupyter Lab environment. The goals and expectations of the study were to predict future sales revenue

utilizing the provided superstore sales dataset. The multiple linear regression analysis uses historic sales data to find patterns between the independent and dependent variables to predict future sales. Predicting future sales revenue and the variables that have the most statistically significant impact on the sales revenue provides actionable insights for smart business decisions.

Data Summary and Implications

The analysis implies that a multiple linear regression model can be constructed with the superstore sales data. An initial multiple linear regression model was constructed using the kitchen sink approach and putting all variables into the model. The root mean square error or rmse score for that model was 640497.38, the mean absolute error or mae score was 273.77, and the r squared score was 0.04. Another multiple linear regression model was calculated this one without the intercept. The r squared for this model was .164, and the adjusted r squared was .163. The variance inflation factor or VIF has been run and variables with a VIF of greater than 10 will be considered for removal in the reduced model due to a high VIF meaning there is multicollinearity. Variables with P values of greater than .05 will be considered for removal in the reduced model. The first reduced model consisted of LMR = ols(formula="Sales ~ SameDayShipping + SecondClassShipping + Corporate + HomeOffice + EastRegion + SouthRegion + WestRegion + OfficeSupplies + Technology + OrderDateDay + ShipDateDay", data=df).fit(). The r squared of this model was .051 and the adjusted r squared was .050. The model could be improved further as the research suggests that the best multiple linear regression model would have five or fewer variables. The next reduced model included LMRF = ols(formula="Sales ~ SameDayShipping + OfficeSupplies + Technology", data=df).fit(). The r squared remained at .051 with an adjusted r squared of .050. There was no improvement in this model in regards to the statistic r squared and adjusted r squared. The reduced model was then calculated with no intercept. The r squared then changed to .106 with an adjusted r squared

of .106. The data frame was then standardized and the final multiple linear regression model was calculated. The rmse of this model was 1.63, mae was .44 and the r squared was .04. The closer to zero the rmse score is the more accurate the prediction is. The closer the mae is to zero the more accurate the models' prediction. The closer to 1 the r squared value is the better fit the regression is. This indicates that the variable office supplies and technology have the greatest impact on the sales revenue. Based on these findings it is recommended that the sale of technology and office supplies be prioritized in future sale efforts as they produce the greatest return of the variables in this study. Future studies of this dataset should include a market basket analysis or another type of classification model. A limitation of the study is the lack of review data for products and superstore locations. To improve the dataset reviews of the products and/or store locations could be beneficial in future studies on increasing sales revenue.

References

Anseur, A. (2024). *Superstore Sales EDA + ML.* Kaggle.com. Retrieved April 16, 2024, from Superstore Sales | EDA + ML (kaggle.com)

Chattopadhyay, R. (2016). *Effective Businesss Solutions With Big Data Analytics: Key for Business Growth.* Globsyn Management Journal, 10(1/2), 87–96. Retrieved April 16, 2024, from https://search.ebscohost.com/login.aspx?direct=true&db=buh&AN=132005022&site=eds-live&scope=site&authtype=shib&custid=ns017578

Darlington, R. & Hayes, A. (2017). *Regression Analysis and Linear Models : Concepts, Applications, and Implementation.* The Guilford Press. Retrieved April 16, 2024, from https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1340168&site=eds-live&scope=site&authtype=shib&custid=ns017578

Dong, J., Chen, Y., Gu, A., Chen, J., Li, L., Chen, Q., Li, S., & Xun, Q. (2020). *Potential Trend for Online Shopping Data Based on the Linear Regression and Sentiment Analysis.* Mathematical Problems in Engineering, 1–11. Retrieved April 16, 2024, from https://doi.org/10.1155/2020/4591260

Mishra, P., Pandey, C. M., Singh, U., Keshri, A., & Sabaretnam, M. (2019). *Selection of appropriate statistical methods for data analysis.* Annals of Cardiac Anaesthesia, 22(3), 297–301. Retrieved April 16, 2024, from https://doi.org/10.4103/aca.ACA_248_18

Multiple Regression. (n.d.). *Multiple Regression.* Csulb.edu. Retrieved April 16, 2024, from Multiple Regression (csulb.edu)

Open Knowledge. (n.d.). *Open Data Commons Public Domain.* Opendatacommons.org. Open Data Commons Public Domain Dedication and License (PDDL) v1.0 — Open Data Commons: legal tools for open data

Ozgur, C., Colliau, T., Rogers, G., Hughes, Z., & Myer-Tyson, E. "Bennie." (2017). *MatLab vs. Python vs. R.* Journal of Data Science, 15(3), 355–371. Retrieved April 16, 2024, from https://search.ebscohost.com/login.aspx?direct=true&db=asn&AN=125220011&site=eds-live&scope=site&authtype=shib&custid=ns017578

Siddiqui, T., Alkadri, M., & Khan, N. A. (2017). *Review of Programming Languages and Tools for Big Data Analytics.* International Journal of Advanced Research in Computer Science, 8(5), 1112–1118. Retrieved April 16, 2024, from https://search.ebscohost.com/login.aspx?direct=true&db=asf&AN=124636531&site=eds-live&scope=site&authtype=shib&custid=ns017578

Walker, M. (2020). *Python Data Cleaning Cookbook : Modern Techniques and Python Tools to Detect and Remove Dirty Data and Extract Key Insights.* Packt Publishing. Retrieved April 16, 2024, from https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=2706043&site=eds-live&scope=site&authtype=shib&custid=ns017578

Zach. (2020). *The four assumptions of linear regression.*

The Four Assumptions of Linear Regression - Statology

Zach. (2020). *Four examples of using multiple linear regression.*

4 Examples of Using Linear Regression in Real Life - Statology

Zach. (2020). *Multiple linear regression by hand.*

Multiple Linear Regression by Hand (Step-by-Step) - Statology