Final Project (Continuation of Mid-Term Project)

Rishabh Kaushick
NU ID: 002808996,
kaushick.r@northeastern.edu

College of Engineering,
Northeastern University
Toronto, ON

In [9]:
```
# importing the required libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

In [10]:
```
# loading the data
# structured data
scm_dataset_path = 'data/DataCoSupplyChainDataset.csv'
scm_df = pd.read_csv(scm_dataset_path, encoding='latin-1')

# unstructured data
scm_unstructured_path = 'data/tokenized_access_logs.csv'
scm_unstructured_df = pd.read_csv(scm_unstructured_path)
```

In [11]:
```
# Let's see what the first five rows data looks like
scm_df.head()
```

Out[11]:

	Type	Days for shipping (real)	Days for shipment (scheduled)	Benefit per order	Sales per customer	Delivery Status	Late_deli
0	DEBIT	3	4	91.250000	314.640015	Advance shipping	
1	TRANSFER	5	4	-249.089996	311.359985	Late delivery	
2	CASH	4	4	-247.779999	309.720001	Shipping on time	
3	DEBIT	3	4	22.860001	304.809998	Advance shipping	
4	PAYMENT	2	4	134.210007	298.250000	Advance shipping	

5 rows × 53 columns

In [12]:
```
# total number of rows in the above csv:
scm_df.count()
```

Out[12]:
```
Type                           180519
Days for shipping (real)       180519
Days for shipment (scheduled)  180519
Benefit per order              180519
Sales per customer             180519
Delivery Status                180519
Late_delivery_risk             180519
Category Id                    180519
Category Name                  180519
Customer City                  180519
Customer Country               180519
Customer Email                 180519
Customer Fname                 180519
Customer Id                    180519
Customer Lname                 180511
Customer Password              180519
Customer Segment               180519
Customer State                 180519
Customer Street                180519
Customer Zipcode               180516
Department Id                  180519
Department Name                180519
Latitude                       180519
Longitude                      180519
Market                         180519
Order City                     180519
Order Country                  180519
Order Customer Id              180519
order date (DateOrders)        180519
Order Id                       180519
Order Item Cardprod Id         180519
Order Item Discount            180519
Order Item Discount Rate       180519
Order Item Id                  180519
Order Item Product Price       180519
Order Item Profit Ratio        180519
Order Item Quantity            180519
Sales                          180519
Order Item Total               180519
Order Profit Per Order         180519
Order Region                   180519
Order State                    180519
Order Status                   180519
Order Zipcode                   24840
Product Card Id                180519
Product Category Id            180519
Product Description                 0
Product Image                  180519
Product Name                   180519
Product Price                  180519
Product Status                 180519
shipping date (DateOrders)     180519
Shipping Mode                  180519
dtype: int64
```

In [13]:
```
scm_unstructured_df.head()
```

Out[13]:

	Product	Category	Date	Month	Hour	Department	ip	
0	adidas Brazuca 2017 Official Match Ball	baseball & softball	9/1/2017 6:00	Sep	6	fitness	37.97.182.65	
1	The North Face Women's Recon Backpack	hunting & shooting	9/1/2017 6:00	Sep	6	fan shop	206.56.112.1	/de
2	adidas Kids' RG III Mid Football Cleat	featured shops	9/1/2017 6:00	Sep	6	apparel	215.143.180.0	
3	Under Armour Men's Compression EV SL Slide	electronics	9/1/2017 6:00	Sep	6	footwear	206.56.112.1	
4	Pelican Sunstream 100 Kayak	water sports	9/1/2017 6:01	Sep	6	fan shop	136.108.56.242	/

In [14]:
```
scm_unstructured_df.count()
```

Out[14]:
```
Product       469977
Category      469977
Date          469977
Month         469977
Hour          469977
Department    469977
ip            469977
url           469977
dtype: int64
```

In the main csv dataset, there are 53 columns and 180,519 rows. Therefore there are 53 X 180,519 = 9,567,507 data points.

However, from the unstructured dataset, we can see that there are 8 columns and 469,977 rows. Therefore there are 8 X 469,977 = 3,759,816 data points

Since the columns in each of the datasets are different from each other, and since there is a lot more data in the structured dataframe, we will use the same and disregard the unstructured data for the scope of this project.

In [15]:
```
# understanding the number of distinct non null numbers are present in ea
scm_df.nunique()
```

Out[15]:
```
Type                             4
Days for shipping (real)         7
Days for shipment (scheduled)    4
Benefit per order            21998
Sales per customer            2927
Delivery Status                  4
Late_delivery_risk               2
Category Id                     51
Category Name                   50
Customer City                  563
Customer Country                 2
Customer Email                   1
Customer Fname                 782
Customer Id                  20652
Customer Lname                1109
Customer Password                1
Customer Segment                 3
Customer State                  46
Customer Street               7458
Customer Zipcode               995
Department Id                   11
Department Name                 11
Latitude                     11250
Longitude                     4487
Market                           5
Order City                    3597
Order Country                  164
Order Customer Id            20652
order date (DateOrders)      65752
Order Id                     65752
Order Item Cardprod Id         118
Order Item Discount           1017
Order Item Discount Rate        18
Order Item Id                180519
Order Item Product Price         75
Order Item Profit Ratio         162
Order Item Quantity               5
Sales                           193
Order Item Total               2927
Order Profit Per Order        21998
Order Region                     23
Order State                   1089
Order Status                     9
Order Zipcode                  609
Product Card Id                118
Product Category Id             51
Product Description              0
Product Image                  118
Product Name                   118
Product Price                   75
Product Status                   1
shipping date (DateOrders)   63701
Shipping Mode                    4
dtype: int64
```

In [16]:
```
# checking which columns have null rows
scm_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180519 entries, 0 to 180518
Data columns (total 53 columns):
 #   Column                         Non-Null Count   Dtype  
---  ------                         --------------   -----  
 0   Type                           180519 non-null  object 
 1   Days for shipping (real)       180519 non-null  int64  
 2   Days for shipment (scheduled)  180519 non-null  int64  
 3   Benefit per order              180519 non-null  float64
 4   Sales per customer             180519 non-null  float64
 5   Delivery Status                180519 non-null  object 
 6   Late_delivery_risk             180519 non-null  int64  
 7   Category Id                    180519 non-null  int64  
 8   Category Name                  180519 non-null  object 
 9   Customer City                  180519 non-null  object 
 10  Customer Country               180519 non-null  object 
 11  Customer Email                 180519 non-null  object 
 12  Customer Fname                 180519 non-null  object 
 13  Customer Id                    180519 non-null  int64  
 14  Customer Lname                 180511 non-null  object 
 15  Customer Password              180519 non-null  object 
 16  Customer Segment               180519 non-null  object 
 17  Customer State                 180519 non-null  object 
 18  Customer Street                180519 non-null  object 
 19  Customer Zipcode               180516 non-null  float64
 20  Department Id                  180519 non-null  int64  
 21  Department Name                180519 non-null  object 
 22  Latitude                       180519 non-null  float64
 23  Longitude                      180519 non-null  float64
 24  Market                         180519 non-null  object 
 25  Order City                     180519 non-null  object 
 26  Order Country                  180519 non-null  object 
 27  Order Customer Id              180519 non-null  int64  
 28  order date (DateOrders)        180519 non-null  object 
 29  Order Id                       180519 non-null  int64  
 30  Order Item Cardprod Id         180519 non-null  int64  
 31  Order Item Discount            180519 non-null  float64
 32  Order Item Discount Rate       180519 non-null  float64
 33  Order Item Id                  180519 non-null  int64  
 34  Order Item Product Price       180519 non-null  float64
 35  Order Item Profit Ratio        180519 non-null  float64
 36  Order Item Quantity            180519 non-null  int64  
 37  Sales                          180519 non-null  float64
 38  Order Item Total               180519 non-null  float64
 39  Order Profit Per Order         180519 non-null  float64
 40  Order Region                   180519 non-null  object 
 41  Order State                    180519 non-null  object 
 42  Order Status                   180519 non-null  object 
 43  Order Zipcode                  24840 non-null   float64
 44  Product Card Id                180519 non-null  int64  
 45  Product Category Id            180519 non-null  int64  
 46  Product Description            0 non-null       float64
 47  Product Image                  180519 non-null  object 
 48  Product Name                   180519 non-null  object 
 49  Product Price                  180519 non-null  float64
 50  Product Status                 180519 non-null  int64  
 51  shipping date (DateOrders)     180519 non-null  object 
 52  Shipping Mode                  180519 non-null  object 
dtypes: float64(15), int64(14), object(24)
memory usage: 73.0+ MB
```

From the above - we can see that there are null values in the following columns

- Order Zipcode
- Project Description

In [17]:
```
scm_df.describe()
```

Out[17]:

	Days for shipping (real)	Days for shipment (scheduled)	Benefit per order	Sales per customer	Late_delivery
count	180519.000000	180519.000000	180519.000000	180519.000000	180519.00
mean	3.497654	2.931847	21.974989	183.107609	0.54
std	1.623722	1.374449	104.433526	120.043670	0.49
min	0.000000	0.000000	-4274.979980	7.490000	0.00
25%	2.000000	2.000000	7.000000	104.379997	0.00
50%	3.000000	4.000000	31.520000	163.990005	1.00
75%	5.000000	4.000000	64.800003	247.399994	1.00
max	6.000000	4.000000	911.799988	1939.989990	1.00

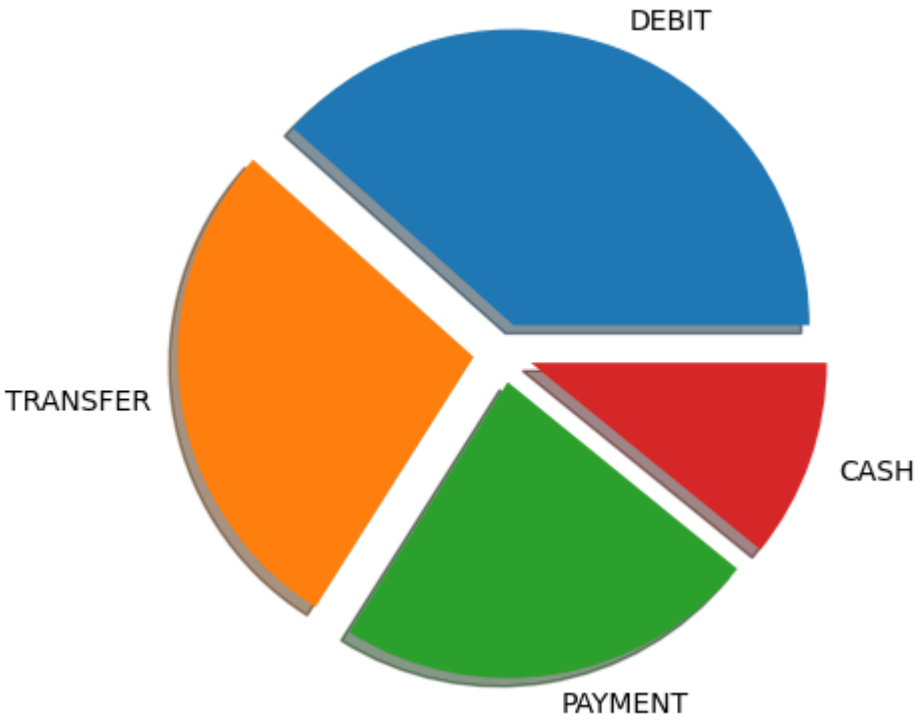
8 rows × 29 columns

In [18]:
```
# Let's see what kind of data is present in the 'Type' Column
scm_df['Type'].value_counts()
```

Out[18]:
```
Type
DEBIT       69295
TRANSFER    49883
PAYMENT     41725
CASH        19616
Name: count, dtype: int64
```

In [19]:
```
# visualizing the same thing in a pie chart

# Trying to create a plot with each section exploded from the pie.
my_explode = [0.1, 0.1, 0.1, 0.1]
my_labels = np.array(['DEBIT', 'TRANSFER', 'PAYMENT', 'CASH'])
type_value_array = (scm_df['Type'].value_counts())
plt.pie(type_value_array, labels=my_labels, shadow=True, explode=my_explo
```

Out[19]:
```
([<matplotlib.patches.Wedge at 0x178b78250>,
  <matplotlib.patches.Wedge at 0x178b782b0>,
  <matplotlib.patches.Wedge at 0x178342df0>,
  <matplotlib.patches.Wedge at 0x178b7ba90>],
 [Text(0.428168266257398, 1.1210137982068438, 'DEBIT'),
  Text(-1.1885215498911712, -0.16557936297826523, 'TRANSFER'),
  Text(0.19342361628901053, -1.1843087877161436, 'PAYMENT'),
  Text(1.130752535999951, -0.40174457349126647, 'CASH')])
```

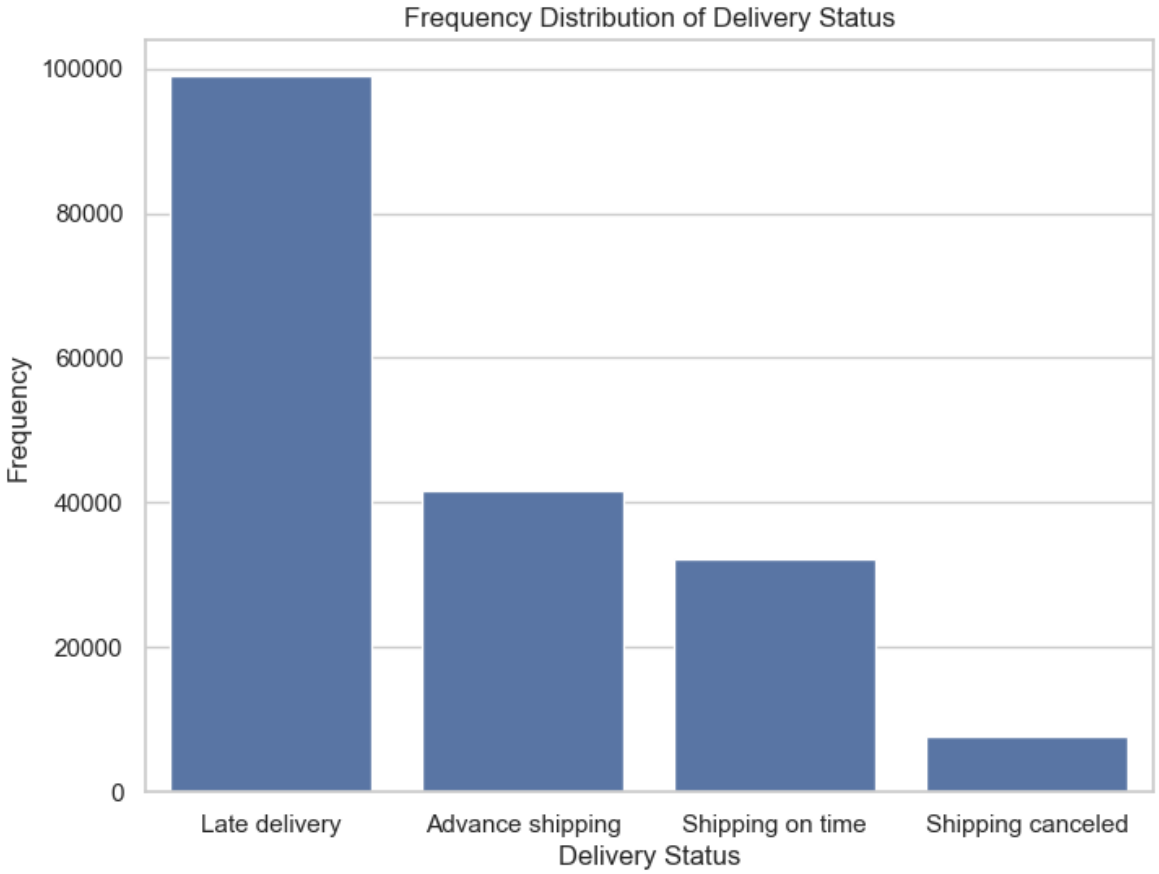


In [20]:
```
# Delivery Status
scm_df['Delivery Status'].value_counts()
```

Out[20]:
```
Delivery Status
Late delivery        98977
Advance shipping     41592
Shipping on time     32196
Shipping canceled     7754
Name: count, dtype: int64
```

In [21]:
```
# plotting it in a graph to visualize it better
sns.set(style="whitegrid")
plt.figure(figsize = (8,6))
sns.barplot(scm_df['Delivery Status'].value_counts())
plt.xlabel("Delivery Status")
plt.ylabel("Frequency")
plt.title("Frequency Distribution of Delivery Status")
```

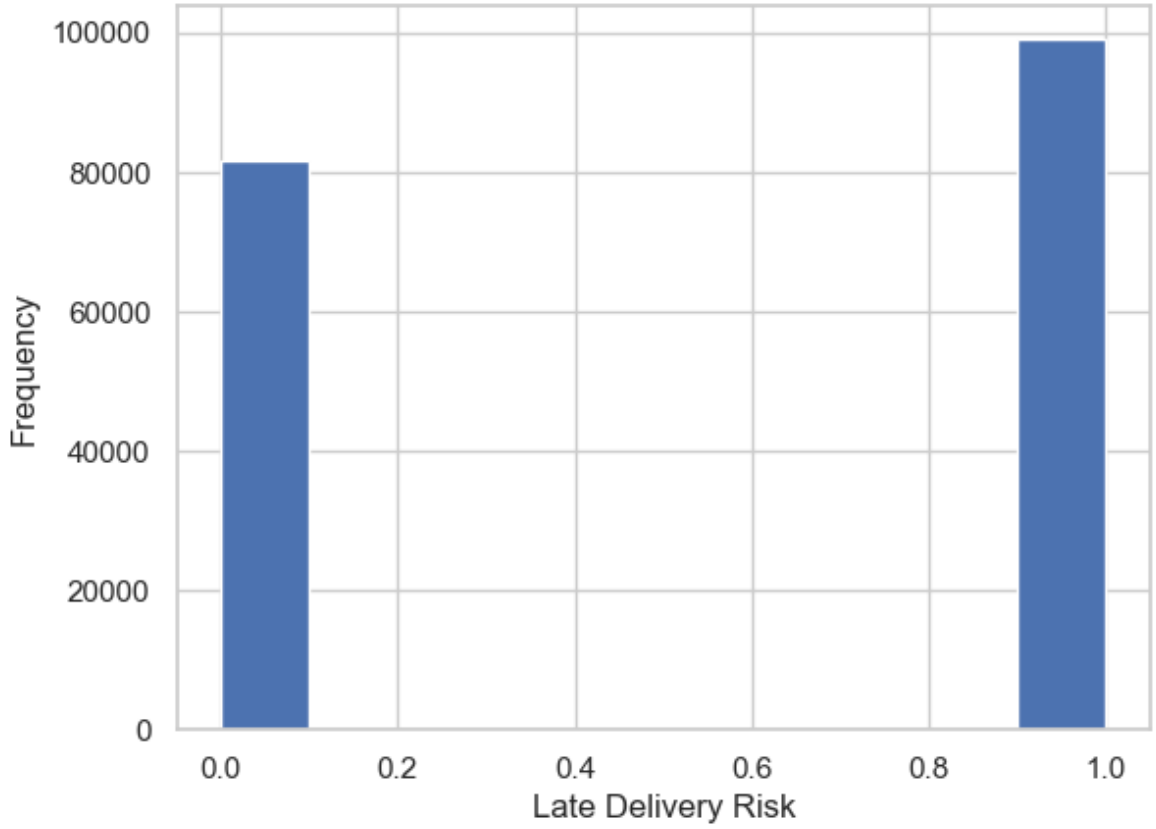
Out[21]:
```
Text(0.5, 1.0, 'Frequency Distribution of Delivery Status')
```



We can see that the Delivery Status for most of the deliveries in the dataset are 'Late delivery'

In [22]:
```
# late delivery risk column
# scm_df['Late_delivery_risk'].value_counts()
scm_df['Late_delivery_risk'].plot.hist(xlabel='Late Delivery Risk')
```

Out[22]:
```
<Axes: xlabel='Late Delivery Risk', ylabel='Frequency'>
```



In [23]:
```
# late delivery risk and delivery status
sns.countplot(scm_df, x='Delivery Status', hue='Late_delivery_risk')
```

Out[23]:
```
<Axes: xlabel='Delivery Status', ylabel='count'>
```

The above graph is counter-intuitive. Only on Late deliveries - we have 'Late delivery risk' as 1, and on all other Delivery statuses, the late delivery risk is 0.

Therefore, while training the model we can remove the Delivery Status column - since it is a redundant version of the 'Late delivery risk' column.

In [24]:
```
# checking which columns have null rows
scm_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180519 entries, 0 to 180518
Data columns (total 53 columns):
 #   Column                         Non-Null Count   Dtype  
---  ------                         --------------   -----  
 0   Type                           180519 non-null  object 
 1   Days for shipping (real)       180519 non-null  int64  
 2   Days for shipment (scheduled)  180519 non-null  int64  
 3   Benefit per order              180519 non-null  float64
 4   Sales per customer             180519 non-null  float64
 5   Delivery Status                180519 non-null  object 
 6   Late_delivery_risk             180519 non-null  int64  
 7   Category Id                    180519 non-null  int64  
 8   Category Name                  180519 non-null  object 
 9   Customer City                  180519 non-null  object 
 10  Customer Country               180519 non-null  object 
 11  Customer Email                 180519 non-null  object 
 12  Customer Fname                 180519 non-null  object 
 13  Customer Id                    180519 non-null  int64  
 14  Customer Lname                 180511 non-null  object 
 15  Customer Password              180519 non-null  object 
 16  Customer Segment               180519 non-null  object 
 17  Customer State                 180519 non-null  object 
 18  Customer Street                180519 non-null  object 
 19  Customer Zipcode               180516 non-null  float64
 20  Department Id                  180519 non-null  int64  
 21  Department Name                180519 non-null  object 
 22  Latitude                       180519 non-null  float64
 23  Longitude                      180519 non-null  float64
 24  Market                         180519 non-null  object 
 25  Order City                     180519 non-null  object 
 26  Order Country                  180519 non-null  object 
 27  Order Customer Id              180519 non-null  int64  
 28  order date (DateOrders)        180519 non-null  object 
 29  Order Id                       180519 non-null  int64  
 30  Order Item Cardprod Id         180519 non-null  int64  
 31  Order Item Discount            180519 non-null  float64
 32  Order Item Discount Rate       180519 non-null  float64
 33  Order Item Id                  180519 non-null  int64  
 34  Order Item Product Price       180519 non-null  float64
 35  Order Item Profit Ratio        180519 non-null  float64
 36  Order Item Quantity            180519 non-null  int64  
 37  Sales                          180519 non-null  float64
 38  Order Item Total               180519 non-null  float64
 39  Order Profit Per Order         180519 non-null  float64
 40  Order Region                   180519 non-null  object 
 41  Order State                    180519 non-null  object 
 42  Order Status                   180519 non-null  object 
 43  Order Zipcode                  24840 non-null   float64
 44  Product Card Id                180519 non-null  int64  
 45  Product Category Id            180519 non-null  int64  
 46  Product Description            0 non-null       float64
 47  Product Image                  180519 non-null  object 
 48  Product Name                   180519 non-null  object 
 49  Product Price                  180519 non-null  float64
 50  Product Status                 180519 non-null  int64  
 51  shipping date (DateOrders)     180519 non-null  object 
 52  Shipping Mode                  180519 non-null  object 
dtypes: float64(15), int64(14), object(24)
memory usage: 73.0+ MB
```

Let us try to find out more about location data of the customers which result in late delivery.

In [25]:
```
# Plotting geographic data of the customer locations which resulted in La
sns.countplot(scm_df, x='Order Region', hue='Delivery Status')
```

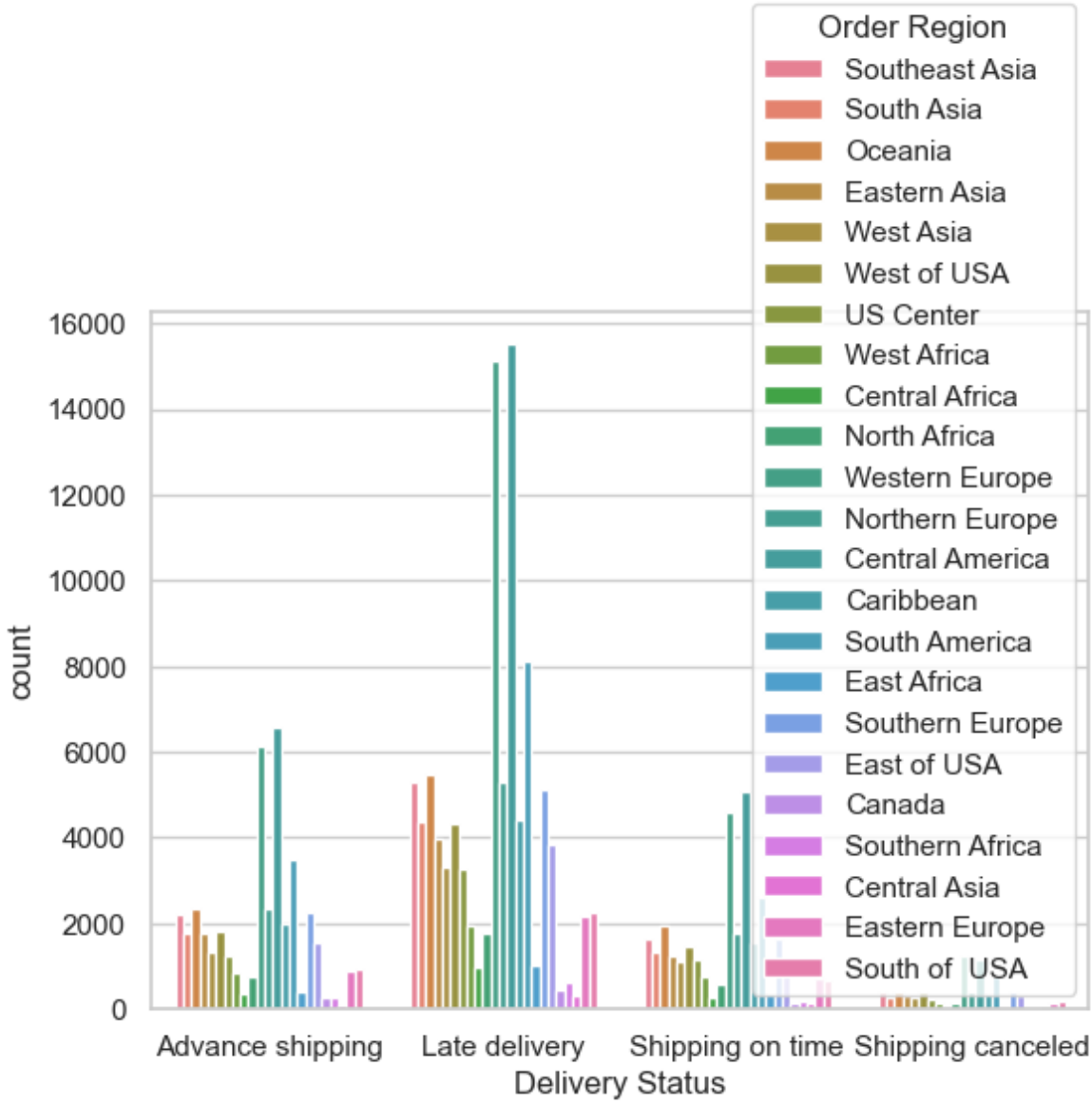
Out[25]:
```
<Axes: xlabel='Order Region', ylabel='count'>
```

The above plot is not useful since we are unable to see the regions on the x-axis

However, we can plot the same data but instead of order region in the x axis & different colors for delivery status, let us consider delivery status on the x-axis and regions with different colors

In [26]:
```
# trying to plot the same figure in a different way
sns.countplot(scm_df, x='Delivery Status', hue='Order Region')
```

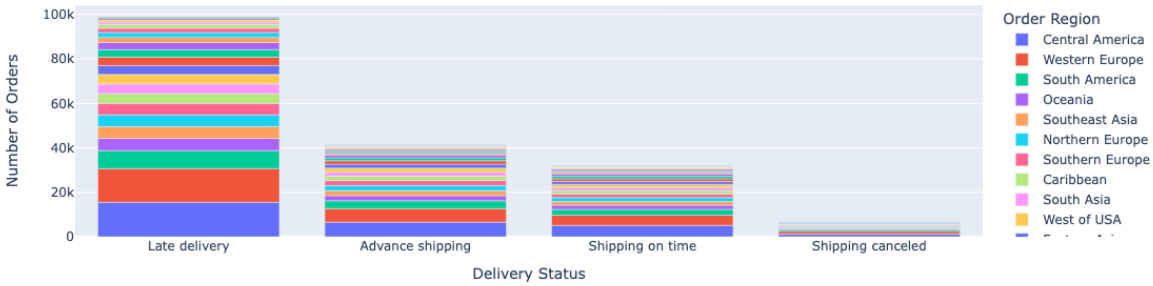
Out[26]:
```
<Axes: xlabel='Delivery Status', ylabel='count'>
```



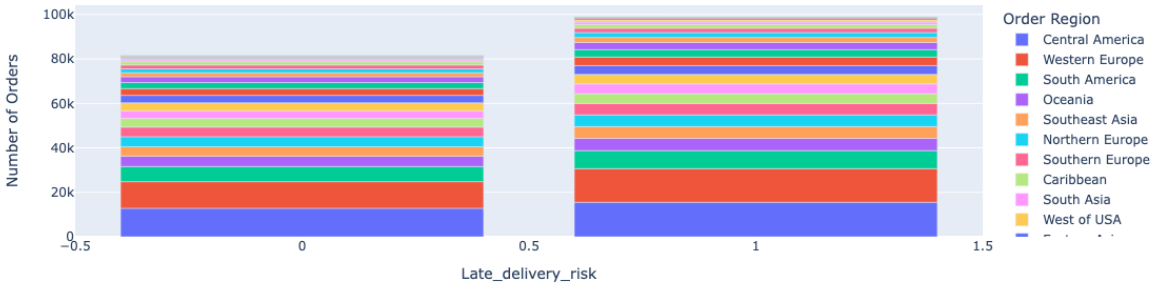
This is better, but it is still hard to read the graph to understand exactly which region has most delivery statuses as 'Late delievry'

Therefore, using the plotly library as shown below in order to have an interactive graph to plot the same data.

In [27]:
```
# displaying the same data using plotly library
data_delivery_status_region=scm_df.groupby(['Delivery Status', 'Order Reg
px.bar(data_delivery_status_region, x='Delivery Status', y='Number of Ord
       )
```



In [28]:
```
# late delivery risk vs order region
data_delivery_status_region=scm_df.groupby(['Late_delivery_risk', 'Order
px.bar(data_delivery_status_region, x='Late_delivery_risk', y='Number of
       )
```



In [29]:
```
# Customer Segment
scm_df['Customer Segment'].value_counts()
```

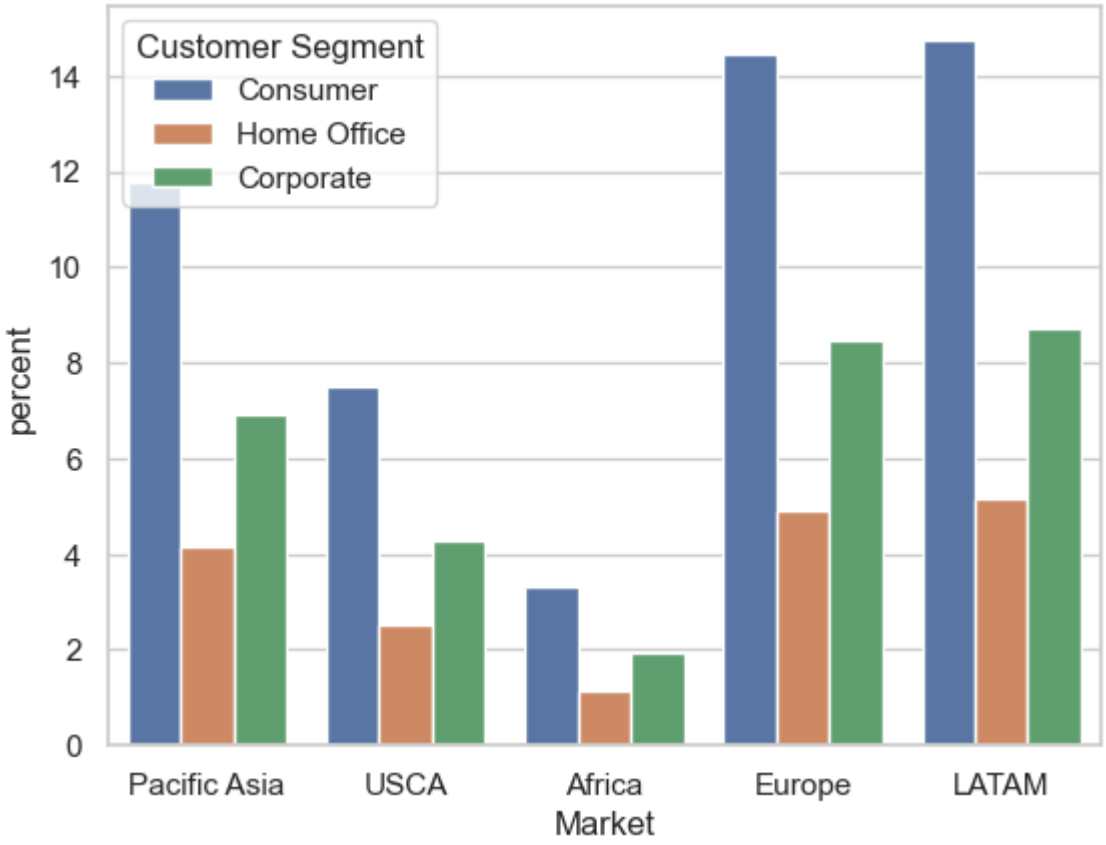
Out[29]:
```
Customer Segment
Consumer       93504
Corporate      54789
Home Office    32226
Name: count, dtype: int64
```

In [30]:
```
# Market
scm_df['Market'].value_counts()
```

Out[30]:
```
Market
LATAM           51594
Europe          50252
Pacific Asia    41260
USCA            25799
Africa          11614
Name: count, dtype: int64
```

In [31]:
```
# In each market - finding the customer segment (Consumer, Home Office an
sns.countplot(scm_df, x="Market", hue="Customer Segment", stat="percent",
```

Out[31]:
```
<Axes: xlabel='Market', ylabel='percent'>
```

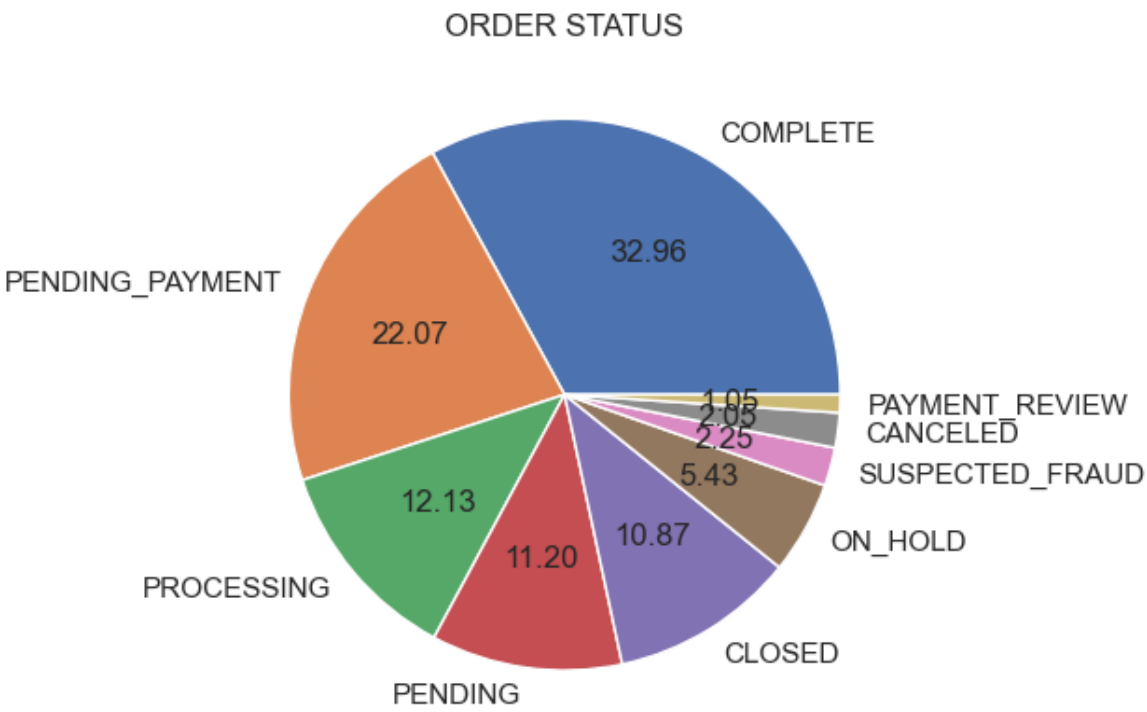


In [32]:
```
# Order Status
scm_df['Order Status'].value_counts()
```

Out[32]:
```
Order Status
COMPLETE           59491
PENDING_PAYMENT    39832
PROCESSING         21902
PENDING            20227
CLOSED             19616
ON_HOLD             9804
SUSPECTED_FRAUD     4062
CANCELED            3692
PAYMENT_REVIEW      1893
Name: count, dtype: int64
```

In [33]:
```
# again visualizing the order status with pie charts
scm_df['Order Status'].value_counts().plot.pie(title='ORDER STATUS',autop
```

Out[33]:
```
<Axes: title={'center': 'ORDER STATUS'}>
```



In [34]:
```
# Product Status
scm_df['Product Status'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 180519 entries, 0 to 180518
Series name: Product Status
Non-Null Count   Dtype
--------------   -----
180519 non-null  int64
dtypes: int64(1)
memory usage: 1.4 MB
```

In [35]:
```
scm_df['Product Status'].value_counts()
```

Out[35]:
```
Product Status
0    180519
Name: count, dtype: int64
```

Product Status can be 1 - not available and 0 available. In this dataset, we can see that all the values are 0 (available).

Therefore this data is not useful to us, and we can drop this column.

In [36]:
```
# Shipping Mode
scm_df['Shipping Mode'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 180519 entries, 0 to 180518
Series name: Shipping Mode
Non-Null Count   Dtype
--------------   -----
180519 non-null  object
dtypes: object(1)
memory usage: 1.4+ MB
```

In [37]:
```
scm_df['Shipping Mode'].describe()
```

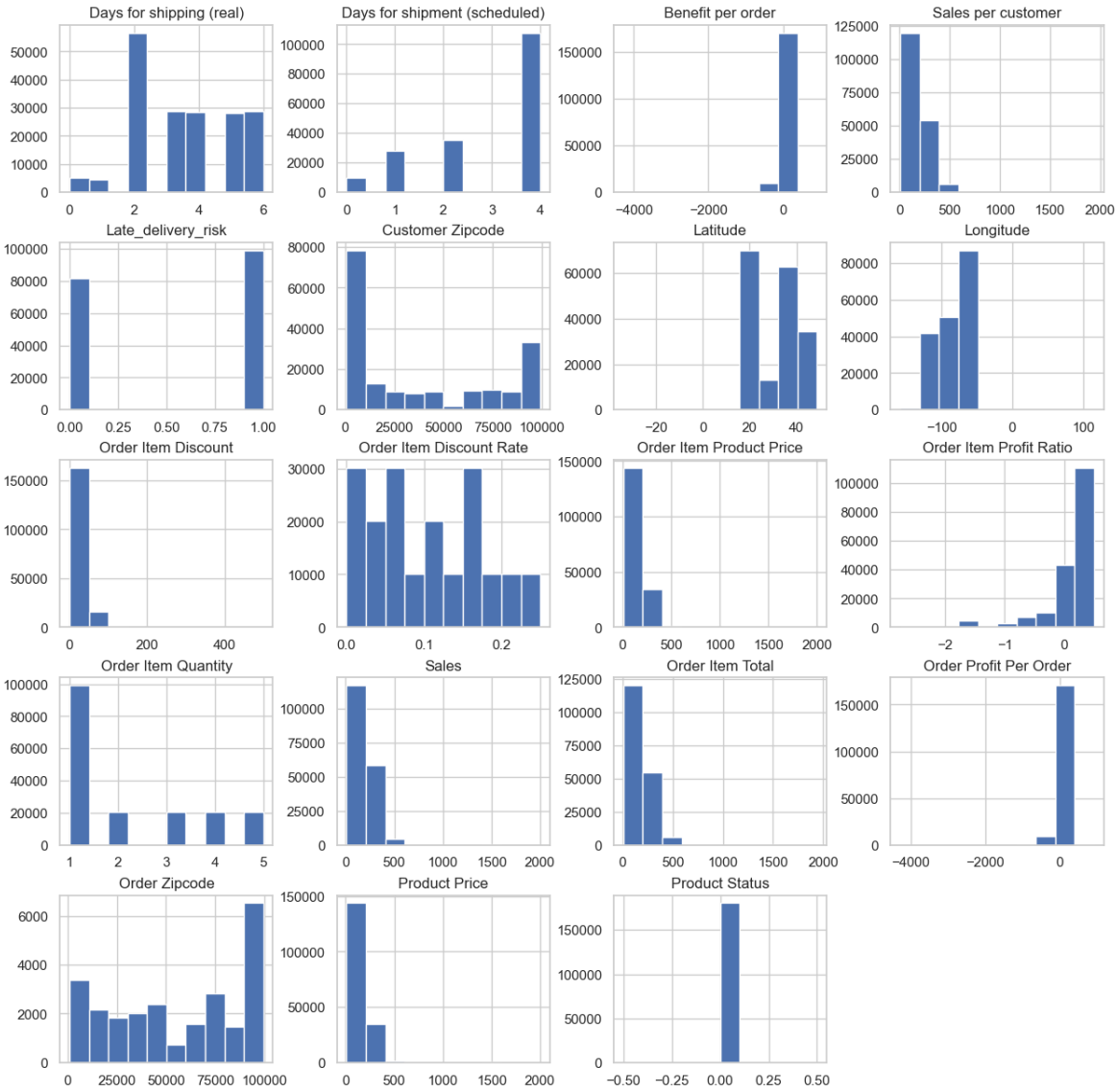
Out[37]:
```
count             180519
unique                 4
top       Standard Class
freq              107752
Name: Shipping Mode, dtype: object
```

In [38]:
```
scm_df['Shipping Mode'].value_counts()
```

Out[38]:
```
Shipping Mode
Standard Class    107752
Second Class       35216
First Class        27814
Same Day            9737
Name: count, dtype: int64
```

In [39]:
```
# plotting a histogram of all the columns

# creating a temporary dataframe for visualizing only those columns which
scm_df_temp_visualization = scm_df.drop(columns=['Category Id',
                                                 'Customer Id',
                                                 'Department Id',
                                                 'Order Customer Id',
                                                 'Order Id',
                                                 'Order Item Cardprod Id'
                                                 'Order Item Id',
                                                 'Product Card Id',
                                                 'Product Category Id',
                                                 'Product Description'])
scm_df_temp_visualization.hist(figsize=(15,15))
```

Out[39]:
```
array([[<Axes: title={'center': 'Days for shipping (real)'}>,
        <Axes: title={'center': 'Days for shipment (scheduled)'}>,
        <Axes: title={'center': 'Benefit per order'}>,
        <Axes: title={'center': 'Sales per customer'}>],
       [<Axes: title={'center': 'Late_delivery_risk'}>,
        <Axes: title={'center': 'Customer Zipcode'}>,
        <Axes: title={'center': 'Latitude'}>,
        <Axes: title={'center': 'Longitude'}>],
       [<Axes: title={'center': 'Order Item Discount'}>,
        <Axes: title={'center': 'Order Item Discount Rate'}>,
        <Axes: title={'center': 'Order Item Product Price'}>,
        <Axes: title={'center': 'Order Item Profit Ratio'}>],
       [<Axes: title={'center': 'Order Item Quantity'}>,
        <Axes: title={'center': 'Sales'}>,
        <Axes: title={'center': 'Order Item Total'}>,
        <Axes: title={'center': 'Order Profit Per Order'}>],
       [<Axes: title={'center': 'Order Zipcode'}>,
        <Axes: title={'center': 'Product Price'}>,
        <Axes: title={'center': 'Product Status'}>, <Axes: >]],
      dtype=object)
```



In [40]:
```
print(*scm_df['Category Name'].unique(), sep='\n')
```

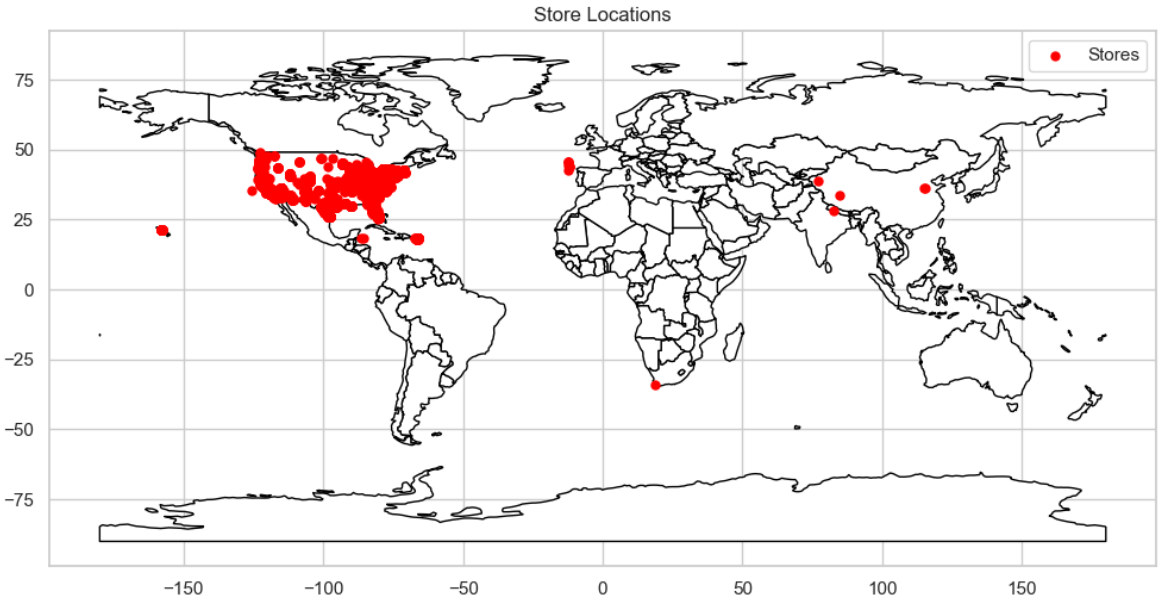
```
Sporting Goods
Cleats
Shop By Sport
Women's Apparel
Electronics
Boxing & MMA
Cardio Equipment
Trade-In
Kids' Golf Clubs
Hunting & Shooting
Baseball & Softball
Men's Footwear
Camping & Hiking
Consumer Electronics
Cameras
Computers
Basketball
Soccer
Girls' Apparel
Accessories
Women's Clothing
Crafts
Men's Clothing
Tennis & Racquet
Fitness Accessories
As Seen on  TV!
Golf Balls
Strength Training
Children's Clothing
Lacrosse
Baby
Fishing
Books
DVDs
CDs
Garden
Hockey
Pet Supplies
Health and Beauty
Music
Video Games
Golf Gloves
Golf Bags & Carts
Golf Shoes
Golf Apparel
Women's Golf Clubs
Men's Golf Clubs
Toys
Water Sports
Indoor/Outdoor Games
```

In [41]:
```
# using the latitude & longitude values from store to plot it on the worl
import geopandas as gpd

# dropping the duplicates and null rows
df_stores_location_lat_n_long = scm_df[['Latitude', 'Longitude']].drop_du

df_geo = gpd.GeoDataFrame(
    df_stores_location_lat_n_long,
    geometry=gpd.points_from_xy(df_stores_location_lat_n_long['Longitude'
```

```
)

#get the map image
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))

#plot the world map
ax = world.plot(figsize=(12, 8), color='white', edgecolor='black')

#plot the store locations as dots
df_geo.plot(ax=ax, marker='o', color='red', markersize=25, label='Stores'

plt.title('Store Locations')
plt.legend(loc='upper right')
```

```
/var/folders/3m/tt_kbz7d2tx3f5cx3v4nd4940000gn/T/ipykernel_76908/282516111
0.py:13: FutureWarning:

The geopandas.dataset module is deprecated and will be removed in GeoPanda
s 1.0. You can get the original 'naturalearth_lowres' data from https://ww
w.naturalearthdata.com/downloads/110m-cultural-vectors/.
```

Out[41]: <matplotlib.legend.Legend at 0x292e735e0>



Model Preparation

Let us start with removing the following columns

1. ID Columns:
    - 'Category Id',
    - 'Customer Id',
    - 'Department Id',
    - 'Order Customer Id',
    - 'Order Id',
    - 'Order Item Cardprod Id',
    - 'Order Item Id',
    - 'Product Card Id',
    - 'Product Category Id',

2. Irrelevant Non-numeric Columns
    - 'Product Description'
    - 'Customer Email'
    - 'Customer Password'
    - 'Customer Fname'
    - 'Customer Lname'
    - 'Customer State'
    - 'Customer Street'
    - 'Customer Zipcode'
    - Order Region
    - Order State
    - Order Status
    - Order Zipcode
    - Product Image
3. Redundant Columns:
    - 'Delivery Status'
    - 'Product Status'
    - 'Customer City'
    - 'Customer Country' (since we already have Order City and Order Country)

In [42]:
```
from sklearn import svm, metrics, tree, preprocessing, linear_model
```

In [43]:
```
scm_clean_df = scm_df.drop(columns=['Category Id',
                                    'Customer Id',
                                    'Department Id',
                                    'Order Customer Id',
                                    'Order Id',
                                    'Order Item Cardprod Id',
                                    'Order Item Id',
                                    'Product Card Id',
                                    'Product Category Id',
                                    'Product Description',
                                    'Delivery Status',
                                    'Customer Email',
                                    'Customer Password',
                                    'Product Status',
                                    'Customer Fname',
                                    'Customer Lname',
                                    'Customer State',
                                    'Customer Street',
                                    'Customer Zipcode',
                                    'Customer City',
                                    'Customer Country',
                                    'Order Region',
                                    'Order State',
                                    'Order Status',
                                    'Order Zipcode',
                                    'Product Image'])
```

Text Columns with Categorical Data

- Type
- Category Name
- Customer Segment
- Department Name
- Market
- Order City
- Order Country
- order date (DateOrders)
- Product Name
- shipping date (DateOrders)
- Shipping Mode

In [44]:
```
# need to use LabelEncoders to convert the categorical text values
label_encoder = preprocessing.LabelEncoder()
```

In [45]:
```
scm_clean_df['Type'] = label_encoder.fit_transform(scm_clean_df['Type'])
scm_clean_df['Category Name'] = label_encoder.fit_transform(scm_clean_df[
scm_clean_df['Customer Segment'] = label_encoder.fit_transform(scm_clean_
scm_clean_df['Department Name'] = label_encoder.fit_transform(scm_clean_d
scm_clean_df['Market'] = label_encoder.fit_transform(scm_clean_df['Market
scm_clean_df['Order City'] = label_encoder.fit_transform(scm_clean_df['Or
scm_clean_df['Order Country'] = label_encoder.fit_transform(scm_clean_df[
scm_clean_df['order date (DateOrders)'] = label_encoder.fit_transform(scm
scm_clean_df['Product Name'] = label_encoder.fit_transform(scm_clean_df['
scm_clean_df['shipping date (DateOrders)'] = label_encoder.fit_transform(
scm_clean_df['Shipping Mode'] = label_encoder.fit_transform(scm_clean_df[
```

In [46]:
```
scm_clean_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180519 entries, 0 to 180518
Data columns (total 27 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   Type                            180519 non-null  int64  
 1   Days for shipping (real)        180519 non-null  int64  
 2   Days for shipment (scheduled)   180519 non-null  int64  
 3   Benefit per order               180519 non-null  float64
 4   Sales per customer              180519 non-null  float64
 5   Late_delivery_risk              180519 non-null  int64  
 6   Category Name                   180519 non-null  int64  
 7   Customer Segment                180519 non-null  int64  
 8   Department Name                 180519 non-null  int64  
 9   Latitude                        180519 non-null  float64
 10  Longitude                       180519 non-null  float64
 11  Market                          180519 non-null  int64  
 12  Order City                      180519 non-null  int64  
 13  Order Country                   180519 non-null  int64  
 14  order date (DateOrders)         180519 non-null  int64  
 15  Order Item Discount             180519 non-null  float64
 16  Order Item Discount Rate        180519 non-null  float64
 17  Order Item Product Price        180519 non-null  float64
 18  Order Item Profit Ratio         180519 non-null  float64
 19  Order Item Quantity             180519 non-null  int64  
 20  Sales                           180519 non-null  float64
 21  Order Item Total                180519 non-null  float64
 22  Order Profit Per Order          180519 non-null  float64
 23  Product Name                    180519 non-null  int64  
 24  Product Price                   180519 non-null  float64
 25  shipping date (DateOrders)      180519 non-null  int64  
 26  Shipping Mode                   180519 non-null  int64  
dtypes: float64(12), int64(15)
memory usage: 37.2 MB
```

Now all the values are in either int or float.

Splitting Train & Test Data

In [47]:
```
# Next creating the train and test set

# creating a function which splits the data randomly
def split_train_test(data, test_ratio):
    np.random.seed(23) # setting the random number generator's seed will
    shuffled_indices = np.random.permutation(len(data))
    test_set_size = int(len(data) * test_ratio)
    test_indices = shuffled_indices[:test_set_size]
    train_indices = shuffled_indices[test_set_size:]
    return data.iloc[train_indices], data.iloc[test_indices]
```

In [48]:
```
# here we are splitting the data into 80% training and 20% testing sets.
scm_trainset, scm_testset = split_train_test(scm_clean_df, 0.2)
```

In [49]:
```
len(scm_trainset)
```

Out[49]: 144416

In [50]:
```
len(scm_testset)
```

Out[50]:
```
36103
```

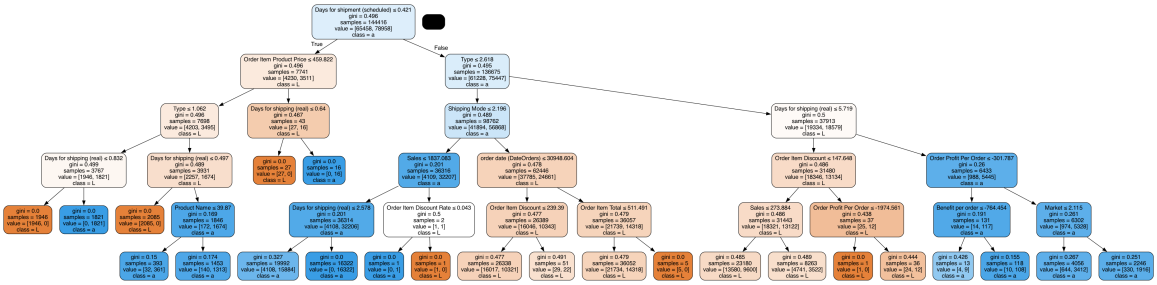
Decision Tree Model #1

In [51]:
```
from sklearn.tree import DecisionTreeClassifier
y = scm_trainset["Late_delivery_risk"]
X = scm_trainset.drop(columns="Late_delivery_risk")

tree_clf = DecisionTreeClassifier(max_depth=5, max_features="sqrt", rando
tree_clf.fit(X, y)
```

Out[51]:
```
DecisionTreeClassifier(max_depth=5, max_features='sqrt', random_s
tate=23,
                       splitter='random')
```

In [52]:
```
from sklearn.tree import export_graphviz
import pydotplus
# from sklearn.externals.six import StringIO
from six import StringIO
from IPython.display import Image

dot_data = StringIO()
export_graphviz(
        tree_clf,
        out_file=dot_data,
        feature_names= X.columns.values.tolist(),
        class_names= "Late_delivery_risk",
        rounded=True,
        filled=True,
        special_characters=True
    )
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('decision_tree_model_1.png')
Image(graph.create_png())
```

Out[52]:



In [53]:
```
# The decision tree model has been trained. We can now test it using the

predictions = tree_clf.predict(X)
predictions
```

Out[53]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [54]:
```
# Using mean sequared error to calculate the error percentage
from sklearn.metrics import mean_squared_error
tree_mean_sq_error = mean_squared_error(y, predictions)
tree_root_mean_sq_err = np.sqrt(tree_mean_sq_error)
tree_root_mean_sq_err
```

Out[54]:
```
0.5460651806110116
```

Looks like the model is not able to predict the values well. The % of error is 54.6%, which means the train accuracy is 46.4%.

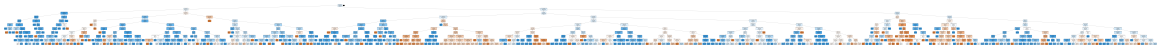
Decision Tree Model #2

In [55]:
```
# Training the a second model on different parameters

tree_clf = DecisionTreeClassifier(max_depth=10, max_features="log2", rand
tree_clf.fit(X, y)
```

Out[55]:
```
DecisionTreeClassifier(max_depth=10, max_features='log2', random_
state=5,
                       splitter='random')
```

In [56]:
```
from sklearn.tree import export_graphviz
import pydotplus
# from sklearn.externals.six import StringIO
from six import StringIO
from IPython.display import Image

dot_data = StringIO()
export_graphviz(
        tree_clf,
        out_file=dot_data,
        feature_names= X.columns.values.tolist(),
        class_names= "Late_delivery_risk",
        rounded=True,
        filled=True,
        special_characters=True
    )
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('decision_tree_model_2.png')
Image(graph.create_png())
```

```
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.658726 to
fit

dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.658726 to
fit
```

Out[56]:


In [57]:
```
predictions = tree_clf.predict(X)
predictions
```

Out[57]:
```
array([0, 0, 0, ..., 0, 1, 0])
```

In [58]:
```
tree_mean_sq_error = mean_squared_error(y, predictions)
tree_root_mean_sq_err = np.sqrt(tree_mean_sq_error)
tree_root_mean_sq_err
```

Out[58]:
```
0.48176791227108456
```

This has helped us improve the model from 46% to 52%.

In [59]:
```
y_test = scm_testset['Late_delivery_risk']
X_test = scm_testset.drop(columns=['Late_delivery_risk'])
test_predictions = tree_clf.predict(X_test)
test_predictions
```

Out[59]:
```
array([1, 1, 0, ..., 0, 1, 0])
```

In [60]:
```
tree_mean_sq_error = mean_squared_error(y_test, test_predictions)
tree_root_mean_sq_err = np.sqrt(tree_mean_sq_error)
tree_root_mean_sq_err
```

Out[60]:
```
0.48137915965998845
```

We can see both train and test accuracy is around 52%.

Mid Term Project finishes here.

Final Project begins here.

Revisiting The Data

There are some columns which can benefit from One Hot Encoding rather than Label Encoding: These columns are as follows:

In [61]:
```
scm_updated = scm_df
```

In [62]:
```
# just like above, let us remove the columns which are not needed
scm_updated.drop(columns=['Category Id',
                          'Customer Id',
                          'Department Id',
                          'Order Customer Id',
                          'Order Id',
                          'Order Item Cardprod Id',
                          'Order Item Id',
                          'Product Card Id',
                          'Product Category Id',
```

```
                                    'Product Description',
                                    'Delivery Status',
                                    'Customer Email',
                                    'Customer Password',
                                    'Product Status',
                                    'Customer Fname',
                                    'Customer Lname',
                                    'Order Status',
                                    'Product Image'],
                                inplace=True)
```

In [63]:
```
scm_updated['Type'].value_counts()
```

Out[63]:
```
Type
DEBIT       69295
TRANSFER    49883
PAYMENT     41725
CASH        19616
Name: count, dtype: int64
```

In [64]:
```
scm_updated['Customer Segment'].value_counts()
```

Out[64]:
```
Customer Segment
Consumer       93504
Corporate      54789
Home Office    32226
Name: count, dtype: int64
```

In [65]:
```
scm_updated['Department Name'].value_counts()
```

Out[65]:
```
Department Name
Fan Shop              66861
Apparel               48998
Golf                  33220
Footwear              14525
Outdoors               9686
Fitness                2479
Discs Shop             2026
Technology             1465
Pet Shop                492
Book Shop               405
Health and Beauty       362
Name: count, dtype: int64
```

In [66]:
```
scm_updated['Market'].value_counts()
```

Out[66]:
```
Market
LATAM           51594
Europe          50252
Pacific Asia    41260
USCA            25799
Africa          11614
Name: count, dtype: int64
```

In [67]:
```
scm_updated['Product Name'].value_counts()
```

Out[67]:
```
Product Name
Perfect Fitness Perfect Rip Deck                 24515
Nike Men's CJ Elite 2 TD Football Cleat          22246
Nike Men's Dri-FIT Victory Golf Polo             21035
O'Brien Men's Neoprene Life Vest                 19298
Field & Stream Sportsman 16 Gun Fire Safe        17325
                                                 ...
Stiga Master Series ST3100 Competition Indoor       27
SOLE E35 Elliptical                                 15
Bushnell Pro X7 Jolt Slope Rangefinder              11
Bowflex SelectTech 1090 Dumbbells                   10
SOLE E25 Elliptical                                 10
Name: count, Length: 118, dtype: int64
```

In [68]:
```
scm_updated['Shipping Mode'].value_counts()
```

Out[68]:
```
Shipping Mode
Standard Class    107752
Second Class       35216
First Class        27814
Same Day            9737
Name: count, dtype: int64
```

One Hot Encoding

We can use One Hot Encoding for the following columns:

1. Type
2. Delivery Status
3. Customer Segment
4. Market
5. Shipping Mode

In [69]:
```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

label_encoder_type = LabelEncoder()
one_hot_encoder = OneHotEncoder(sparse=False)
```

In [70]:
```
# First performing encoding on 'Type' column
# before converting to One Hot Encoding, we are using Label Encoding
type_integer_encoded = label_encoder_type.fit_transform(scm_updated['Type
# initially the shape of type_integer_encoded is (180519,)
# therefore, we need to reshape it to (180519,1)
type_integer_encoded = type_integer_encoded.reshape(len(type_integer_enco
type_integer_encoded
```

Out[70]:
```
array([[1],
       [3],
       [0],
       ...,
       [3],
       [2],
       [2]])
```

In [71]:
```
# Let us see these categories of the Label Encoding
label_encoder_type.classes_
```

Out[71]:
```
array(['CASH', 'DEBIT', 'PAYMENT', 'TRANSFER'], dtype=object)
```

From the above we can see that the label encoding has converted the text to numbers in the following way:

0: CASH

1: DEBIT

2: PAYMENT

3: TRANSFER

In [72]:
```
# Now converting the Label Encoded values into One Hot Encoding
type_one_hot = one_hot_encoder.fit_transform(type_integer_encoded)
print(type_one_hot)
print() # printing new line char
print(type_one_hot.shape)
```

```
[[0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]
 ...
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]
 [0. 0. 1. 0.]]

(180519, 4)
```

Each of the columns represent the respective Label Encoder. Therefore:

- Column 0 represents CASH
- Column 1 represents DEBIT
- Column 2 represents PAYMENT
- Column 3 represents TRANSFER

Issue with One Hot Encoding: Multi-Collinearity

Many times when doing One Hot Encoding, the issue of Dummy Variable trap comes to the picture - which further leads to the problem of multi-collinearity.

For the 'Type' column above (and others), we will check for multi-collinearity using the Variance Inflation Factor (VIF) score.

In [73]:
```
# importing a library for VIF
from statsmodels.stats.outliers_influence import variance_inflation_facto
```

In [74]:
```
# defining a function which calculates VIF for all the columns
def calculate_vif(X):
    vif = pd.DataFrame()
    vif["Columns"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X
    return(vif)
```

In [75]:
```
# first converting the one hot encoded value to pandas df object
calculate_vif(pd.DataFrame(type_one_hot))
```

Out[75]:

	Columns	VIF
0	0	1.0
1	1	1.0
2	2	1.0
3	3	1.0

We can see that all the four columns have VIF Value = 1. This means that all these columns can be added to the pandas data frame.

In [76]:
```
scm_updated[['Cash Type', 'Debit Type', 'Payment Type', 'Transfer Type']]
```

In [77]:
```
# Verifying that the column names in the one hot encoding is correct
scm_updated.head(10)
```

Out[77]:

	Type	Days for shipping (real)	Days for shipment (scheduled)	Benefit per order	Sales per customer	Late_delivery_risk
0	DEBIT	3	4	91.250000	314.640015	0
1	TRANSFER	5	4	-249.089996	311.359985	1
2	CASH	4	4	-247.779999	309.720001	0
3	DEBIT	3	4	22.860001	304.809998	0
4	PAYMENT	2	4	134.210007	298.250000	0
5	TRANSFER	6	4	18.580000	294.980011	0
6	DEBIT	2	1	95.180000	288.420013	1
7	TRANSFER	2	1	68.430000	285.140015	1
8	CASH	3	2	133.720001	278.589996	1
9	CASH	2	1	132.149994	275.309998	1

10 rows × 39 columns

In [78]:
```
# And finally, we can now drop the Type column.
scm_updated = scm_updated.drop(columns=['Type'])
```

Similar to the above process, we will do the One Hot Encoding for the other columns as well.

In [79]:
```
label_encoder_cust_seg = LabelEncoder()
label_encoder_market = LabelEncoder()
label_encoder_ship_mode = LabelEncoder()
```

In [80]:
```
# label encoder & one hot encoder for - customer segment
cust_seg_integer_encoded = label_encoder_cust_seg.fit_transform(scm_updat
cust_seg_integer_encoded = cust_seg_integer_encoded.reshape(len(cust_seg_
cust_seg_one_hot = one_hot_encoder.fit_transform(cust_seg_integer_encoded

# label encoder & one hot encoder for - market
market_integer_encoded = label_encoder_market.fit_transform(scm_updated['
market_integer_encoded = market_integer_encoded.reshape(len(market_intege
market_one_hot = one_hot_encoder.fit_transform(market_integer_encoded)

# label encoder & one hot encoder for - shipping mode
ship_mode_integer_encoded = label_encoder_ship_mode.fit_transform(scm_upd
ship_mode_integer_encoded = ship_mode_integer_encoded.reshape(len(ship_mo
ship_mode_one_hot = one_hot_encoder.fit_transform(ship_mode_integer_encod
```

In [81]:
```
# now checking the vif for each of the above

print('VIF for Customer Segment')
print(calculate_vif(pd.DataFrame(cust_seg_one_hot)))

print('VIF for Market')
print(calculate_vif(pd.DataFrame(market_one_hot)))

print('VIF for Shipping Mode')
print(calculate_vif(pd.DataFrame(ship_mode_one_hot)))
```

```
VIF for Customer Segment
   Columns  VIF
0        0  1.0
1        1  1.0
2        2  1.0
VIF for Market
   Columns  VIF
0        0  1.0
1        1  1.0
2        2  1.0
3        3  1.0
4        4  1.0
VIF for Shipping Mode
   Columns  VIF
0        0  1.0
1        1  1.0
2        2  1.0
3        3  1.0
```

Surprisingly the VIF values for all the other columns are also 1.0. Therefore, we can add them all into the pandas data frame.

In [82]:
```
# Customer Segment Encoding
print(label_encoder_cust_seg.classes_)
```

```
['Consumer' 'Corporate' 'Home Office']
```

In [83]:
```
scm_updated[['Consumer Customer Segment', 'Corporate Customer Segment', '
```

In [84]:
```
scm_updated[['Customer Segment', 'Consumer Customer Segment', 'Corporate
```

Out[84]:

	Customer Segment	Consumer Customer Segment	Corporate Customer Segment	Home Office Customer Segment
0	Consumer	1.0	0.0	0.0
1	Consumer	1.0	0.0	0.0
2	Consumer	1.0	0.0	0.0
3	Home Office	0.0	0.0	1.0
4	Corporate	0.0	1.0	0.0
5	Consumer	1.0	0.0	0.0
6	Home Office	0.0	0.0	1.0
7	Corporate	0.0	1.0	0.0
8	Corporate	0.0	1.0	0.0
9	Corporate	0.0	1.0	0.0

In [85]:
```
scm_updated = scm_updated.drop(columns=['Customer Segment'])
```

In [86]:
```
# Market Encoding
print(label_encoder_market.classes_)
```

```
['Africa' 'Europe' 'LATAM' 'Pacific Asia' 'USCA']
```

In [87]:
```
scm_updated[['Africa Market', 'Europe Market', 'LATAM Market', 'Pacific A
```

In [88]:
```
scm_updated = scm_updated.drop(columns=['Market'])
```

In [89]:
```
# Shipping Mode Encoding
print(label_encoder_ship_mode.classes_)
```

```
['First Class' 'Same Day' 'Second Class' 'Standard Class']
```

In [90]:
```
scm_updated[['First Class Shipping Mode','Same Day Shipping Mode','Second
```

In [91]:
```
scm_updated = scm_updated.drop(columns=['Shipping Mode'])
```

In [92]:
```
# Let's see how the data frame looks like now
scm_updated.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180519 entries, 0 to 180518
Data columns (total 47 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   Days for shipping (real)        180519 non-null  int64  
 1   Days for shipment (scheduled)   180519 non-null  int64  
 2   Benefit per order               180519 non-null  float64
 3   Sales per customer              180519 non-null  float64
 4   Late_delivery_risk              180519 non-null  int64  
 5   Category Name                   180519 non-null  object 
 6   Customer City                   180519 non-null  object 
 7   Customer Country                180519 non-null  object 
 8   Customer State                  180519 non-null  object 
 9   Customer Street                 180519 non-null  object 
 10  Customer Zipcode                180516 non-null  float64
 11  Department Name                 180519 non-null  object 
 12  Latitude                        180519 non-null  float64
 13  Longitude                       180519 non-null  float64
 14  Order City                      180519 non-null  object 
 15  Order Country                   180519 non-null  object 
 16  order date (DateOrders)         180519 non-null  object 
 17  Order Item Discount             180519 non-null  float64
 18  Order Item Discount Rate        180519 non-null  float64
 19  Order Item Product Price        180519 non-null  float64
 20  Order Item Profit Ratio         180519 non-null  float64
 21  Order Item Quantity             180519 non-null  int64  
 22  Sales                           180519 non-null  float64
 23  Order Item Total                180519 non-null  float64
 24  Order Profit Per Order          180519 non-null  float64
 25  Order Region                    180519 non-null  object 
 26  Order State                     180519 non-null  object 
 27  Order Zipcode                   24840 non-null   float64
 28  Product Name                    180519 non-null  object 
 29  Product Price                   180519 non-null  float64
 30  shipping date (DateOrders)      180519 non-null  object 
 31  Cash Type                       180519 non-null  float64
 32  Debit Type                      180519 non-null  float64
 33  Payment Type                    180519 non-null  float64
 34  Transfer Type                   180519 non-null  float64
 35  Consumer Customer Segment       180519 non-null  float64
 36  Corporate Customer Segment      180519 non-null  float64
 37  Home Office Customer Segment    180519 non-null  float64
 38  Africa Market                   180519 non-null  float64
 39  Europe Market                   180519 non-null  float64
 40  LATAM Market                    180519 non-null  float64
 41  Pacific Asia Market             180519 non-null  float64
 42  USCA Market                     180519 non-null  float64
 43  First Class Shipping Mode       180519 non-null  float64
 44  Same Day Shipping Mode          180519 non-null  float64
 45  Second Class Shipping Mode      180519 non-null  float64
 46  Standard Class Shipping Mode    180519 non-null  float64
dtypes: float64(30), int64(4), object(13)
memory usage: 64.7+ MB
```

Binary Encoding

Let us consider the other columns:

- Category Name
- Department Name
- Product Name

For the columns listed above, there are too many categories of data. This makes it unreasonable to use One Hot Encoding as the model will become highly complex. Furthermore, the values in this column do not have any inherent order. This means that Label Encoders will also not be a good idea. For this project I have decided to explore the use of Binary Encoders as a solution to encode the above columns.

In [93]:
```
# Let us explore the other columns
print(scm_updated[['Category Name']].nunique())
print()
print(scm_updated[['Department Name']].nunique())
print()
print(scm_updated[['Product Name']].nunique())
```

```
Category Name    50
dtype: int64

Department Name    11
dtype: int64

Product Name    118
dtype: int64
```

We can see that there are a total of 50 unique category names, 11 unique department names, and 118 unique product names in this dataset.

In [94]:
```
# importing category_encoders
import category_encoders as ce
```

In [95]:
```
binary_encoder_cat_name = ce.BinaryEncoder(cols=['Category Name'], return
cat_name_binary_encoded_data = binary_encoder_cat_name.fit_transform(scm_
cat_name_binary_encoded_data
```

Out[95]:

	Category Name_0	Category Name_1	Category Name_2	Category Name_3	Category Name_4	Category Name_5
0	0	0	0	0	0	1
1	0	0	0	0	0	1
2	0	0	0	0	0	1
3	0	0	0	0	0	1
4	0	0	0	0	0	1
...	...	...	...	...	...	...
180514	1	0	0	0	0	0
180515	1	0	0	0	0	0
180516	1	0	0	0	0	0
180517	1	0	0	0	0	0
180518	1	0	0	0	0	0

180519 rows × 6 columns

Converting 50 (decimal) to binary will result in 110010. This requires 6 columns (since there are 6 bits) to represnt all the data.
Practically, from the above we can see that the binary encoding has created 6 columns - column 0 to column 5.

In [96]:
```
scm_updated[['Category Name 0', 'Category Name 1', 'Category Name 2', 'Ca
scm_updated[['Category Name','Category Name 0', 'Category Name 1', 'Categ
```

Out[96]:

	Category Name	Category Name 0	Category Name 1	Category Name 2	Category Name 3	Category Name 4	Category Name 5
0	Sporting Goods	0	0	0	0	0	1
1	Sporting Goods	0	0	0	0	0	1
2	Sporting Goods	0	0	0	0	0	1
3	Sporting Goods	0	0	0	0	0	1
4	Sporting Goods	0	0	0	0	0	1
...	...	...	...	...	...	...	...
180504	Fishing	1	0	0	0	0	0
180505	Fishing	1	0	0	0	0	0
180506	Fishing	1	0	0	0	0	0
180507	Fishing	1	0	0	0	0	0
180508	Fishing	1	0	0	0	0	0

180509 rows × 7 columns

We can see that 'Sporting Goods' category is encoded as '000001' and 'Fishing' is encoded as '100000'.

In [97]:
```
scm_updated.drop(columns=['Category Name'], inplace=True)
scm_updated.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180519 entries, 0 to 180518
Data columns (total 52 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   Days for shipping (real)        180519 non-null  int64  
 1   Days for shipment (scheduled)   180519 non-null  int64  
 2   Benefit per order               180519 non-null  float64
 3   Sales per customer              180519 non-null  float64
 4   Late_delivery_risk              180519 non-null  int64  
 5   Customer City                   180519 non-null  object 
 6   Customer Country                180519 non-null  object 
 7   Customer State                  180519 non-null  object 
 8   Customer Street                 180519 non-null  object 
 9   Customer Zipcode                180516 non-null  float64
 10  Department Name                 180519 non-null  object 
 11  Latitude                        180519 non-null  float64
 12  Longitude                       180519 non-null  float64
 13  Order City                      180519 non-null  object 
 14  Order Country                   180519 non-null  object 
 15  order date (DateOrders)         180519 non-null  object 
 16  Order Item Discount             180519 non-null  float64
 17  Order Item Discount Rate        180519 non-null  float64
 18  Order Item Product Price        180519 non-null  float64
 19  Order Item Profit Ratio         180519 non-null  float64
 20  Order Item Quantity             180519 non-null  int64  
 21  Sales                           180519 non-null  float64
 22  Order Item Total                180519 non-null  float64
 23  Order Profit Per Order          180519 non-null  float64
 24  Order Region                    180519 non-null  object 
 25  Order State                     180519 non-null  object 
 26  Order Zipcode                   24840 non-null   float64
 27  Product Name                    180519 non-null  object 
 28  Product Price                   180519 non-null  float64
 29  shipping date (DateOrders)      180519 non-null  object 
 30  Cash Type                       180519 non-null  float64
 31  Debit Type                      180519 non-null  float64
 32  Payment Type                    180519 non-null  float64
 33  Transfer Type                   180519 non-null  float64
 34  Consumer Customer Segment       180519 non-null  float64
 35  Corporate Customer Segment      180519 non-null  float64
 36  Home Office Customer Segment    180519 non-null  float64
 37  Africa Market                   180519 non-null  float64
 38  Europe Market                   180519 non-null  float64
 39  LATAM Market                    180519 non-null  float64
 40  Pacific Asia Market             180519 non-null  float64
 41  USCA Market                     180519 non-null  float64
 42  First Class Shipping Mode       180519 non-null  float64
 43  Same Day Shipping Mode          180519 non-null  float64
 44  Second Class Shipping Mode      180519 non-null  float64
 45  Standard Class Shipping Mode    180519 non-null  float64
 46  Category Name 0                 180519 non-null  int64  
 47  Category Name 1                 180519 non-null  int64  
 48  Category Name 2                 180519 non-null  int64  
 49  Category Name 3                 180519 non-null  int64  
 50  Category Name 4                 180519 non-null  int64  
 51  Category Name 5                 180519 non-null  int64  
dtypes: float64(30), int64(10), object(12)
memory usage: 71.6+ MB
```

In [98]:
```
# similarly, let us use binary encoding for the other two columns as well

binary_encoder_dept_name = ce.BinaryEncoder(cols=['Department Name'], ret
dept_name_binary_encoded_data = binary_encoder_dept_name.fit_transform(sc
# Department has 10 categories = 1010 in binary. Therefore we need 4 colu
scm_updated[['Department Name 0', 'Department Name 1', 'Department Name 2

binary_encoder_prod_name = ce.BinaryEncoder(cols=['Product Name'], return
prod_name_binary_encoded_data = binary_encoder_prod_name.fit_transform(sc
# Product Name has 118 categories = 1110110 in binary. Therefore we need
scm_updated[['Product Name 0', 'Product Name 1', 'Product Name 2', 'Produ
```

In [99]:
```
scm_updated.drop(columns=['Department Name', 'Product Name'], inplace=Tru
```

In [100…
```
scm_updated.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180519 entries, 0 to 180518
Data columns (total 61 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   Days for shipping (real)        180519 non-null  int64  
 1   Days for shipment (scheduled)   180519 non-null  int64  
 2   Benefit per order               180519 non-null  float64
 3   Sales per customer              180519 non-null  float64
 4   Late_delivery_risk              180519 non-null  int64  
 5   Customer City                   180519 non-null  object 
 6   Customer Country                180519 non-null  object 
 7   Customer State                  180519 non-null  object 
 8   Customer Street                 180519 non-null  object 
 9   Customer Zipcode                180516 non-null  float64
 10  Latitude                        180519 non-null  float64
 11  Longitude                       180519 non-null  float64
 12  Order City                      180519 non-null  object 
 13  Order Country                   180519 non-null  object 
 14  order date (DateOrders)         180519 non-null  object 
 15  Order Item Discount             180519 non-null  float64
 16  Order Item Discount Rate        180519 non-null  float64
 17  Order Item Product Price        180519 non-null  float64
 18  Order Item Profit Ratio         180519 non-null  float64
 19  Order Item Quantity             180519 non-null  int64  
 20  Sales                           180519 non-null  float64
 21  Order Item Total                180519 non-null  float64
 22  Order Profit Per Order          180519 non-null  float64
 23  Order Region                    180519 non-null  object 
 24  Order State                     180519 non-null  object 
 25  Order Zipcode                   24840 non-null   float64
 26  Product Price                   180519 non-null  float64
 27  shipping date (DateOrders)      180519 non-null  object 
 28  Cash Type                       180519 non-null  float64
 29  Debit Type                      180519 non-null  float64
 30  Payment Type                    180519 non-null  float64
 31  Transfer Type                   180519 non-null  float64
 32  Consumer Customer Segment       180519 non-null  float64
 33  Corporate Customer Segment      180519 non-null  float64
 34  Home Office Customer Segment    180519 non-null  float64
 35  Africa Market                   180519 non-null  float64
 36  Europe Market                   180519 non-null  float64
 37  LATAM Market                    180519 non-null  float64
 38  Pacific Asia Market             180519 non-null  float64
 39  USCA Market                     180519 non-null  float64
 40  First Class Shipping Mode       180519 non-null  float64
 41  Same Day Shipping Mode          180519 non-null  float64
 42  Second Class Shipping Mode      180519 non-null  float64
 43  Standard Class Shipping Mode    180519 non-null  float64
 44  Category Name 0                 180519 non-null  int64  
 45  Category Name 1                 180519 non-null  int64  
 46  Category Name 2                 180519 non-null  int64  
 47  Category Name 3                 180519 non-null  int64  
 48  Category Name 4                 180519 non-null  int64  
 49  Category Name 5                 180519 non-null  int64  
 50  Department Name 0               180519 non-null  int64  
 51  Department Name 1               180519 non-null  int64  
 52  Department Name 2               180519 non-null  int64  
 53  Department Name 3               180519 non-null  int64  
 54  Product Name 0                  180519 non-null  int64  
```

```
 55  Product Name 1                180519 non-null  int64
 56  Product Name 2                180519 non-null  int64
 57  Product Name 3                180519 non-null  int64
 58  Product Name 4                180519 non-null  int64
 59  Product Name 5                180519 non-null  int64
 60  Product Name 6                180519 non-null  int64
dtypes: float64(30), int64(21), object(10)
memory usage: 84.0+ MB
```

Handling Date Columns

We can see that there are some date/ time columns which pandas is currently reading as 'object' or string. They are:

- order date (DateOrders)
- shipping date (DateOrders)

Let us convert these strings to Date datatype, so that we can split each value into different columns.
For example, order date can be stored in the following way:

- Order Date Month
- Order Date Day
- Order Date Year
- Order Date Hour
- Order Date Min

In [101…
```
scm_updated['order date (DateOrders)'].head()
```

Out[101…
```
0    1/31/2018 22:56
1    1/13/2018 12:27
2    1/13/2018 12:06
3    1/13/2018 11:45
4    1/13/2018 11:24
Name: order date (DateOrders), dtype: object
```

In [102…
```
scm_updated['shipping date (DateOrders)'].head()
```

Out[102…
```
0     2/3/2018 22:56
1    1/18/2018 12:27
2    1/17/2018 12:06
3    1/16/2018 11:45
4    1/15/2018 11:24
Name: shipping date (DateOrders), dtype: object
```

From the data, we can see that order date and shipping date are both in the Month/Day/Year HH:MM format.

In [103…
```
# first let us convert this into a date datatype.
scm_updated['order date (DateOrders)'] = pd.to_datetime(scm_updated['orde
scm_updated['order date (DateOrders)'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 180519 entries, 0 to 180518
Series name: order date (DateOrders)
Non-Null Count   Dtype
--------------   -----
180519 non-null  datetime64[ns]
dtypes: datetime64[ns](1)
memory usage: 1.4 MB
```

In [104…
```
# similarly for shipping date
scm_updated['shipping date (DateOrders)'] = pd.to_datetime(scm_updated['s
```

In [110…
```
# now splitting the values into multiple columns:

scm_updated['Order Date Year'] = scm_updated['order date (DateOrders)'].d
scm_updated['Order Date Month'] = scm_updated['order date (DateOrders)'].
scm_updated['Order Date Day'] = scm_updated['order date (DateOrders)'].dt
scm_updated['Order Date Hour'] = scm_updated['order date (DateOrders)'].d
scm_updated['Order Date Min'] = scm_updated['order date (DateOrders)'].dt

scm_updated['Shipping Date Year'] = scm_updated['shipping date (DateOrder
scm_updated['Shipping Date Month'] = scm_updated['shipping date (DateOrde
scm_updated['Shipping Date Day'] = scm_updated['shipping date (DateOrders
scm_updated['Shipping Date Hour'] = scm_updated['shipping date (DateOrder
scm_updated['Shipping Date Min'] = scm_updated['shipping date (DateOrders
```

In [111…
```
# Let us look at the new columns now:
scm_updated[['order date (DateOrders)', 'Order Date Year', 'Order Date Mo
```

Out[111…

	order date (DateOrders)	Order Date Year	Order Date Month	Order Date Day	Order Date Hour	Order Date Min
0	2018-01-31 22:56:00	2018	1	31	22	56
1	2018-01-13 12:27:00	2018	1	13	12	27
2	2018-01-13 12:06:00	2018	1	13	12	6
3	2018-01-13 11:45:00	2018	1	13	11	45
4	2018-01-13 11:24:00	2018	1	13	11	24

In [112…
```
# Let us look at the new columns now:
scm_updated[['shipping date (DateOrders)', 'Shipping Date Year', 'Shippin
```

Out[112…

	shipping date (DateOrders)	Shipping Date Year	Shipping Date Month	Shipping Date Day	Shipping Date Hour	Shipping Date Min
0	2018-02-03 22:56:00	2018	2	3	22	56
1	2018-01-18 12:27:00	2018	1	18	12	27
2	2018-01-17 12:06:00	2018	1	17	12	6
3	2018-01-16 11:45:00	2018	1	16	11	45
4	2018-01-15 11:24:00	2018	1	15	11	24

In [113…
```
# now we can drop order date (DateOrders) & shipping date (DateOrders)
scm_updated.drop(columns=['order date (DateOrders)', 'shipping date (Date
```

In [114…
```
# Analysing the date columns
scm_updated['Order Date Year'].describe()
```

Out[114…
```
count    180519.000000
mean       2015.971150
std           0.829429
min        2015.000000
25%        2015.000000
50%        2016.000000
75%        2017.000000
max        2018.000000
Name: Order Date Year, dtype: float64
```

In [117…
```
scm_updated.groupby('Order Date Year').size()
```

Out[117…
```
Order Date Year
2015    62650
2016    62550
2017    53196
2018     2123
dtype: int64
```

From the above we can see that the orders were mainly between 2015 - 2017, and there are only a few year values in 2018.

In [118…
```
scm_updated.groupby('Order Date Month').size()
```

Out[118…
```
Order Date Month
1     17979
2     14529
3     15919
4     15435
5     15976
6     15139
7     15922
8     15912
9     15489
10    12955
11    12500
12    12764
dtype: int64
```

In [120…
```
scm_updated.groupby('Order Date Day').size()
```

Out[120…
```
Order Date Day
1     6050
2     6023
3     6060
4     5837
5     5853
6     5917
7     5990
8     5794
9     5936
10    5774
11    5973
12    5896
13    5923
14    5837
15    5815
16    5969
17    5889
18    5999
19    5954
20    5934
21    6004
22    5979
23    5841
24    5822
25    6018
26    5964
27    5868
28    5949
29    5584
30    5616
31    3451
dtype: int64
```

In [128…
```
scm_updated.groupby(['Order Date Year', 'Order Date Month']).size()
```

Out[128…

Order Date Year	Order Date Month	
2015	1	5322
	2	4729
	3	5362
	4	5126
	5	5357
	6	5134
	7	5299
	8	5273
	9	5140
	10	5302
	11	5235
	12	5371
2016	1	5317
	2	4894
	3	5210
	4	5097
	5	5302
	6	5054
	7	5305
	8	5334
	9	5160
	10	5398
	11	5210
	12	5269
2017	1	5217
	2	4906
	3	5347
	4	5212
	5	5317
	6	4951
	7	5318
	8	5305
	9	5189
	10	2255
	11	2055
	12	2124
2018	1	2123

dtype: int64

Handling Null Values

In [136…

```
# setting pandas to display all rows
pd.set_option('display.max_rows', None)

scm_updated.isna().sum()
```

Out[136…

Days for shipping (real)	0
Days for shipment (scheduled)	0
Benefit per order	0
Sales per customer	0
Late_delivery_risk	0
Customer City	0
Customer Country	0
Customer State	0
Customer Street	0
Customer Zipcode	3
Latitude	0
Longitude	0
Order City	0
Order Country	0
Order Item Discount	0
Order Item Discount Rate	0
Order Item Product Price	0
Order Item Profit Ratio	0
Order Item Quantity	0
Sales	0
Order Item Total	0
Order Profit Per Order	0
Order Region	0
Order State	0
Order Zipcode	155679
Product Price	0
Cash Type	0
Debit Type	0
Payment Type	0
Transfer Type	0
Consumer Customer Segment	0
Corporate Customer Segment	0
Home Office Customer Segment	0
Africa Market	0
Europe Market	0
LATAM Market	0
Pacific Asia Market	0
USCA Market	0
First Class Shipping Mode	0
Same Day Shipping Mode	0
Second Class Shipping Mode	0
Standard Class Shipping Mode	0
Category Name 0	0
Category Name 1	0
Category Name 2	0
Category Name 3	0
Category Name 4	0
Category Name 5	0
Department Name 0	0
Department Name 1	0
Department Name 2	0
Department Name 3	0
Product Name 0	0
Product Name 1	0
Product Name 2	0
Product Name 3	0
Product Name 4	0
Product Name 5	0
Product Name 6	0
Order Date Year	0

```
Order Date Month                0
Order Date Day                  0
Order Date Hour                 0
Order Date Min                  0
Shipping Date Year              0
Shipping Date Month             0
Shipping Date Day               0
Shipping Date Hour              0
Shipping Date Min               0
dtype: int64
```

We can see that the Order Zip Code has 155,679 null rows. Considering that our dataset has a total of 180,519 rows, it would be better to drop the order zip code column altogether.

Also the Customer Zip Code has 3 null values. Since 3 is extremely small compared to the number of rows in our dataset, we can remove those rows containing null rows for the customer zip-code.

In [146…
```
# dropping entire column for the order zipcode
scm_updated.drop(columns=['Order Zipcode'], inplace=True)
```

In [148…
```
# dropping the 3 null rows
scm_updated.dropna(inplace=True)
```

In [154…
```
# setting pandas to display 50 rows
pd.set_option('display.max_rows', 100)
scm_updated.isna().sum()
```

Out[154…

Days for shipping (real)	0
Days for shipment (scheduled)	0
Benefit per order	0
Sales per customer	0
Late_delivery_risk	0
Customer City	0
Customer Country	0
Customer State	0
Customer Street	0
Customer Zipcode	0
Latitude	0
Longitude	0
Order City	0
Order Country	0
Order Item Discount	0
Order Item Discount Rate	0
Order Item Product Price	0
Order Item Profit Ratio	0
Order Item Quantity	0
Sales	0
Order Item Total	0
Order Profit Per Order	0
Order Region	0
Order State	0
Product Price	0
Cash Type	0
Debit Type	0
Payment Type	0
Transfer Type	0
Consumer Customer Segment	0
Corporate Customer Segment	0
Home Office Customer Segment	0
Africa Market	0
Europe Market	0
LATAM Market	0
Pacific Asia Market	0
USCA Market	0
First Class Shipping Mode	0
Same Day Shipping Mode	0
Second Class Shipping Mode	0
Standard Class Shipping Mode	0
Category Name 0	0
Category Name 1	0
Category Name 2	0
Category Name 3	0
Category Name 4	0
Category Name 5	0
Department Name 0	0
Department Name 1	0
Department Name 2	0
Department Name 3	0
Product Name 0	0
Product Name 1	0
Product Name 2	0
Product Name 3	0
Product Name 4	0
Product Name 5	0
Product Name 6	0
Order Date Year	0
Order Date Month	0

```
Order Date Day                 0
Order Date Hour                0
Order Date Min                 0
Shipping Date Year             0
Shipping Date Month            0
Shipping Date Day              0
Shipping Date Hour             0
Shipping Date Min              0
dtype: int64
```

In [155…
```
len(scm_updated)
```

Out[155…
```
180516
```

Feature Engineering

Latitude & Longitude for Customer Locations

In the mid-term project, I visualized the stores on the world map. This was possible becaues the stores data had numerical values for the latitude and the longitude.

However, there are no latitude and longitude values for the customer locations. Therefore in this section I will try to convert the values from text to latitude & longitude.

In [145…
```
# setting pandas to display 10 rows
pd.set_option('display.max_rows', 10)

scm_updated[['Customer Street', 'Customer City', 'Customer State', 'Custo
```

Out[145...

	Customer Street	Customer City	Customer State	Customer Country	Customer Zipcode	Order City	
0	5365 Noble Nectar Island	Caguas	PR	Puerto Rico	725.0	Bekasi	Occi
1	2679 Rustic Loop	Caguas	PR	Puerto Rico	725.0	Bikaner	Ra
2	8510 Round Bear Gate	San Jose	CA	EE. UU.	95125.0	Bikaner	Ra
3	3200 Amber Bend	Los Angeles	CA	EE. UU.	90027.0	Townsville	Queer
4	8671 Iron Anchor Corners	Caguas	PR	Puerto Rico	725.0	Townsville	Queer
...	...	...	...	...	...	...	
180504	2137 Silver Brook Square	Caguas	PR	Puerto Rico	725.0	Canberra	Ter C Austr
180505	7127 Quiet Zephyr Nook	Highland	CA	EE. UU.	92346.0	Qingdao	Shar
180506	7106 Rustic Hickory Front	San Antonio	TX	EE. UU.	78228.0	Sanya	H
180507	5057 Fallen Knoll	Philadelphia	PA	EE. UU.	19134.0	Hanoi	Th?
180508	5237 Rustic Village	Waipahu	HI	EE. UU.	96797.0	Guangshui	

180509 rows × 10 columns

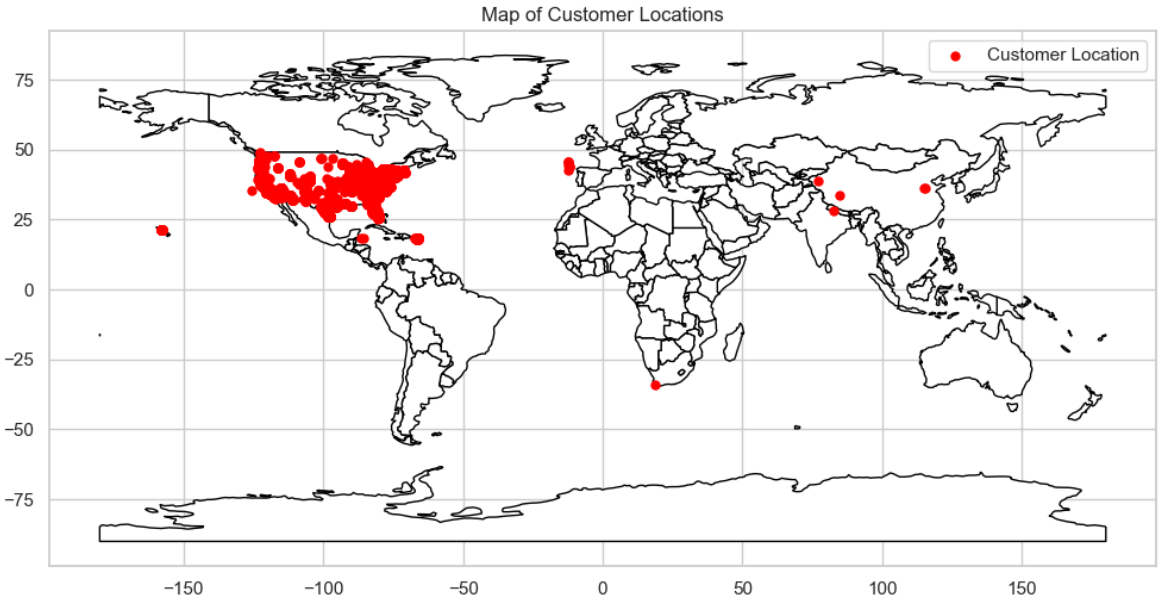
In the data definition csv file, it is mentioned that the latitude & longitude belong to the store, however I did some analysis and realized that the latitude & longitude data is of the customer location. For example on row 3, we can see that customer city is Los Angeles and order city is Townsville. I used an online Latitude & Longitude Finder and noticed that the latitude & longitude value as mentioned in the data frame correspond to the customer city.

Therefore, when we mapped the latitude & longitude on the map of the world initially, we were mapping the customer location, and not the store location.

In [156…
```
# using the latitude & longitude values to plot the customer location on

# dropping the duplicates rows
df_stores_location_lat_n_long = scm_df[['Latitude', 'Longitude']].drop_du

df_geo = gpd.GeoDataFrame(
    df_stores_location_lat_n_long,
    geometry=gpd.points_from_xy(df_stores_location_lat_n_long['Longitude'
)

#get the map image
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))

#plot the world map
ax = world.plot(figsize=(12, 8), color='white', edgecolor='black')

#plot the store locations as dots
df_geo.plot(ax=ax, marker='o', color='red', markersize=25, label='Custome

plt.title('Map of Customer Locations')
plt.legend(loc='upper right')
```

```
/var/folders/3m/tt_kbz7d2tx3f5cx3v4nd4940000gn/T/ipykernel_76908/250512671
2.py:12: FutureWarning:

The geopandas.dataset module is deprecated and will be removed in GeoPanda
s 1.0. You can get the original 'naturalearth_lowres' data from https://ww
w.naturalearthdata.com/downloads/110m-cultural-vectors/.
```

Out[156… `<matplotlib.legend.Legend at 0x29bb9d8b0>`



Similar to the above figure, it is important for us to understand the order store locations. One major issue for the late deliveries could be because the customer locations are far away from the stores.

Therefore, let us try to generate the latitude and longitude values of the Order Location.

In [175…
```
# in order to use geopy library, I had to create an account in geonames.o

from geopy import geocoders
geonames = geocoders.GeoNames(username='rkaushick')
geonames.geocode("Bikaner, Rajastán")
```

Out[175… Location(Bikaner, Rajasthan, India, (28.01762, 73.31495, 0.0))

In [176…
```
geonames.geocode("Bikaner, Rajastán")[1]
```

Out[176… (28.01762, 73.31495)

In [185…
```
print("Latitude: "),
print(geonames.geocode("Bikaner, Rajastán")[1][0])
print("Longitude: "),
print(geonames.geocode("Bikaner, Rajastán")[1][1])
```

```
Latitude:
28.01762
Longitude:
73.31495
```

From the above we can see that we are able to get the correct latitude and longitude values for the city and state input.

In [291…
```
# crearted this function to iterate through the data frame, and return a
from geopy import geocoders
import pandas as pd

def get_latitude_longitude_values(df):
    geoname = geocoders.GeoNames(username='rkaushick')
    new_df = pd.DataFrame(columns=['Order Latitude', 'Order Longitude'])
    for ind, row in df.iterrows():
        new_row = pd.DataFrame({'Order Latitude': geoname.geocode(str(row
                                    'Order Longitude': geoname.geocode(st
                                },
                                index=[ind]
                               )
        new_df = pd.concat([new_df, new_row], ignore_index=True)
        print("Done with "+str(ind))
        # if(ind == 9):
            #safety break
            # break
    return new_df
```

In [292…
```
new_df = get_latitude_longitude_values(scm_updated)
new_df.head()
```

```
Done with 0
Done with 1
Done with 2
Done with 3
Done with 4
Done with 5
Done with 6
Done with 7
Done with 8
Done with 9
Done with 10
Done with 11
Done with 12
Done with 13
Done with 14
Done with 15
Done with 16
Done with 17
Done with 18
Done with 19
Done with 20
Done with 21
Done with 22
Done with 23
Done with 24
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb Cell 161 line 1
----> <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y405sZmlsZQ%3D%3D?line=0'>1</a> new_df = get_latitude_longitude_values(scm_updated)
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y405sZmlsZQ%3D%3D?line=1'>2</a> new_df.head()

/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb Cell 161 line 9
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y405sZmlsZQ%3D%3D?line=6'>7</a> new_df = pd.DataFrame(columns=['Order Latitude', 'Order Longitude'])
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y405sZmlsZQ%3D%3D?line=7'>8</a> for ind, row in df.iterrows():
----> <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y405sZmlsZQ%3D%3D?line=8'>9</a>     new_row = pd.DataFrame({'Order Latitude': geoname.geocode(str(row['Order City'])+", "+str(row['Order State']))[1][0],
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y405sZmlsZQ%3D%3D?line=9'>10</a>                                'Order Longitude': geoname.geocode(str(row['Order City'])+", "+str(row['Order State']))[1][1]
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y405sZmlsZQ%3D%3D?line=10'>11</a>                               },
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y405sZmlsZQ%3D%3D?line=11'>12</a>                               index=[ind]
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y405sZmlsZQ%3D%3D?line=12'>13</a>                               )
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y405sZmlsZQ%3D%3D?line=13'>14</a>     new_df = pd.concat([new_df, new_row], ignore_index=True)
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y405sZmlsZQ%3D%3D?line=14'>15</a>     print("Done with "+str(ind))

TypeError: 'NoneType' object is not subscriptable
```

We received an error saying None Type object is not subscriptable. Therefore, in the function we added a print to debug while line in the data frame we are receiving this error. It has come from pandas line 25. Let's see the data in the 25th row of the dataframe:

In [295…
```
print(str(scm_updated['Order City'][25])+", "+str(scm_updated['Order Stat
```

```
Mandurah, Australia Occidental
```

In [298…
```
print(geonames.geocode("Mandurah, Australia Occidental"))
```

```
None
```

Here the issue is that there is no value returned for 'Mandurah, Australia Occidental'. Let us try to find the latitude and longitude of 'Mandurah, Australia'

In [302…
```
print(geonames.geocode("Mandurah, Australia")[1])
```

```
(-32.5269, 115.7217)
```

Now we are getting the values based on 'Australia' instead of 'Australia Occidental'. Therefore, in our pandas data-frame let us change all the occurances of State - 'Australia Occidental' to 'Australia'.

In [308…
```
scm_updated.loc[scm_updated['Order State'] =='Australia Occidental', 'Ord
```

Out[308…
```
25        Australia Occidental
26        Australia Occidental
245       Australia Occidental
396       Australia Occidental
397       Australia Occidental
                  ...
180161    Australia Occidental
180196    Australia Occidental
180273    Australia Occidental
180333    Australia Occidental
180377    Australia Occidental
Name: Order State, Length: 917, dtype: object
```

In [309…
```
scm_updated.loc[scm_updated['Order State'] =='Australia Occidental', 'Ord
```

In [310…
```
scm_updated.loc[scm_updated['Order State'] =='Australia Occidental', 'Ord
```

Out[310…
```
Series([], Name: Order State, dtype: object)
```

In [311…
```
# now let us try to run it again
new_df = get_latitude_longitude_values(scm_updated)
pd.set_option('display.max_rows', 10)
new_df.head(-10)
```

```
Done with 0
Done with 1
Done with 2
Done with 3
Done with 4
Done with 5
Done with 6
Done with 7
Done with 8
Done with 9
Done with 10
Done with 11
Done with 12
Done with 13
Done with 14
Done with 15
Done with 16
Done with 17
Done with 18
Done with 19
Done with 20
Done with 21
Done with 22
Done with 23
Done with 24
Done with 25
Done with 26
Done with 27
Done with 28
Done with 29
Done with 30
Done with 31
Done with 32
Done with 33
Done with 34
Done with 35
Done with 36
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb Cell 171 line 2
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y434sZmlsZQ%3D%3D?line=0'>1</a> # now let us try to run it again
----> <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y434sZmlsZQ%3D%3D?line=1'>2</a> new_df = get_latitude_longitude_values(scm_updated)
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y434sZmlsZQ%3D%3D?line=2'>3</a> pd.set_option('display.max_rows', 10)
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y434sZmlsZQ%3D%3D?line=3'>4</a> new_df.head(-10)

/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb Cell 171 line 9
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y434sZmlsZQ%3D%3D?line=6'>7</a> new_df = pd.DataFrame(columns=['Order Latitude', 'Order Longitude'])
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y434sZmlsZQ%3D%3D?line=7'>8</a> for ind, row in df.iterrows():
----> <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y434sZmlsZQ%3D%3D?line=8'>9</a>     new_row = pd.DataFrame({'Order Latitude': geoname.geocode(str(row['Order City'])+", "+str(row['Order State']))[1][0],
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y434sZmlsZQ%3D%3D?line=9'>10</a>                                'Order Longitude': geoname.geocode(str(row['Order City'])+", "+str(row['Order State']))[1][1]
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y434sZmlsZQ%3D%3D?line=10'>11</a>                               },
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y434sZmlsZQ%3D%3D?line=11'>12</a>                               index=[ind]
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y434sZmlsZQ%3D%3D?line=12'>13</a>                              )
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y434sZmlsZQ%3D%3D?line=13'>14</a>     new_df = pd.concat([new_df, new_row], ignore_index=True)
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y434sZmlsZQ%3D%3D?line=14'>15</a>     print("Done with "+str(ind))

TypeError: 'NoneType' object is not subscriptable
```

In [312…
```
print(str(scm_updated['Order City'][37])+", "+str(scm_updated['Order Stat
```

```
Wollongong, Nueva Gales del Sur
```

'Nueva Gales del Sur' is New South Wales' in another language. Similarly let us change it from 'Nueva Gales del Sur' to 'New South Wales'

In [313…
```
print(geonames.geocode("Wollongong, New South Wales")[1])
```

```
(-34.424, 150.89345)
```

In [314…
```
scm_updated.loc[scm_updated['Order State'] =='Nueva Gales del Sur', 'Orde
```

In [315…
```
# now let us try to run it again
new_df = get_latitude_longitude_values(scm_updated)
pd.set_option('display.max_rows', 10)
new_df.head(-10)
```

```
Done with 0
Done with 1
Done with 2
Done with 3
Done with 4
Done with 5
Done with 6
Done with 7
Done with 8
Done with 9
Done with 10
Done with 11
Done with 12
Done with 13
Done with 14
Done with 15
Done with 16
Done with 17
Done with 18
Done with 19
Done with 20
Done with 21
Done with 22
Done with 23
Done with 24
Done with 25
Done with 26
Done with 27
Done with 28
Done with 29
Done with 30
Done with 31
Done with 32
Done with 33
Done with 34
Done with 35
Done with 36
Done with 37
Done with 38
Done with 39
Done with 40
Done with 41
Done with 42
Done with 43
Done with 44
Done with 45
Done with 46
Done with 47
Done with 48
Done with 49
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb Cell 176 line 2
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y445sZmlsZQ%3D%3D?line=0'>1</a> # now let us try to run it again
----> <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y445sZmlsZQ%3D%3D?line=1'>2</a> new_df = get_latitude_longitude_values(scm_updated)
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y445sZmlsZQ%3D%3D?line=2'>3</a> pd.set_option('display.max_rows', 10)
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y445sZmlsZQ%3D%3D?line=3'>4</a> new_df.head(-10)

/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb Cell 176 line 9
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y445sZmlsZQ%3D%3D?line=6'>7</a> new_df = pd.DataFrame(columns=['Order Latitude', 'Order Longitude'])
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y445sZmlsZQ%3D%3D?line=7'>8</a> for ind, row in df.iterrows():
----> <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y445sZmlsZQ%3D%3D?line=8'>9</a>     new_row = pd.DataFrame({'Order Latitude': geoname.geocode(str(row['Order City'])+", "+str(row['Order State']))[1][0],
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y445sZmlsZQ%3D%3D?line=9'>10</a>                                 'Order Longitude': geoname.geocode(str(row['Order City'])+", "+str(row['Order State']))[1][1]
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y445sZmlsZQ%3D%3D?line=10'>11</a>                                 },
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y445sZmlsZQ%3D%3D?line=11'>12</a>                                 index=[ind]
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y445sZmlsZQ%3D%3D?line=12'>13</a>                                 )
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y445sZmlsZQ%3D%3D?line=13'>14</a>     new_df = pd.concat([new_df, new_row], ignore_index=True)
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y445sZmlsZQ%3D%3D?line=14'>15</a>     print("Done with "+str(ind))

TypeError: 'NoneType' object is not subscriptable
```

In [316…
```
print(str(scm_updated['Order City'][50])+", "+str(scm_updated['Order Stat
```

```
Murray Bridge, Australia del Sur
```

In [317…
```
print(geonames.geocode("Murray Bridge, Australia")[1])
```

```
(-35.11986, 139.27345)
```

In [318…
```
scm_updated.loc[scm_updated['Order State'] =='Australia del Sur', 'Order
```

In [319…
```
# now let us try to run it again
new_df = get_latitude_longitude_values(scm_updated)
pd.set_option('display.max_rows', 10)
new_df.head(-10)
```

```
Done with 0
Done with 1
Done with 2
Done with 3
Done with 4
Done with 5
Done with 6
Done with 7
Done with 8
Done with 9
Done with 10
Done with 11
Done with 12
Done with 13
Done with 14
Done with 15
Done with 16
Done with 17
Done with 18
Done with 19
Done with 20
Done with 21
Done with 22
Done with 23
Done with 24
Done with 25
Done with 26
Done with 27
Done with 28
Done with 29
Done with 30
Done with 31
Done with 32
Done with 33
Done with 34
Done with 35
Done with 36
Done with 37
Done with 38
Done with 39
Done with 40
Done with 41
Done with 42
Done with 43
Done with 44
Done with 45
Done with 46
Done with 47
Done with 48
Done with 49
Done with 50
Done with 51
Done with 52
Done with 53
Done with 54
Done with 55
Done with 56
Done with 57
Done with 58
Done with 59
```

```
Done with 60
Done with 61
Done with 62
Done with 63
Done with 64
Done with 65
Done with 66
Done with 67
Done with 68
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb Cell 180 line 2
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y452sZmlsZQ%3D%3D?line=0'>1</a> # now let us try to run it again
----> <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y452sZmlsZQ%3D%3D?line=1'>2</a> new_df = get_latitude_longitude_values(scm_updated)
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y452sZmlsZQ%3D%3D?line=2'>3</a> pd.set_option('display.max_rows', 10)
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y452sZmlsZQ%3D%3D?line=3'>4</a> new_df.head(-10)

/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb Cell 180 line 9
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y452sZmlsZQ%3D%3D?line=6'>7</a> new_df = pd.DataFrame(columns=['Order Latitude', 'Order Longitude'])
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y452sZmlsZQ%3D%3D?line=7'>8</a> for ind, row in df.iterrows():
----> <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y452sZmlsZQ%3D%3D?line=8'>9</a>     new_row = pd.DataFrame({'Order Latitude': geoname.geocode(str(row['Order City'])+", "+str(row['Order State']))[1][0],
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y452sZmlsZQ%3D%3D?line=9'>10</a>                                'Order Longitude': geoname.geocode(str(row['Order City'])+", "+str(row['Order State']))[1][1]
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y452sZmlsZQ%3D%3D?line=10'>11</a>                               },
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y452sZmlsZQ%3D%3D?line=11'>12</a>                               index=[ind]
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y452sZmlsZQ%3D%3D?line=12'>13</a>                              )
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y452sZmlsZQ%3D%3D?line=13'>14</a>     new_df = pd.concat([new_df, new_row], ignore_index=True)
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y452sZmlsZQ%3D%3D?line=14'>15</a>     print("Done with "+str(ind))

TypeError: 'NoneType' object is not subscriptable
```

In [320…
```
print(str(scm_updated['Order City'][69])+", "+str(scm_updated['Order Stat
```

```
Hayange, Alsacia-Champaña-Ardenas-Lorena
```

The above is a place in France. But since we can see that it is able to recognize the latitude and longitude without the state. Therefore, let us try to find the latitude and longitude of only the cities.

In [321…
```
print(geonames.geocode("Hayange")[1])
```

```
(49.32881, 6.06278)
```

In [325…
```
# re-writing the function
def get_latitude_longitude_values(df):
    geoname = geocoders.GeoNames(username='rkaushick')
    new_df = pd.DataFrame(columns=['Order Latitude', 'Order Longitude'])
    for ind, row in df.iterrows():
        new_row = pd.DataFrame({'Order Latitude': geoname.geocode(str(row
                                    'Order Longitude': geoname.geocode(st
                                },
                                index=[ind]
                               )
        new_df = pd.concat([new_df, new_row], ignore_index=True)
        print(str(ind))
        # if(ind == 9):
            #safety break
            # break
    return new_df
```

In [326…
```
# now let us try to run it again
new_df = get_latitude_longitude_values(scm_updated)
# new_df.head(-10)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
```

```
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
```

```
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
```

```
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
```

```
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
```

600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659

660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719

```
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
```

960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019

```
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
```

1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139

1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199

1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259

```
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
```

```
---------------------------------------------------------------------------
GeocoderQuotaExceeded                     Traceback (most recent call last)
/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb Cell 185 line 2
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y455sZmlsZQ%3D%3D?line=0'>1</a> # now let us try to run it again
----> <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y455sZmlsZQ%3D%3D?line=1'>2</a> new_df = get_latitude_longitude_values(scm_updated)
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y455sZmlsZQ%3D%3D?line=2'>3</a> # new_df.head(-10)

/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb Cell 185 line 6
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y455sZmlsZQ%3D%3D?line=3'>4</a> new_df = pd.DataFrame(columns=['Order Latitude', 'Order Longitude'])
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y455sZmlsZQ%3D%3D?line=4'>5</a> for ind, row in df.iterrows():
----> <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y455sZmlsZQ%3D%3D?line=5'>6</a>     new_row = pd.DataFrame({'Order Latitude': geoname.geocode(str(row['Order City']))[1][0],
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y455sZmlsZQ%3D%3D?line=6'>7</a>                                 'Order Longitude': geoname.geocode(str(row['Order City']))[1][1]
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y455sZmlsZQ%3D%3D?line=7'>8</a>                                 },
      <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y455sZmlsZQ%3D%3D?line=8'>9</a>                                 index=[ind]
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y455sZmlsZQ%3D%3D?line=9'>10</a>                                )
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y455sZmlsZQ%3D%3D?line=10'>11</a>     new_df = pd.concat([new_df, new_row], ignore_index=True)
     <a href='vscode-notebook-cell:/Users/rishabhkaushick/Documents/Northeastern/INFO6105/FinalProject/RishLK_FinalProject.ipynb#Y455sZmlsZQ%3D%3D?line=11'>12</a>     print(str(ind))

File /opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/geopy/geocoders/geonames.py:161, in GeoNames.geocode(self, query, exactly_one, timeout, country, country_bias)
    159 logger.debug("%s.geocode: %s", self.__class__.__name__, url)
    160 callback = partial(self._parse_json, exactly_one=exactly_one)
--> 161 return self._call_geocoder(url, callback, timeout=timeout)

File /opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/geopy/geocoders/base.py:386, in Geocoder._call_geocoder(self, url, callback, timeout, is_json, headers)
```

```
    384         return fut()
    385     else:
--> 386         return callback(result)
    387 except Exception as error:
    388     res = self._adapter_error_handler(error)

File /opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/geo
py/geocoders/geonames.py:334, in GeoNames._parse_json(self, doc, exactly_o
ne)
    330 """
    331 Parse JSON response body.
    332 """
    333 places = doc.get('geonames', [])
--> 334 self._raise_for_error(doc)
    335 if not len(places):
    336     return None

File /opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/geo
py/geocoders/geonames.py:312, in GeoNames._raise_for_error(self, body)
    310     raise GeocoderAuthenticationFailure(message)
    311 if code in (18, 19, 20):
--> 312     raise GeocoderQuotaExceeded(message)
    313 raise GeocoderServiceError(message)

GeocoderQuotaExceeded: the hourly limit of 1000 credits for rkaushick has
been exceeded. Please throttle your requests or use the commercial servic
e.
```

Based on the above error it looks like we do not have enough credits in the free account to generaete the latitude and longitude values of all the cities. After calculating more than 1,280 latitude and longitude values, I encountered the problem.

Hence, I will abandon this feature extraction of the Order store latitude and longitude values as out of scope for the this final project.

In [328…

```
# Therefore let us use Binary Encoders for encoding the Order City & Coun
scm_updated[['Order City', 'Order Country', 'Customer City', 'Customer Co
```

Out [328…

	Order City	Order Country	Customer City	Customer Country
0	Bekasi	Indonesia	Caguas	Puerto Rico
1	Bikaner	India	Caguas	Puerto Rico
2	Bikaner	India	San Jose	EE. UU.
3	Townsville	Australia	Los Angeles	EE. UU.
4	Townsville	Australia	Caguas	Puerto Rico
...	...	...	...	...
180504	Canberra	Australia	Caguas	Puerto Rico
180505	Qingdao	China	Highland	EE. UU.
180506	Sanya	China	San Antonio	EE. UU.
180507	Hanoi	Vietnam	Philadelphia	EE. UU.
180508	Guangshui	China	Waipahu	EE. UU.

180506 rows × 4 columns

In [334…

```
# binary encoding of order city & customer city
binary_encoder_city = ce.BinaryEncoder(cols=['Order City', 'Customer City
city_binary_encoded_data = binary_encoder_city.fit_transform(scm_updated[
city_binary_encoded_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 180516 entries, 0 to 180518
Data columns (total 22 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   Order City_0     180516 non-null  int64
 1   Order City_1     180516 non-null  int64
 2   Order City_2     180516 non-null  int64
 3   Order City_3     180516 non-null  int64
 4   Order City_4     180516 non-null  int64
 5   Order City_5     180516 non-null  int64
 6   Order City_6     180516 non-null  int64
 7   Order City_7     180516 non-null  int64
 8   Order City_8     180516 non-null  int64
 9   Order City_9     180516 non-null  int64
 10  Order City_10    180516 non-null  int64
 11  Order City_11    180516 non-null  int64
 12  Customer City_0  180516 non-null  int64
 13  Customer City_1  180516 non-null  int64
 14  Customer City_2  180516 non-null  int64
 15  Customer City_3  180516 non-null  int64
 16  Customer City_4  180516 non-null  int64
 17  Customer City_5  180516 non-null  int64
 18  Customer City_6  180516 non-null  int64
 19  Customer City_7  180516 non-null  int64
 20  Customer City_8  180516 non-null  int64
 21  Customer City_9  180516 non-null  int64
dtypes: int64(22)
memory usage: 31.7 MB
```

In [335…
```
# binary encoding of order country
binary_encoder_country = ce.BinaryEncoder(cols=['Order Country', 'Custome
country_binary_encoded_data = binary_encoder_country.fit_transform(scm_up
country_binary_encoded_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 180516 entries, 0 to 180518
Data columns (total 10 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   Order Country_0     180516 non-null  int64
 1   Order Country_1     180516 non-null  int64
 2   Order Country_2     180516 non-null  int64
 3   Order Country_3     180516 non-null  int64
 4   Order Country_4     180516 non-null  int64
 5   Order Country_5     180516 non-null  int64
 6   Order Country_6     180516 non-null  int64
 7   Order Country_7     180516 non-null  int64
 8   Customer Country_0  180516 non-null  int64
 9   Customer Country_1  180516 non-null  int64
dtypes: int64(10)
memory usage: 15.1 MB
```

In [336…
```
# interesting how customer country only has 2 values
scm_updated['Customer Country'].value_counts()
```

Out[336…
```
Customer Country
EE. UU.        111143
Puerto Rico     69373
Name: count, dtype: int64
```

In [337…
```
# Now setting these values in the data frame
scm_updated[['Order City 0', 'Order City 1', 'Order City 2', 'Order City
             'Order City 4', 'Order City 5', 'Order City 6', 'Order City
             'Order City 8', 'Order City 9', 'Order City 11', 'Order City
             'Customer City 0', 'Customer City 1', 'Customer City 2', 'Cu
             'Customer City 5', 'Customer City 6', 'Customer City 7', 'Cu

scm_updated[['Order Country 0', 'Order Country 1',
             'Order Country 2', 'Order Country 3',
             'Order Country 4', 'Order Country 5',
             'Order Country 6', 'Order Country 7',
             'Customer Country 0', 'Customer Country 1']]  = country_bina
```

In [339…
```
scm_updated[['Order City',
             'Order City 0', 'Order City 1', 'Order City 2', 'Order City
             'Order City 4', 'Order City 5', 'Order City 6', 'Order City
             'Order City 8', 'Order City 9', 'Order City 11', 'Order City
             ]].head()
```

Out[339…

	Order City	Order City 0	Order City 1	Order City 2	Order City 3	Order City 4	Order City 5	Order City 6	Order City 7	Order City 8	Orc C
0	Bekasi	0	0	0	0	0	0	0	0	0	
1	Bikaner	0	0	0	0	0	0	0	0	0	
2	Bikaner	0	0	0	0	0	0	0	0	0	
3	Townsville	0	0	0	0	0	0	0	0	0	
4	Townsville	0	0	0	0	0	0	0	0	0	

Now that we have done the binary encoding of the order city, order country, customer city & customer country, we can drop all the address related columns present in the dataframe.

In [340…

```
scm_updated.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 180516 entries, 0 to 180518
Data columns (total 100 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   Days for shipping (real)        180516 non-null  int64  
 1   Days for shipment (scheduled)   180516 non-null  int64  
 2   Benefit per order               180516 non-null  float64
 3   Sales per customer              180516 non-null  float64
 4   Late_delivery_risk              180516 non-null  int64  
 5   Customer City                   180516 non-null  object 
 6   Customer Country                180516 non-null  object 
 7   Customer State                  180516 non-null  object 
 8   Customer Street                 180516 non-null  object 
 9   Customer Zipcode                180516 non-null  float64
 10  Latitude                        180516 non-null  float64
 11  Longitude                       180516 non-null  float64
 12  Order City                      180516 non-null  object 
 13  Order Country                   180516 non-null  object 
 14  Order Item Discount             180516 non-null  float64
 15  Order Item Discount Rate        180516 non-null  float64
 16  Order Item Product Price        180516 non-null  float64
 17  Order Item Profit Ratio         180516 non-null  float64
 18  Order Item Quantity             180516 non-null  int64  
 19  Sales                           180516 non-null  float64
 20  Order Item Total                180516 non-null  float64
 21  Order Profit Per Order          180516 non-null  float64
 22  Order Region                    180516 non-null  object 
 23  Order State                     180516 non-null  object 
 24  Product Price                   180516 non-null  float64
 25  Cash Type                       180516 non-null  float64
 26  Debit Type                      180516 non-null  float64
 27  Payment Type                    180516 non-null  float64
 28  Transfer Type                   180516 non-null  float64
 29  Consumer Customer Segment       180516 non-null  float64
 30  Corporate Customer Segment      180516 non-null  float64
 31  Home Office Customer Segment    180516 non-null  float64
 32  Africa Market                   180516 non-null  float64
 33  Europe Market                   180516 non-null  float64
 34  LATAM Market                    180516 non-null  float64
 35  Pacific Asia Market             180516 non-null  float64
 36  USCA Market                     180516 non-null  float64
 37  First Class Shipping Mode       180516 non-null  float64
 38  Same Day Shipping Mode          180516 non-null  float64
 39  Second Class Shipping Mode      180516 non-null  float64
 40  Standard Class Shipping Mode    180516 non-null  float64
 41  Category Name 0                 180516 non-null  int64  
 42  Category Name 1                 180516 non-null  int64  
 43  Category Name 2                 180516 non-null  int64  
 44  Category Name 3                 180516 non-null  int64  
 45  Category Name 4                 180516 non-null  int64  
 46  Category Name 5                 180516 non-null  int64  
 47  Department Name 0               180516 non-null  int64  
 48  Department Name 1               180516 non-null  int64  
 49  Department Name 2               180516 non-null  int64  
 50  Department Name 3               180516 non-null  int64  
 51  Product Name 0                  180516 non-null  int64  
 52  Product Name 1                  180516 non-null  int64  
 53  Product Name 2                  180516 non-null  int64  
 54  Product Name 3                  180516 non-null  int64  
```

```
 55  Product Name 4                180516 non-null  int64
 56  Product Name 5                180516 non-null  int64
 57  Product Name 6                180516 non-null  int64
 58  Order Date Year               180516 non-null  int32
 59  Order Date Month              180516 non-null  int32
 60  Order Date Day                180516 non-null  int32
 61  Order Date Hour               180516 non-null  int32
 62  Order Date Min                180516 non-null  int32
 63  Shipping Date Year            180516 non-null  int32
 64  Shipping Date Month           180516 non-null  int32
 65  Shipping Date Day             180516 non-null  int32
 66  Shipping Date Hour            180516 non-null  int32
 67  Shipping Date Min             180516 non-null  int32
 68  Order City 0                  180516 non-null  int64
 69  Order City 1                  180516 non-null  int64
 70  Order City 2                  180516 non-null  int64
 71  Order City 3                  180516 non-null  int64
 72  Order City 4                  180516 non-null  int64
 73  Order City 5                  180516 non-null  int64
 74  Order City 6                  180516 non-null  int64
 75  Order City 7                  180516 non-null  int64
 76  Order City 8                  180516 non-null  int64
 77  Order City 9                  180516 non-null  int64
 78  Order City 11                 180516 non-null  int64
 79  Order City 12                 180516 non-null  int64
 80  Customer City 0               180516 non-null  int64
 81  Customer City 1               180516 non-null  int64
 82  Customer City 2               180516 non-null  int64
 83  Customer City 3               180516 non-null  int64
 84  Customer City 4               180516 non-null  int64
 85  Customer City 5               180516 non-null  int64
 86  Customer City 6               180516 non-null  int64
 87  Customer City 7               180516 non-null  int64
 88  Customer City 8               180516 non-null  int64
 89  Customer City 9               180516 non-null  int64
 90  Order Country 0               180516 non-null  int64
 91  Order Country 1               180516 non-null  int64
 92  Order Country 2               180516 non-null  int64
 93  Order Country 3               180516 non-null  int64
 94  Order Country 4               180516 non-null  int64
 95  Order Country 5               180516 non-null  int64
 96  Order Country 6               180516 non-null  int64
 97  Order Country 7               180516 non-null  int64
 98  Customer Country 0            180516 non-null  int64
 99  Customer Country 1            180516 non-null  int64
dtypes: float64(29), int32(10), int64(53), object(8)
memory usage: 136.2+ MB
```

Address Columns to drop:

- Customer City
- Customer Country
- Customer State
- Customer Street
- Customer Zipcode
- Order City
- Order Country

- Order Region
- Order State

In [341…
```
scm_updated.drop(columns=['Customer City', 'Customer Country', 'Customer
                          'Order City', 'Order Country', 'Order Region',
```

In [342…
```
scm_updated.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 180516 entries, 0 to 180518
Data columns (total 91 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   Days for shipping (real)        180516 non-null  int64  
 1   Days for shipment (scheduled)   180516 non-null  int64  
 2   Benefit per order               180516 non-null  float64
 3   Sales per customer              180516 non-null  float64
 4   Late_delivery_risk              180516 non-null  int64  
 5   Latitude                        180516 non-null  float64
 6   Longitude                       180516 non-null  float64
 7   Order Item Discount             180516 non-null  float64
 8   Order Item Discount Rate        180516 non-null  float64
 9   Order Item Product Price        180516 non-null  float64
 10  Order Item Profit Ratio         180516 non-null  float64
 11  Order Item Quantity             180516 non-null  int64  
 12  Sales                           180516 non-null  float64
 13  Order Item Total                180516 non-null  float64
 14  Order Profit Per Order          180516 non-null  float64
 15  Product Price                   180516 non-null  float64
 16  Cash Type                       180516 non-null  float64
 17  Debit Type                      180516 non-null  float64
 18  Payment Type                    180516 non-null  float64
 19  Transfer Type                   180516 non-null  float64
 20  Consumer Customer Segment       180516 non-null  float64
 21  Corporate Customer Segment      180516 non-null  float64
 22  Home Office Customer Segment    180516 non-null  float64
 23  Africa Market                   180516 non-null  float64
 24  Europe Market                   180516 non-null  float64
 25  LATAM Market                    180516 non-null  float64
 26  Pacific Asia Market             180516 non-null  float64
 27  USCA Market                     180516 non-null  float64
 28  First Class Shipping Mode       180516 non-null  float64
 29  Same Day Shipping Mode          180516 non-null  float64
 30  Second Class Shipping Mode      180516 non-null  float64
 31  Standard Class Shipping Mode    180516 non-null  float64
 32  Category Name 0                 180516 non-null  int64  
 33  Category Name 1                 180516 non-null  int64  
 34  Category Name 2                 180516 non-null  int64  
 35  Category Name 3                 180516 non-null  int64  
 36  Category Name 4                 180516 non-null  int64  
 37  Category Name 5                 180516 non-null  int64  
 38  Department Name 0               180516 non-null  int64  
 39  Department Name 1               180516 non-null  int64  
 40  Department Name 2               180516 non-null  int64  
 41  Department Name 3               180516 non-null  int64  
 42  Product Name 0                  180516 non-null  int64  
 43  Product Name 1                  180516 non-null  int64  
 44  Product Name 2                  180516 non-null  int64  
 45  Product Name 3                  180516 non-null  int64  
 46  Product Name 4                  180516 non-null  int64  
 47  Product Name 5                  180516 non-null  int64  
 48  Product Name 6                  180516 non-null  int64  
 49  Order Date Year                 180516 non-null  int32  
 50  Order Date Month                180516 non-null  int32  
 51  Order Date Day                  180516 non-null  int32  
 52  Order Date Hour                 180516 non-null  int32  
 53  Order Date Min                  180516 non-null  int32  
 54  Shipping Date Year              180516 non-null  int32  
```

```
 55  Shipping Date Month            180516 non-null  int32
 56  Shipping Date Day              180516 non-null  int32
 57  Shipping Date Hour             180516 non-null  int32
 58  Shipping Date Min              180516 non-null  int32
 59  Order City 0                   180516 non-null  int64
 60  Order City 1                   180516 non-null  int64
 61  Order City 2                   180516 non-null  int64
 62  Order City 3                   180516 non-null  int64
 63  Order City 4                   180516 non-null  int64
 64  Order City 5                   180516 non-null  int64
 65  Order City 6                   180516 non-null  int64
 66  Order City 7                   180516 non-null  int64
 67  Order City 8                   180516 non-null  int64
 68  Order City 9                   180516 non-null  int64
 69  Order City 11                  180516 non-null  int64
 70  Order City 12                  180516 non-null  int64
 71  Customer City 0                180516 non-null  int64
 72  Customer City 1                180516 non-null  int64
 73  Customer City 2                180516 non-null  int64
 74  Customer City 3                180516 non-null  int64
 75  Customer City 4                180516 non-null  int64
 76  Customer City 5                180516 non-null  int64
 77  Customer City 6                180516 non-null  int64
 78  Customer City 7                180516 non-null  int64
 79  Customer City 8                180516 non-null  int64
 80  Customer City 9                180516 non-null  int64
 81  Order Country 0                180516 non-null  int64
 82  Order Country 1                180516 non-null  int64
 83  Order Country 2                180516 non-null  int64
 84  Order Country 3                180516 non-null  int64
 85  Order Country 4                180516 non-null  int64
 86  Order Country 5                180516 non-null  int64
 87  Order Country 6                180516 non-null  int64
 88  Order Country 7                180516 non-null  int64
 89  Customer Country 0             180516 non-null  int64
 90  Customer Country 1             180516 non-null  int64
dtypes: float64(28), int32(10), int64(53)
memory usage: 123.9 MB
```

With this we have preprocessed all the columns into the required format for the machine learning model. Now we do not have any column with null values and we have also successfully converted all the text columns into relevant numeric values. So we can now train the models and see the results

Split Train & Test Data

In [343…

```
from sklearn.model_selection import train_test_split

X_train_v2, X_test_v2, y_train_v2, y_test_v2 = train_test_split(scm_updat
# by default setting the train size to be 25 %
```

Decision Tree Model #3

In [344…

```
tree_clf_3 = DecisionTreeClassifier(max_depth=5, max_features="sqrt", ran
tree_clf_3.fit(X_train_v2, y_train_v2)
```

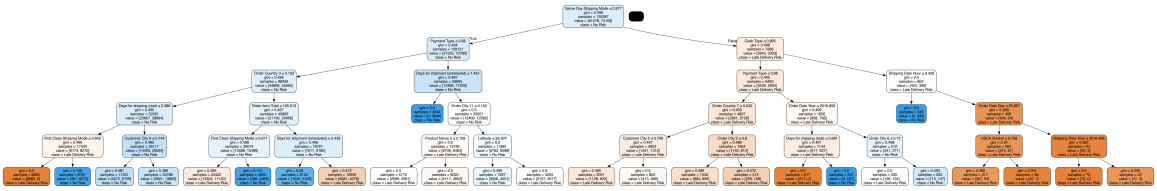
Out[344…

```
DecisionTreeClassifier(max_depth=5, max_features='sqrt', random_state=23,
                       splitter='random')
```

In [353…

```
# plotting the decision tree

dot_data = StringIO()
export_graphviz(
        tree_clf_3,
        out_file=dot_data,
        feature_names= X_train_v2.columns.values.tolist(),
        class_names= ["Late Delivery Risk", "No Risk"],
        rounded=True,
        filled=True,
        special_characters=True
    )
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('decision_tree_model_3.png')
Image(graph.create_png())
```

Out[353…



In [346…

```
# Let us see how well the training does:
tree_clf_3.score(X_train_v2, y_train_v2)
```

Out[346… 0.6596128136379416

In [347…

```
tree_clf_3.score(X_test_v2, y_test_v2)
```

Out[347… 0.6607724523033969

Random Forest Classifier #1

In [348…

```
from sklearn.ensemble import RandomForestClassifier

rf_clf = RandomForestClassifier(random_state=23,
                                n_jobs=-1,
                                max_depth=5,
                                n_estimators=100,
                                oob_score=True)
```

In [350…

```
%%time
rf_clf.fit(X_train_v2, y_train_v2)
```

```
CPU times: user 25min 27s, sys: 8.24 s, total: 25min 36s
Wall time: 3min 49s
```

Out[350…

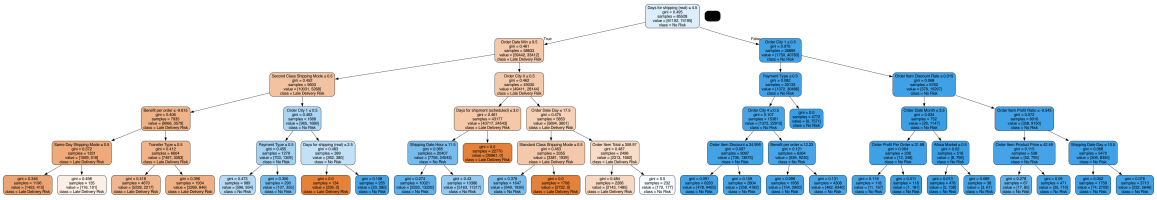
RandomForestClassifier

```
RandomForestClassifier(max_depth=5, n_jobs=-1, oob_score=True, ra
ndom_state=23)
```

In [402…

```
# plotting the random forest's 1st tree

dot_data = StringIO()
export_graphviz(
        rf_clf.estimators_[0],
        out_file=dot_data,
        feature_names= X_train_v2.columns.values.tolist(),
        class_names= ["Late Delivery Risk", "No Risk"],
        rounded=True,
        filled=True,
        special_characters=True
    )
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('random_forest_model_1.png')
Image(graph.create_png())
```

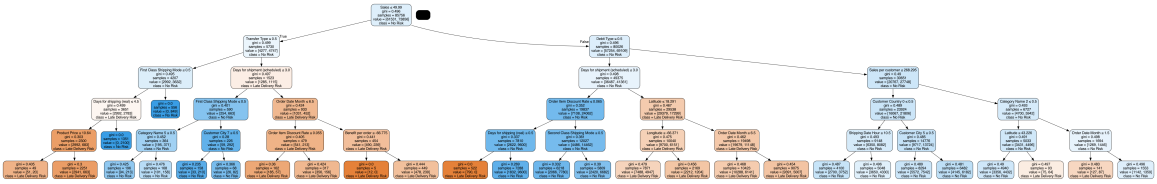
Out[402…



In [403…

```
# plotting the random forest's 2nd tree

dot_data = StringIO()
export_graphviz(
        rf_clf.estimators_[1],
        out_file=dot_data,
        feature_names= X_train_v2.columns.values.tolist(),
        class_names= ["Late Delivery Risk", "No Risk"],
        rounded=True,
        filled=True,
        special_characters=True
    )
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('random_forest_model_1_2nd_estimater.png')
Image(graph.create_png())
```

Out[403…



In [351…

```
# checking the training accuracy
rf_clf.score(X_train_v2, y_train_v2)
```

Out[351…

```
0.9359982863938192
```

For the random forest model, we can see that the training acuracy is 93.60%. Let us look at the accuracy of the test set.

In [352…
```
# checking the test accuracy
rf_clf.score(X_test_v2, y_test_v2)
```

Out[352…
```
0.9358284030224467
```

The test accuracy is very close at 93.58%.

Feature Importance

In [364…
```
pd.set_option('display.max_rows', None)
important_features_rf_clf = pd.DataFrame({
    "Column": X_train_v2.columns,
    "Importance Score": rf_clf.feature_importances_
})
important_features_rf_clf.sort_values(by='Importance Score', ascending=Fa
```

Out[364...

	Column	Importance Score
0	Days for shipping (real)	0.346944
1	Days for shipment (scheduled)	0.216117
30	Standard Class Shipping Mode	0.202429
27	First Class Shipping Mode	0.120478
29	Second Class Shipping Mode	0.045490
56	Shipping Date Hour	0.017773
28	Same Day Shipping Mode	0.016205
51	Order Date Hour	0.015866
18	Transfer Type	0.007911
55	Shipping Date Day	0.001983
50	Order Date Day	0.001364
16	Debit Type	0.001320
17	Payment Type	0.001060
4	Latitude	0.000513
5	Longitude	0.000413
57	Shipping Date Min	0.000310
52	Order Date Min	0.000302
13	Order Profit Per Order	0.000210
12	Order Item Total	0.000209
2	Benefit per order	0.000201
54	Shipping Date Month	0.000200
49	Order Date Month	0.000195
6	Order Item Discount	0.000153
3	Sales per customer	0.000139
9	Order Item Profit Ratio	0.000136
15	Cash Type	0.000105
63	Order City 5	0.000101
7	Order Item Discount Rate	0.000092
58	Order City 0	0.000085
8	Order Item Product Price	0.000082
11	Sales	0.000078
14	Product Price	0.000075
24	LATAM Market	0.000069
72	Customer City 2	0.000063

	Column	Importance Score
68	Order City 11	0.000062
71	Customer City 1	0.000061
80	Order Country 0	0.000056
75	Customer City 5	0.000047
88	Customer Country 0	0.000046
53	Shipping Date Year	0.000045
86	Order Country 6	0.000044
20	Corporate Customer Segment	0.000044
74	Customer City 4	0.000042
85	Order Country 5	0.000042
48	Order Date Year	0.000041
61	Order City 3	0.000040
82	Order Country 2	0.000039
77	Customer City 7	0.000038
62	Order City 4	0.000034
66	Order City 8	0.000033
59	Order City 1	0.000032
70	Customer City 0	0.000030
87	Order Country 7	0.000029
64	Order City 6	0.000029
67	Order City 9	0.000029
23	Europe Market	0.000028
69	Order City 12	0.000028
83	Order Country 3	0.000026
84	Order Country 4	0.000024
25	Pacific Asia Market	0.000024
78	Customer City 8	0.000023
60	Order City 2	0.000023
21	Home Office Customer Segment	0.000022
65	Order City 7	0.000020
22	Africa Market	0.000019
19	Consumer Customer Segment	0.000018
36	Category Name 5	0.000018
32	Category Name 1	0.000016

	Column	Importance Score
79	Customer City 9	0.000016
81	Order Country 1	0.000015
39	Department Name 2	0.000015
26	USCA Market	0.000012
73	Customer City 3	0.000012
44	Product Name 3	0.000011
10	Order Item Quantity	0.000011
46	Product Name 5	0.000010
33	Category Name 2	0.000009
45	Product Name 4	0.000008
37	Department Name 0	0.000008
76	Customer City 6	0.000007
89	Customer Country 1	0.000007
47	Product Name 6	0.000007
41	Product Name 0	0.000007
31	Category Name 0	0.000006
38	Department Name 1	0.000005
42	Product Name 1	0.000005
40	Department Name 3	0.000004
34	Category Name 3	0.000004
35	Category Name 4	0.000004
43	Product Name 2	0.000000

Random Forest Classifier #2 (On 15 Most Important Features)

Let us consider the top 15 features and train another Random Forest model on only these features.

In [367…

```
scm_15 = scm_updated[['Days for shipping (real)',
                      'Days for shipment (scheduled)',
                      'Standard Class Shipping Mode',
                      'First Class Shipping Mode',
                      'Second Class Shipping Mode',
                      'Shipping Date Hour',
                      'Same Day Shipping Mode',
                      'Order Date Hour',
                      'Transfer Type',
                      'Shipping Date Day',
```

```
                       'Order Date Day',
                       'Debit Type',
                       'Payment Type',
                       'Latitude',
                       'Longitude',
                       'Late_delivery_risk']]
scm_15.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 180516 entries, 0 to 180518
Data columns (total 16 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   Days for shipping (real)        180516 non-null  int64  
 1   Days for shipment (scheduled)   180516 non-null  int64  
 2   Standard Class Shipping Mode    180516 non-null  float64
 3   First Class Shipping Mode       180516 non-null  float64
 4   Second Class Shipping Mode      180516 non-null  float64
 5   Shipping Date Hour              180516 non-null  int32  
 6   Same Day Shipping Mode          180516 non-null  float64
 7   Order Date Hour                 180516 non-null  int32  
 8   Transfer Type                   180516 non-null  float64
 9   Shipping Date Day               180516 non-null  int32  
 10  Order Date Day                  180516 non-null  int32  
 11  Debit Type                      180516 non-null  float64
 12  Payment Type                    180516 non-null  float64
 13  Latitude                        180516 non-null  float64
 14  Longitude                       180516 non-null  float64
 15  Late_delivery_risk              180516 non-null  int64  
dtypes: float64(9), int32(4), int64(3)
memory usage: 24.7 MB
```

In [368…
```
# now again splitting this dataset into train and test
X_15_train, X_15_test, y_15_train, y_15_test = train_test_split(scm_15.dr
```

In [369…
```
rf_clf_2 = RandomForestClassifier(random_state=23,
                                  n_jobs=-1,
                                  max_depth=5,
                                  n_estimators=100,
                                  oob_score=True)
```

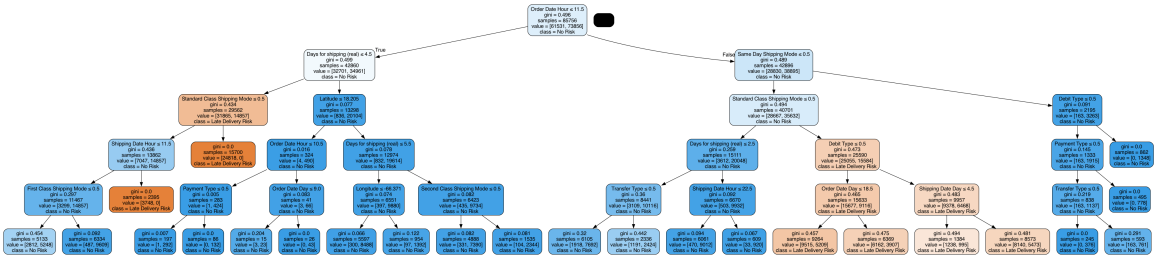
In [370…
```
rf_clf_2.fit(X_15_train, y_15_train)
```

Out[370…
```
RandomForestClassifier(max_depth=5, n_jobs=-1, oob_score=True, ra
ndom_state=23)
```

In [405…
```
# plotting the 2nd random forest model (1st estimator)

dot_data = StringIO()
export_graphviz(
        rf_clf_2.estimators_[1],
        out_file=dot_data,
        feature_names= X_15_train.columns.values.tolist(),
        class_names= ["Late Delivery Risk", "No Risk"],
        rounded=True,
        filled=True,
```

```
        special_characters=True
    )
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('random_forest_model_2.png')
Image(graph.create_png())
```

Out[405…



In [372…
```
rf_clf_2.score(X_15_train, y_15_train)
```

Out[372…
```
0.9754038423186865
```

In [373…
```
rf_clf_2.score(X_15_test, y_15_test)
```

Out[373…
```
0.975780540229121
```

Hyper Parameter Tuning

Randomized Search CV

In [374…
```
rf_clf_2.get_params()
```

Out[374…
```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': 5,
 'max_features': 'sqrt',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': -1,
 'oob_score': True,
 'random_state': 23,
 'verbose': 0,
 'warm_start': False}
```

In [378…
```
# Creating the random grid as a dict
random_grid = {'n_estimators': [25, 50, 100, 200, 400],
               'max_features': ['auto', 'sqrt'],
               'max_depth': [3, 5, 10, 15, 20],
               'min_samples_split': [2, 3, 5],
               'min_samples_leaf': [1, 2],
               'bootstrap': [True, False]}
```

In [379…
```
from sklearn.model_selection import RandomizedSearchCV

rf_random_search_cv = RandomizedSearchCV(estimator=rf_clf_2,
                                         param_distributions=random_grid,
                                         n_iter=25,
                                         cv=2,
                                         verbose=2,
                                         random_state=23,
                                         n_jobs=-1)
```

In [380…
```
%%time
rf_random_search_cv.fit(X_15_train, y_15_train)
```

```
Fitting 2 folds for each of 25 candidates, totalling 50 fits
```

```
112362.07s - pydevd: Sending message related to process being replaced tim
ed-out after 5 seconds
112362.07s - pydevd: Sending message related to process being replaced tim
ed-out after 5 seconds
112362.08s - pydevd: Sending message related to process being replaced tim
ed-out after 5 seconds
112362.09s - pydevd: Sending message related to process being replaced tim
ed-out after 5 seconds
112362.10s - pydevd: Sending message related to process being replaced tim
ed-out after 5 seconds
112362.11s - pydevd: Sending message related to process being replaced tim
ed-out after 5 seconds
112362.14s - pydevd: Sending message related to process being replaced tim
ed-out after 5 seconds
112362.15s - pydevd: Sending message related to process being replaced tim
ed-out after 5 seconds
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
```

```
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
[CV] END bootstrap=False, max_depth=3, max_features=auto, min_samples_leaf
=2, min_samples_split=5, n_estimators=50; total time=   0.1s
[CV] END bootstrap=False, max_depth=3, max_features=auto, min_samples_leaf
=1, min_samples_split=2, n_estimators=100; total time=   0.1s
[CV] END bootstrap=False, max_depth=3, max_features=auto, min_samples_leaf
=2, min_samples_split=5, n_estimators=50; total time=   0.1s
[CV] END bootstrap=False, max_depth=3, max_features=auto, min_samples_leaf
=1, min_samples_split=2, n_estimators=100; total time=   0.1s
[CV] END bootstrap=False, max_depth=3, max_features=auto, min_samples_leaf
=2, min_samples_split=5, n_estimators=400; total time=   0.0s
[CV] END bootstrap=False, max_depth=3, max_features=auto, min_samples_leaf
=2, min_samples_split=5, n_estimators=400; total time=   0.1s
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
[CV] END bootstrap=False, max_depth=5, max_features=auto, min_samples_leaf
=1, min_samples_split=5, n_estimators=400; total time=   0.0s
[CV] END bootstrap=False, max_depth=5, max_features=auto, min_samples_leaf
=1, min_samples_split=5, n_estimators=400; total time=   0.0s
```

```
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
[CV] END bootstrap=True, max_depth=10, max_features=auto, min_samples_leaf
=2, min_samples_split=5, n_estimators=50; total time= 1.1min
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
[CV] END bootstrap=True, max_depth=10, max_features=auto, min_samples_leaf
=1, min_samples_split=5, n_estimators=50; total time= 1.2min
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
[CV] END bootstrap=True, max_depth=10, max_features=auto, min_samples_leaf
=1, min_samples_split=5, n_estimators=50; total time= 1.2min
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
[CV] END bootstrap=True, max_depth=10, max_features=auto, min_samples_leaf
=2, min_samples_split=5, n_estimators=50; total time= 1.2min
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
[CV] END bootstrap=True, max_depth=20, max_features=auto, min_samples_leaf
=1, min_samples_split=5, n_estimators=25; total time=  40.6s
[CV] END bootstrap=True, max_depth=20, max_features=auto, min_samples_leaf
=1, min_samples_split=5, n_estimators=25; total time=  41.8s
[CV] END bootstrap=False, max_depth=15, max_features=sqrt, min_samples_lea
f=1, min_samples_split=3, n_estimators=50; total time=   0.0s
[CV] END bootstrap=False, max_depth=15, max_features=sqrt, min_samples_lea
f=1, min_samples_split=3, n_estimators=50; total time=   0.0s
```

```
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
[CV] END bootstrap=True, max_depth=10, max_features=auto, min_samples_leaf
=2, min_samples_split=2, n_estimators=200; total time= 4.0min
[CV] END bootstrap=False, max_depth=3, max_features=auto, min_samples_leaf
=2, min_samples_split=5, n_estimators=200; total time=   0.0s
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
[CV] END bootstrap=False, max_depth=3, max_features=auto, min_samples_leaf
=2, min_samples_split=5, n_estimators=200; total time=   0.2s
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
```

```
[CV] END bootstrap=False, max_depth=3, max_features=auto, min_samples_leaf
=2, min_samples_split=5, n_estimators=25; total time=   0.2s
[CV] END bootstrap=False, max_depth=3, max_features=auto, min_samples_leaf
=2, min_samples_split=5, n_estimators=25; total time=   0.1s
[CV] END bootstrap=True, max_depth=10, max_features=auto, min_samples_leaf
=2, min_samples_split=2, n_estimators=200; total time= 4.0min
[CV] END bootstrap=True, max_depth=15, max_features=sqrt, min_samples_leaf
=1, min_samples_split=2, n_estimators=50; total time=  53.9s
[CV] END bootstrap=False, max_depth=20, max_features=sqrt, min_samples_lea
f=2, min_samples_split=3, n_estimators=400; total time=   0.1s
[CV] END bootstrap=False, max_depth=20, max_features=sqrt, min_samples_lea
f=2, min_samples_split=3, n_estimators=400; total time=   0.1s
[CV] END bootstrap=True, max_depth=15, max_features=sqrt, min_samples_leaf
=1, min_samples_split=2, n_estimators=50; total time=  56.4s
[CV] END bootstrap=True, max_depth=20, max_features=sqrt, min_samples_leaf
=1, min_samples_split=2, n_estimators=50; total time=  54.6s
[CV] END bootstrap=False, max_depth=15, max_features=auto, min_samples_lea
f=2, min_samples_split=3, n_estimators=50; total time=   0.1s
```

```
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
```

```
[CV] END bootstrap=False, max_depth=15, max_features=auto, min_samples_lea
f=2, min_samples_split=3, n_estimators=50; total time=   0.2s
[CV] END bootstrap=False, max_depth=10, max_features=auto, min_samples_lea
f=1, min_samples_split=2, n_estimators=100; total time=   0.1s
```

```
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
```

```
[CV] END bootstrap=False, max_depth=10, max_features=auto, min_samples_lea
f=1, min_samples_split=2, n_estimators=100; total time=   0.0s
[CV] END bootstrap=False, max_depth=15, max_features=auto, min_samples_lea
f=2, min_samples_split=5, n_estimators=200; total time=   0.0s
```

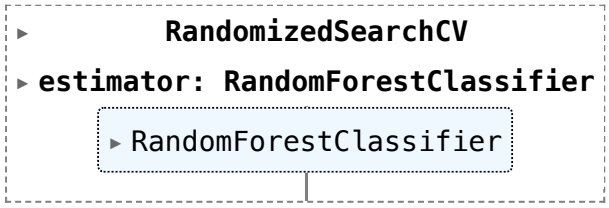
```
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
[CV] END bootstrap=False, max_depth=15, max_features=auto, min_samples_lea
f=2, min_samples_split=5, n_estimators=200; total time=   0.2s
[CV] END bootstrap=False, max_depth=15, max_features=sqrt, min_samples_lea
f=1, min_samples_split=5, n_estimators=100; total time=   0.0s
[CV] END bootstrap=True, max_depth=20, max_features=sqrt, min_samples_leaf
=1, min_samples_split=2, n_estimators=50; total time=  54.9s
[CV] END bootstrap=False, max_depth=15, max_features=sqrt, min_samples_lea
f=1, min_samples_split=5, n_estimators=100; total time=   0.1s
[CV] END bootstrap=True, max_depth=3, max_features=auto, min_samples_leaf=
2, min_samples_split=3, n_estimators=400; total time= 6.2min
[CV] END bootstrap=True, max_depth=3, max_features=auto, min_samples_leaf=
2, min_samples_split=3, n_estimators=400; total time= 6.2min
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
[CV] END bootstrap=True, max_depth=3, max_features=auto, min_samples_leaf=
1, min_samples_split=5, n_estimators=400; total time= 5.8min
[CV] END bootstrap=True, max_depth=3, max_features=auto, min_samples_leaf=
1, min_samples_split=5, n_estimators=400; total time= 5.8min
[CV] END bootstrap=False, max_depth=5, max_features=auto, min_samples_leaf
=2, min_samples_split=2, n_estimators=200; total time=   0.1s
[CV] END bootstrap=False, max_depth=5, max_features=auto, min_samples_leaf
=2, min_samples_split=2, n_estimators=200; total time=   0.1s
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
```

```
[CV] END bootstrap=False, max_depth=15, max_features=auto, min_samples_lea
f=2, min_samples_split=3, n_estimators=25; total time=   0.1s
[CV] END bootstrap=False, max_depth=15, max_features=auto, min_samples_lea
f=2, min_samples_split=3, n_estimators=25; total time=   0.1s
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning: `max_features='auto'` has been dep
recated in 1.1 and will be removed in 1.3. To keep the past behaviour, exp
licitly set `max_features='sqrt'` or remove this parameter as it is also t
he default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
[CV] END bootstrap=True, max_depth=20, max_features=sqrt, min_samples_leaf
=2, min_samples_split=3, n_estimators=100; total time= 1.6min
[CV] END bootstrap=True, max_depth=20, max_features=sqrt, min_samples_leaf
=2, min_samples_split=3, n_estimators=100; total time= 1.6min
[CV] END bootstrap=True, max_depth=5, max_features=auto, min_samples_leaf=
1, min_samples_split=2, n_estimators=400; total time= 5.9min
[CV] END bootstrap=True, max_depth=5, max_features=auto, min_samples_leaf=
1, min_samples_split=2, n_estimators=400; total time= 5.9min
[CV] END bootstrap=True, max_depth=3, max_features=auto, min_samples_leaf=
2, min_samples_split=5, n_estimators=400; total time= 2.5min
[CV] END bootstrap=True, max_depth=3, max_features=auto, min_samples_leaf=
2, min_samples_split=5, n_estimators=400; total time= 2.5min
```

```
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
model_selection/_validation.py:378: FitFailedWarning:


28 fits failed out of a total of 50.
The score on these train-test partitions for these parameters will be set
to nan.
If these failures are not expected, you can try to debug them by setting e
rror_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------
------
28 fits failed with the following error:
Traceback (most recent call last):
  File "/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/
sklearn/model_selection/_validation.py", line 686, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/
sklearn/ensemble/_forest.py", line 437, in fit
    raise ValueError("Out of bag estimation only available if bootstrap=Tr
ue")
ValueError: Out of bag estimation only available if bootstrap=True


/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
model_selection/_search.py:953: UserWarning:

One or more of the test scores are non-finite: [0.97544077        nan
nan 0.97547032        nan 0.93599829
        nan 0.97543339 0.93599829 0.98377983        nan 0.96171717
        nan        nan 0.97927423        nan 0.98770931        nan
        nan        nan        nan 0.98256849 0.93599829        nan
        nan]

/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
model_selection/_search.py:968: RuntimeWarning:

invalid value encountered in cast

/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/
ensemble/_forest.py:427: FutureWarning:

`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.
3. To keep the past behaviour, explicitly set `max_features='sqrt'` or rem
ove this parameter as it is also the default value for RandomForestClassif
iers and ExtraTreesClassifiers.
```

```
CPU times: user 3min 8s, sys: 1.2 s, total: 3min 9s
Wall time: 9min 11s
```

Out[380…



In [381…

```
# Let us see the best parameters
rf_random_search_cv.best_params_
```

Out[381…
```
{'n_estimators': 50,
 'min_samples_split': 5,
 'min_samples_leaf': 2,
 'max_features': 'auto',
 'max_depth': 10,
 'bootstrap': True}
```

Random Forest Classifier #3 (RF + Randomized Search CV)

In [382…
```
rf_clf_3 = RandomForestClassifier(random_state=23,
                                  n_jobs=-1,
                                  max_depth=10,
                                  n_estimators=50,
                                  min_samples_split=5,
                                  min_samples_leaf=2,
                                  max_features='auto',
                                  bootstrap=True,
                                  oob_score=True)
```

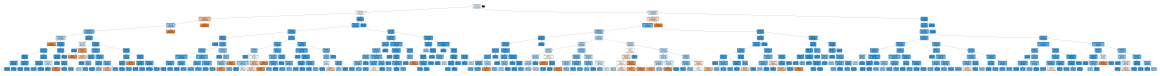
In [383…
```
rf_clf_3.fit(X_15_train, y_15_train)
```

```
/opt/homebrew/anaconda3/envs/info6105/lib/python3.8/site-packages/sklearn/ensemble/_forest.py:427: FutureWarning:

`max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
```

Out[383…
```
RandomForestClassifier(max_depth=10, max_features='auto', min_samples_leaf=2,
                       min_samples_split=5, n_estimators=50, n_jobs=-1,
                       oob_score=True, random_state=23)
```

In [406…
```
# plotting the 3rd random forest model (1st estimator)

dot_data = StringIO()
export_graphviz(
        rf_clf_3.estimators_[1],
        out_file=dot_data,
        feature_names= X_15_train.columns.values.tolist(),
        class_names= ["Late Delivery Risk", "No Risk"],
        rounded=True,
        filled=True,
        special_characters=True
    )
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('random_forest_model_3.png')
Image(graph.create_png())
```

Out[406…



In [384…
```
rf_clf_3.score(X_15_train, y_15_train)
```

Out[384…
```
0.9755146358217555
```

In [385…
```
rf_clf_3.score(X_15_test, y_15_test)
```

Out[385…
```
0.9758470163309624
```

Results

- Decision Tree #1 with 5 levels (mid-term project)
  - Training Accuracy: 46.40 %
  - Testing Accuracy: Decided Not to Test.
- Decision Tree #2 with 10 levels (mid-term project)
  - Training Accuracy: 51.82 %
  - Testing Accuracy: 51.86 %
- Decision Tree #3 with extensively preprocessed data (final project)
  - Training Accuracy: 65.96 %
  - Testing Accuracy: 66.08 %
- Random Forest #1 with extensively preprocessed data (final project)
  - Training Accuracy: 93.60 %
  - Testing Accuracy: 93.58 %
- Random Forest #2 with 15 most important features (final project)
  - Training Accuracy: 97.54 %
  - Testing Accuracy: 97.58 %
- Random Forest #3 with Randomized Search CV (final project)
  - Training Accuracy: 97.55 %
  - Testing Accuracy: 97.58 %