Managing Supply Chain Delays and Risk

Assignment 1 - Group 11

In [219]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from dmba import classificationSummary
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import accuracy_score
```

In [216]:
```
import warnings
warnings.filterwarnings('ignore')
```

Importing file

In [183]:
```
# Read the data file
delays_df = pd.read_excel('DataCoSupplyFull3.xlsx')
```

In [184]:
```
#Present 5 data records
delays_df.head(5)
```

Out[184]:

Benefit per order	Sales per customer	Delivery Status	Late_delivery_risk	Category Id	Category Name	Customer City	...	Order Status	Order Zipcode	Product Card Id	Product Category Id	Product I
91.250000	314.640015	Advance shipping	0	73	Sporting Goods	Caguas	...	COMPLETE	NaN	1360	73	http://images.acmesports.sports/Smart+
-249.089996	311.359985	Late delivery	1	73	Sporting Goods	Caguas	...	PENDING	NaN	1360	73	http://images.acmesports.sports/Smart+
-247.779999	309.720001	Shipping on time	0	73	Sporting Goods	San Jose	...	CLOSED	NaN	1360	73	http://images.acmesports.sports/Smart+
22.860001	304.809998	Advance shipping	0	73	Sporting Goods	Los Angeles	...	COMPLETE	NaN	1360	73	http://images.acmesports.sports/Smart+
134.210007	298.250000	Advance shipping	0	73	Sporting Goods	Caguas	...	PENDING_PAYMENT	NaN	1360	73	http://images.acmesports.sports/Smart+

In [185]:
```
delays_df.shape
delays_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180516 entries, 0 to 180515
Data columns (total 52 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   Type                           180516 non-null  object
 1   Days for shipping (real)       180516 non-null  int64
 2   Days for shipment (scheduled)  180516 non-null  int64
 3   Benefit per order              180516 non-null  float64
 4   Sales per customer             180516 non-null  float64
 5   Delivery Status                180516 non-null  object
 6   Late_delivery_risk             180516 non-null  int64
 7   Category Id                    180516 non-null  int64
 8   Category Name                  180516 non-null  object
 9   Customer City                  180516 non-null  object
 10  Customer Country               180516 non-null  object
 11  Customer Email                 180516 non-null  object
 12  Customer Fname                 180516 non-null  object
 13  Customer Id                    180516 non-null  int64
 14  Customer Lname                 180508 non-null  object
 15  Customer Password              180516 non-null  object
 16  Customer Segment               180516 non-null  object
 17  Customer State                 180516 non-null  object
 18  Customer Street                180516 non-null  object
 19  Customer Zipcode               180516 non-null  int64
 20  Department Id                  180516 non-null  int64
 21  Department Name                180516 non-null  object
 22  Latitude                       180516 non-null  float64
 23  Longitude                      180516 non-null  float64
 24  Market                         180516 non-null  object
 25  Order City                     180516 non-null  object
 26  Order Country                  180516 non-null  object
 27  Order Customer Id              180516 non-null  int64
 28  order date (DateOrders)        180516 non-null  object
 29  Order Id                       180516 non-null  int64
 30  Order Item Cardprod Id         180516 non-null  int64
 31  Order Item Discount            180516 non-null  float64
 32  Order Item Discount Rate       180516 non-null  float64
 33  Order Item Id                  180516 non-null  int64
 34  Order Item Product Price       180516 non-null  float64
 35  Order Item Profit Ratio        180516 non-null  float64
 36  Order Item Quantity            180516 non-null  int64
 37  Sales                          180516 non-null  float64
 38  Order Item Total               180516 non-null  float64
 39  Order Profit Per Order         180516 non-null  float64
 40  Order Region                   180516 non-null  object
 41  Order State                    180516 non-null  object
 42  Order Status                   180516 non-null  object
 43  Order Zipcode                  24840 non-null   float64
 44  Product Card Id                180516 non-null  int64
 45  Product Category Id            180516 non-null  int64
 46  Product Image                  180516 non-null  object
 47  Product Name                   180516 non-null  object
 48  Product Price                  180516 non-null  float64
 49  Product Status                 180516 non-null  int64
 50  shipping date (DateOrders)     180516 non-null  object
 51  Shipping Mode                  180516 non-null  object
dtypes: float64(13), int64(15), object(24)
memory usage: 71.6+ MB
```

Data PreProcessing

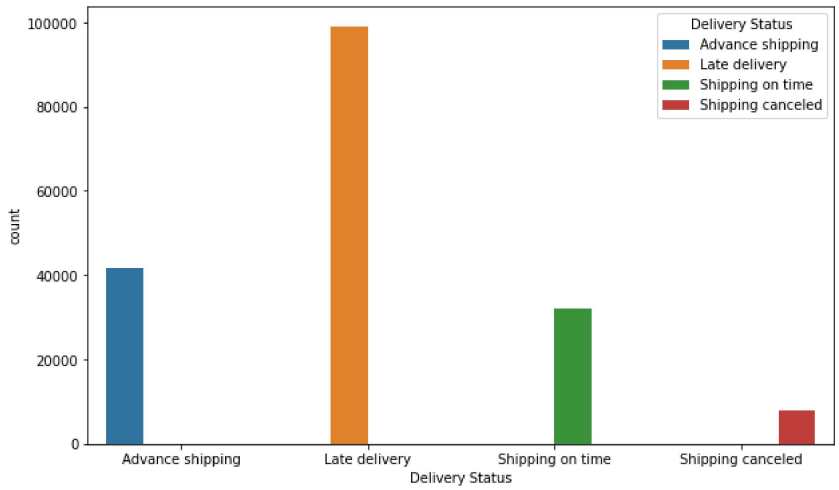
In [186]:
```
 'Delivery Status','Category Name','Customer Email','Customer Fname','Customer Lname','Customer Password','Customer Street'], axis=1)
der Id','Order Status','Order Zipcode','Product Image','Product Name','shipping date (DateOrders)','Customer Id','Product Status'], axis=1)
```

In [187]:
```
delays4 = delays3.rename(columns={"Days for shipment (scheduled)": "shipment_date", "Benefit per order": "Benefit_order", "Sales per custom
```

In [188]:
```
# Datapreprocessing
l1=LabelEncoder()
#perform label encoding on 'team' column
delays4['Type'] = l1.fit_transform(delays4['Type'])
delays4['Customer_City'] = l1.fit_transform(delays4['Customer_City'])
delays4['Customer_Country'] = l1.fit_transform(delays4['Customer_Country'])
delays4['Customer_Segment'] = l1.fit_transform(delays4['Customer_Segment'])
delays4['Customer_State'] = l1.fit_transform(delays4['Customer_State'])
delays4['Department_Name'] = l1.fit_transform(delays4['Department_Name'])
delays4['Market'] = l1.fit_transform(delays4['Market'])
delays4['Order_City'] = l1.fit_transform(delays4['Order_City'])
delays4['Order_Country'] = l1.fit_transform(delays4['Order_Country'])
delays4['Order_Region'] = l1.fit_transform(delays4['Order_Region'])
delays4['Order_State'] = l1.fit_transform(delays4['Order_State'])
delays4['Ship_Mod'] = l1.fit_transform(delays4['Ship_Mod'])
```

Data Exploration

In [189]:
```
sns.countplot(data=delays_df, x="Delivery Status", hue="Delivery Status")
plt.gcf().set_size_inches(10, 6)

# Show the plot
plt.show()
```

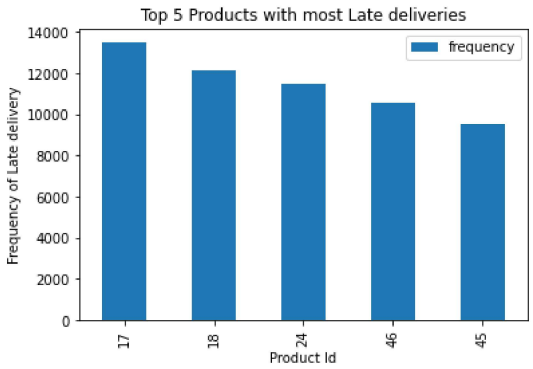


In [190]:
```
suspected_fraud_df = delays_df[delays_df['Order Status'] == 'SUSPECTED_FRAUD']
```

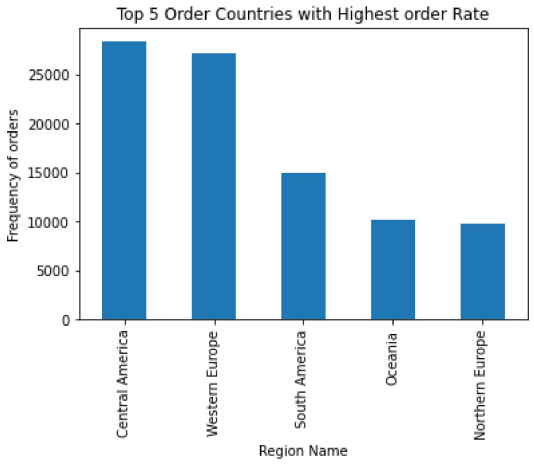
In [191]:
```
top_5 = suspected_fraud_df['Order Country'].value_counts()[:5]

# Plot the top 5 values as a bar plot
top_5.plot(kind='bar')

plt.xlabel('Country Name')
plt.ylabel('Frequency of frauds')
plt.title('Top 5 Order Countries with Highest Fraud Rate')

plt.show()
```

In [192]:
```
top_5_2 = suspected_fraud_df['Customer Country'].value_counts()

top_5_2.plot(kind='bar')

plt.xlabel('Country Name')
plt.ylabel('Frequency of frauds')
plt.title('Top Customer Countries with Highest Fraud Rate')

plt.show()
```

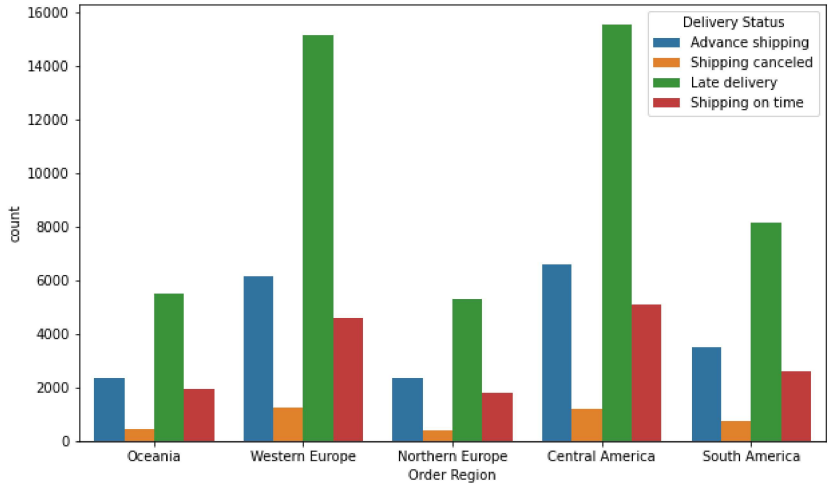
In [193]:
```
grouped_data = delays_df.groupby('Delivery Status')['Category Id'].value_counts()

# Reset the index and rename the columns for better readability
grouped_data = grouped_data.reset_index(name='frequency')
grouped_data = grouped_data.rename(columns={'Category Id': 'value', 'Delivery Status': 'group'})

result = grouped_data[grouped_data['group'] == "Late delivery"][:5]

result.plot.bar(x='value', y='frequency')

plt.xlabel('Product Id')
plt.ylabel('Frequency of Late delivery')
plt.title('Top 5 Products with most Late deliveries')

plt.show()
```



In [194]:
```
top_5 = delays_df['Order Region'].value_counts()[:5]

# Plot the top 5 values as a bar plot
top_5.plot(kind='bar')

plt.xlabel('Region Name')
plt.ylabel('Frequency of orders')
plt.title('Top 5 Order Countries with Highest order Rate')

plt.show()
```



In [195]:
```
sns.countplot(data=delays_df[delays_df['Order Region'].isin(top_5.index)], x="Order Region", hue="Delivery Status")
plt.gcf().set_size_inches(10, 6)

# Show the plot
plt.show()
```



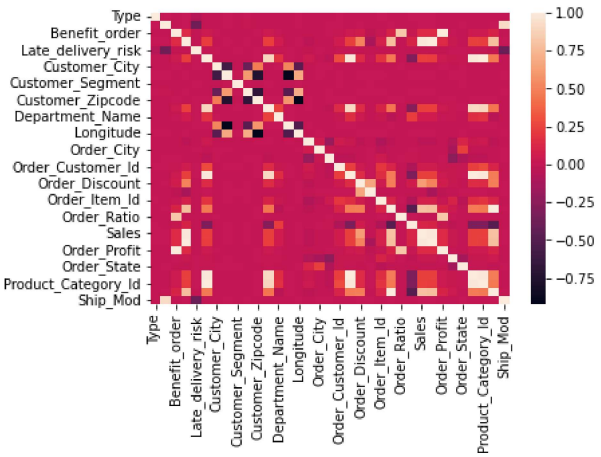
In [196]:
```
Corr_Matrix = delays4.corr()
```

In [197]:
```
Corr_Matrix['Late_delivery_risk']
```

Out[197]:
```
Type                    -0.061528
shipment_date           -0.369342
Benefit_order           -0.003728
Sales_customer          -0.003793
Late_delivery_risk       1.000000
Category_Id              0.001770
Customer_City            0.005075
Customer_Country        -0.001049
Customer_Segment         0.001421
Customer_State          -0.001854
Customer_Zipcode         0.003148
Department_Id            0.001071
Department_Name          0.002332
Latitude                 0.000682
Longitude               -0.001926
Market                  -0.000575
Order_City               0.003839
Order_Country           -0.001641
Order_Customer_Id        0.001501
Order_Item_Cardprod_Id   0.001504
Order_Discount          -0.000755
Order_Discount_Rate      0.000392
Order_Item_Id           -0.001364
Order_Price             -0.002174
Order_Ratio             -0.002315
Order_Quantity          -0.000145
Sales                   -0.003567
Order_Total             -0.003793
Order_Profit            -0.003728
Order_Region             0.006153
Order_State              0.001221
Product_Card             0.001504
Product_Category_Id      0.001770
Product_Price           -0.002174
Ship_Mod                -0.401365
Name: Late_delivery_risk, dtype: float64
```

In [198]:
```
sns.heatmap(delays4.corr())
```

Out[198]:
```
<AxesSubplot:>
```



Splitting the Data and Training the Dataset

Select columns to use as predictors

In [199]:
```
X = delays4.drop(labels=['Type','Benefit_order','Sales_customer','Late_delivery_risk','Department_Id','Order_City','Order_Country','Order_C
y = delays4['Late_delivery_risk']
classes = sorted(y.unique())
```

```
0         0
1         1
2         0
3         0
4         0
         ..
180511    0
180512    1
180513    1
180514    0
180515    0
Name: Late_delivery_risk, Length: 180516, dtype: int64
```

In [224]:
```
train_X, valid_X, train_y, valid_y = train_test_split(X, y, test_size=0.4, random_state=1)

clf = MLPClassifier(hidden_layer_sizes=(3), activation='logistic', solver='lbfgs',random_state=1)
clf.fit(train_X, train_y.values)
```

Out[224]:
```
MLPClassifier(activation='logistic', hidden_layer_sizes=3, random_state=1,
              solver='lbfgs')
```

In [225]:
```
# Network structure

print('Intercepts')
print(clf.intercepts_)
print('Weights')
print(clf.coefs_)

# Prediction
print(pd.concat([delays4,pd.DataFrame(clf.predict_proba(X), columns=classes)], axis=1))
```

```
Intercepts
[array([ 0.11306866, -0.1824051 , -0.14827764]), array([-0.19651509])]
Weights
[array([[-0.05255192,  0.14145285, -0.31615537],
       [-0.12672021, -0.22327072, -0.25778731],
       [-0.20083359, -0.08207337, -0.06505129],
       [ 0.02451299, -0.05114816,  0.11714676],
       [-0.18694246,  0.23909603, -0.29890144],
       [ 0.10655979, -0.0533932 ,  0.03724341],
       [-0.25418259, -0.18507062,  0.19286732],
       [ 0.29604426, -0.11796289,  0.12165384],
       [ 0.2373769 ,  0.24999899, -0.26237392],
       [-0.28908188, -0.20776896,  0.23891485],
       [-0.25409475, -0.04990828,  0.28959842],
       [ 0.01849463,  0.12483472, -0.11447661],
       [ 0.11784503,  0.21168437, -0.30465756],
       [ 0.12058081,  0.31953771,  0.15778361],
       [-0.14056377,  0.18309676, -0.25090151],
       [-0.03543596,  0.26189945, -0.12832833],
       [-0.13427252, -0.23218984, -0.30398264]]), array([[ 0.03130856],
       [-0.0008249 ],
       [ 0.39908035]])]
        Type  shipment_date  Benefit_order  Sales_customer  \
0          1              4      91.250000      314.640015
1          3              4    -249.089996      311.359985
2          0              4    -247.779999      309.720001
3          1              4      22.860001      304.809998
4          2              4     134.210007      298.250000
...      ...            ...            ...             ...
180511     0              4      40.000000      399.980011
180512     1              2    -613.770019      395.980011
180513     3              4     141.110001      391.980011
180514     2              4     186.229996      387.980011
180515     2              4     168.949997      383.980011

        Late_delivery_risk  Category_Id  Customer_City  Customer_Country  \
0                        0           73             65                 1
1                        1           73             65                 1
2                        0           73            451                 0
3                        0           73            284                 0
4                        0           73             65                 1
...                    ...          ...            ...               ...
180511                   0           45             59                 0
180512                   1           45             26                 0
180513                   1           45             55                 0
180514                   0           45             65                 1
180515                   0           45             65                 1

        Customer_Segment  Customer_State  ...  Order_Total  Order_Profit  \
0                      0              34  ...   314.640015     91.250000
1                      0              34  ...   311.359985   -249.089996
2                      0               3  ...   309.720001   -247.779999
3                      2               3  ...   304.809998     22.860001
4                      1              34  ...   298.250000    134.210007
...                  ...             ...  ...          ...           ...
180511                 2              29  ...   399.980011     40.000000
180512                 1               3  ...   395.980011   -613.770019
180513                 1               5  ...   391.980011    141.110001
180514                 0              34  ...   387.980011    186.229996
180515                 0              34  ...   383.980011    168.949997

        Order_Region  Order_State  Product_Card  Product_Category_Id  \
0                 15          475          1360                   73
1                 13          841          1360                   73
2                 13          841          1360                   73
3                 11          835          1360                   73
4                 11          835          1360                   73
...              ...          ...           ...                  ...
180511             7          913          1004                   45
180512             7          770          1004                   45
180513            11           88          1004                   45
180514            11           88          1004                   45
180515            13          967          1004                   45

        Product_Price  Ship_Mod         0         1
0          327.750000         3  0.449735  0.550265
1          327.750000         3  0.449735  0.550265
2          327.750000         3  0.449531  0.550469
3          327.750000         3  0.449531  0.550469
4          327.750000         3  0.449735  0.550265
...               ...       ...       ...       ...
180511     399.980011         3  0.449531  0.550469
180512     399.980011         2  0.449531  0.550469
180513     399.980011         3  0.449531  0.550469
180514     399.980011         3  0.449735  0.550265
180515     399.980011         3  0.449735  0.550265

[180516 rows x 37 columns]
```

Validation performance

In [226]:
```
classificationSummary(y, clf.predict(X), class_names=classes)
```

```
Confusion Matrix (Accuracy 0.5482)

       Prediction
Actual     0     1
     0    83 81457
     1   101 98875
```

Comparing results with Logistic Regression Model

In [208]:
```
LogAcc = []
delays5 = delays4
```

In [217]:
```
for x in range(10):
    delays5 = delays5.sample(frac = 1) #shuffling the dataframe

    #Split the data into training (80%) and testing (20%)
    X = delays4.drop(labels=['Type','Benefit_order','Sales_customer','Late_delivery_risk','Department_Id','Order_City','Order_Country','Ord
    y = delays4['Late_delivery_risk']
    classes = sorted(y.unique())

    # split into training and validation
    X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.20, random_state=1)

    # Run Logistic Regression
    log_regression = LogisticRegression()

    #Fit the model
    log_regression.fit(X_train,y_train)

    #Predictions and accuracy
    y_valid_pred = log_regression.predict(X_valid)
    Temp = metrics.accuracy_score(y_valid,y_valid_pred)

    #Appending the data to list
    LogAcc.append(Temp)
```

Accuracy for logistic Regression Model

In [220]:
```
np.average(LogAcc)
```

Out[220]:
```
0.6521438067804122
```

In [ ]: