In [152]:

```
#Import libraries
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
import seaborn as sns

from sklearn import tree, metrics, preprocessing
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from collections import defaultdict
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import precision_score, accuracy_score, recall_score, f1

from dmba import liftChart , gainsChart
from dmba import regressionSummary , classificationSummary, exhaustive_search

from sklearn.neural_network import MLPClassifier
```

Dataset reading, data explaration and data cleaning

In [153]:

```
supply_df = pd.read_csv('DataCoSupplyChainDataset.csv', encoding='latin-1')
supply_df.head(10)
```

Out[153]:

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

10 rows × 53 columns

In [154]:

```
#renames columns: replace spaces with _
supply_df.columns = [s.strip().replace(' ','_') for s in supply_df.columns]
```

In [155]:

```
supply_df.head(10)
```

Out[155]:

	Type	Days_for_shipping_(real)	Days_for_shipment_(scheduled)	Benefit_per_order	Sal
0	DEBIT	3	4	91.250000	
1	TRANSFER	5	4	-249.089996	
2	CASH	4	4	-247.779999	
3	DEBIT	3	4	22.860001	
4	PAYMENT	2	4	134.210007	
5	TRANSFER	6	4	18.580000	
6	DEBIT	2	1	95.180000	
7	TRANSFER	2	1	68.430000	
8	CASH	3	2	133.720001	
9	CASH	2	1	132.149994	

10 rows × 53 columns

In [156]:

```
#Remove columns based on research
drop_supply = supply_df
df_drop = drop_supply.drop(columns=['Benefit_per_order','Sales_per_customer',
                                    'Order_Zipcode','Product_Card_Id','Product
                                    'Product_Image','Product_Name','Product_Pr
                                     'Customer_Email','Customer_Fname','Custom
                                     'Order_Item_Total','Order_Profit_Per_Orde
                                     'Order_Status','Customer_Lname','Customer
                                     'Customer_State','Customer_Street','order
                                     'Order_Id','Order_Item_Cardprod_Id','Orde
                                     'Order_Item_Discount_Rate','Order_Item_Id
                                     'Order_Item_Quantity','Customer_Zipcode',
                                     'Latitude','Longitude','Market','Order_Cu
                                     'Delivery_Status','Product_Status'])
```

In [157]:
```
df_drop.head(10)
```

Out[157]:

	Days_for_shipping_(real)	Days_for_shipment_(scheduled)	Late_delivery_risk	Customer_City
0	3	4	0	Caguas
1	5	4	1	Caguas
2	4	4	0	San Jose
3	3	4	0	Los Angeles
4	2	4	0	Caguas
5	6	4	0	Tonawanda
6	2	1	1	Caguas
7	2	1	1	Miami
8	3	2	1	Caguas
9	2	1	1	San Ramon

In [158]:
```
df_drop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180519 entries, 0 to 180518
Data columns (total 8 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   Late_delivery_risk            180519 non-null  int64
 3   Customer_City                 180519 non-null  object
 4   Customer_Country              180519 non-null  object
 5   Order_City                    180519 non-null  object
 6   Order_Country                 180519 non-null  object
 7   Shipping_Mode                 180519 non-null  object
dtypes: int64(3), object(5)
memory usage: 11.0+ MB
```

In [159]:
```
#Encoding of columns with data type being object
le = preprocessing.LabelEncoder()
df_drop['Customer_City']  = le.fit_transform(df_drop['Customer_City'])
df_drop['Customer_Country']  = le.fit_transform(df_drop['Customer_Country'])
df_drop['Order_City']  = le.fit_transform(df_drop['Order_City'])
df_drop['Order_Country']  = le.fit_transform(df_drop['Order_Country'])
df_drop['Shipping_Mode']  = le.fit_transform(df_drop['Shipping_Mode'])
```

In [160]:

```
#Gives a summary of the data being analysed
df_drop.describe()
```

Out[160]:

	Days_for_shipping_(real)	Days_for_shipment_(scheduled)	Late_delivery_risk	Customer_C
count	180519.000000	180519.000000	180519.000000	180519.000
mean	3.497654	2.931847	0.548291	193.986
std	1.623722	1.374449	0.497664	160.930
min	0.000000	0.000000	0.000000	0.000
25%	2.000000	2.000000	0.000000	66.000
50%	3.000000	4.000000	1.000000	98.000
75%	5.000000	4.000000	1.000000	324.000
max	6.000000	4.000000	1.000000	562.000

In [161]:

```
#creates a correlation chart for the qualitative fields
supply_cor=df_drop.corr().round(2)
supply_cor
```

Out[161]:

	Days_for_shipping_(real)	Days_for_shipment_(scheduled)	Late_
Days_for_shipping_(real)	1.00	0.52	
Days_for_shipment_(scheduled)	0.52	1.00	
Late_delivery_risk	0.40	-0.37	
Customer_City	-0.00	-0.01	
Customer_Country	0.00	0.01	
Order_City	0.00	-0.00	
Order_Country	0.00	0.00	
Shipping_Mode	0.52	0.92	

In [162]:
```
df_drop
```

Out[162]:

	Days_for_shipping_(real)	Days_for_shipment_(scheduled)	Late_delivery_risk	Customer_
0	3	4	0	
1	5	4	1	
2	4	4	0	
3	3	4	0	
4	2	4	0	
...	...	...	...	
180514	4	4	0	
180515	3	2	1	
180516	5	4	1	
180517	3	4	0	
180518	4	4	0	

180519 rows × 8 columns

In [163]:
```
df_drop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180519 entries, 0 to 180518
Data columns (total 8 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   Late_delivery_risk            180519 non-null  int64
 3   Customer_City                 180519 non-null  int32
 4   Customer_Country              180519 non-null  int32
 5   Order_City                    180519 non-null  int32
 6   Order_Country                 180519 non-null  int32
 7   Shipping_Mode                 180519 non-null  int32
dtypes: int32(5), int64(3)
memory usage: 7.6 MB
```

Neural Network Implementation

Selecting Predictors and Outcome

In [164]:
```
outcome = 'Late_delivery_risk'
predictors = [c for c in df_drop.columns if c != outcome]
```

In [165]:
```
#Data Partition with test size = 40%
X = df_drop[predictors]
y = df_drop[outcome]
train_X, valid_X, train_y, valid_y = train_test_split(X, y, test_size=0.4, ra
```

1) NN model run using most popular, one hidden layer with 1 hidden node

In [166]:
```
# train neural network with 1 hidden nodes
clf = MLPClassifier(hidden_layer_sizes=(1), activation='logistic', solver='lb
clf.fit(train_X, train_y.values)
```

Out[166]:
```
MLPClassifier(activation='logistic', hidden_layer_sizes=1, max_iter=1000,
              random_state=1, solver='lbfgs')
```

In [167]:
```
clf.predict(X)
```

Out[167]:
```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

In [168]:
```
#NN Model Evaluation
# training performance
classificationSummary(train_y, clf.predict(train_X))

# validation performance
classificationSummary(valid_y, clf.predict(valid_X))
```

```
Confusion Matrix (Accuracy 0.5487)

       Prediction
Actual     0     1
     0     0 48884
     1     0 59427
Confusion Matrix (Accuracy 0.5477)

       Prediction
Actual     0     1
     0     0 32658
     1     0 39550
```

In [169]:

```
# Network structure

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

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

# Prediction

print(pd.concat([df_drop,pd.DataFrame(clf.predict_proba(X))], axis=1))
```

```
Intercepts
[array([-0.15443927]), array([0.19528739])]
Weights
[array([[-0.08297799],
       [ 0.22032449],
       [-0.49988562],
       [-0.19766743],
       [-0.35324411],
       [-0.4076614 ],
       [-0.31373979]]), array([[-0.20646505]])]
        Days_for_shipping_(real)  Days_for_shipment_(scheduled)  \
0                              3                              4
1                              5                              4
2                              4                              4
3                              3                              4
4                              2                              4
...                          ...                            ...
180514                         4                              4
180515                         3                              2
180516                         5                              4
180517                         3                              4
180518                         4                              4

        Late_delivery_risk  Customer_City  Customer_Country  Order_City
\
0                        0             66                 1         331
1                        1             66                 1         391
2                        0            452                 0         391
3                        0            285                 0        3226
4                        0             66                 1        3226
...                    ...            ...               ...         ...
180514                   0             59                 0        2922
180515                   1             26                 0        1362
180516                   1             55                 0          25
180517                   0             66                 1          25
180518                   0             66                 1        2203

        Order_Country  Shipping_Mode         0         1
0                  70              3  0.451333  0.548667
1                  69              3  0.451333  0.548667
2                  69              3  0.451333  0.548667
3                   8              3  0.451333  0.548667
4                   8              3  0.451333  0.548667
```

```
...                ...             ...       ...       ...
180514              31              3  0.451333  0.548667
180515              77              2  0.451333  0.548667
180516               8              3  0.451333  0.548667
180517               8              3  0.451333  0.548667
180518              69              3  0.451333  0.548667

[180519 rows x 10 columns]
```

In [170]:

```
clf.predict(X)
```

Out[170]:

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

In [171]:

```
#NN Model Evaluation
# training performance
classificationSummary(train_y, clf.predict(train_X))

# validation performance
classificationSummary(valid_y, clf.predict(valid_X))
```

```
Confusion Matrix (Accuracy 0.5487)

       Prediction
Actual     0     1
     0     0 48884
     1     0 59427
Confusion Matrix (Accuracy 0.5477)

       Prediction
Actual     0     1
     0     0 32658
     1     0 39550
```

2) NN Model Run using - Midway between input and output layers - 4 nodes (1HL + 4 nodes)

In [172]:

```
# train neural network with 4 hidden nodes
clf = MLPClassifier(hidden_layer_sizes=(4), activation='logistic', solver='lb
clf.fit(train_X, train_y.values)
```

Out[172]:

```
MLPClassifier(activation='logistic', hidden_layer_sizes=4, max_iter=1000,
              random_state=1, solver='lbfgs')
```

In [173]:

```
clf.predict(X)
```

Out[173]:

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

In [174]:

```
#NN Model Evaluation
# training performance
classificationSummary(train_y, clf.predict(train_X))

# validation performance
classificationSummary(valid_y, clf.predict(valid_X))
```

```
Confusion Matrix (Accuracy 0.5489)

       Prediction
Actual     0     1
     0   304 48580
     1   282 59145
Confusion Matrix (Accuracy 0.5472)

       Prediction
Actual     0     1
     0   163 32495
     1   204 39346
```

In [175]:

```
# Network structure

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

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

# Prediction

print(pd.concat([df_drop,pd.DataFrame(clf.predict_proba(X))], axis=1))
```

```
Intercepts
[array([-0.28141245,  0.32248063, -0.34251368, -0.06639009]), array([0.
19503204])]
Weights
[array([[-0.01207942,  0.18789204, -0.42626937, -0.19190578],
       [-0.3421372 , -0.34765781, -0.26754279, -0.10656541],
       [ 0.00720763,  0.03306329, -0.06849163,  0.08155611],
       [-0.25341969,  0.32245875, -0.40304458,  0.14681218],
       [-0.21381403,  0.05004637, -0.30667889, -0.31062711],
       [ 0.23248867,  0.39933333, -0.15907733,  0.24068873],
       [ 0.29008214,  0.33651878, -0.35387732, -0.37312586]]), array([[
0.26694911],
       [ 0.00137467],
       [ 0.24278099],
       [-0.23761444]])]
        Days_for_shipping_(real)  Days_for_shipment_(scheduled)  \
0                              3                               4
1                              5                               4
2                              4                               4
3                              3                               4
4                              2                               4
...                          ...                             ...
180514                         4                               4
180515                         3                               2
180516                         5                               4
180517                         3                               4
180518                         4                               4

        Late_delivery_risk  Customer_City  Customer_Country  Order_City
\
0                        0             66                 1         331
1                        1             66                 1         391
2                        0            452                 0         391
3                        0            285                 0        3226
4                        0             66                 1        3226
...                    ...            ...               ...         ...
180514                   0             59                 0        2922
180515                   1             26                 0        1362
180516                   1             55                 0          25
180517                   0             66                 1          25
180518                   0             66                 1        2203

        Order_Country  Shipping_Mode         0         1
0                  70              3  0.451056  0.548944
```

```
1                   69              3  0.451056  0.548944
2                   69              3  0.451056  0.548944
3                    8              3  0.451056  0.548944
4                    8              3  0.451056  0.548944
...                ...            ...       ...       ...
180514              31              3  0.451056  0.548944
180515              77              2  0.451056  0.548944
180516               8              3  0.450919  0.549081
180517               8              3  0.454428  0.545572
180518              69              3  0.451056  0.548944

[180519 rows x 10 columns]
```

In [176]:

```
clf.predict(X)
```

Out[176]:

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

In [177]:

```
#NN Model Evaluation
# training performance
classificationSummary(train_y, clf.predict(train_X))

# validation performance
classificationSummary(valid_y, clf.predict(valid_X))
```

```
Confusion Matrix (Accuracy 0.5489)

       Prediction
Actual     0     1
     0   304 48580
     1   282 59145
Confusion Matrix (Accuracy 0.5472)

       Prediction
Actual     0     1
     0   163 32495
     1   204 39346
```

3) NN model run using - Less than 2X the input nodes = [12 < 14 (2*7) nodes] [1 HL + 12 nodes]

In [178]:

```
# train neural network with 12 hidden nodes
clf = MLPClassifier(hidden_layer_sizes=(12), activation='logistic', solver='l
clf.fit(train_X, train_y.values)
```

Out[178]:

```
MLPClassifier(activation='logistic', hidden_layer_sizes=12, max_iter=1000,
              random_state=1, solver='lbfgs')
```

In [179]:

```
clf.predict(X)
```

Out[179]:

```
array([0, 1, 0, ..., 1, 0, 0], dtype=int64)
```

In [180]:

```
#NN Model Evaluation
# training performance
classificationSummary(train_y, clf.predict(train_X))

# validation performance
classificationSummary(valid_y, clf.predict(valid_X))
```

```
Confusion Matrix (Accuracy 0.9756)

       Prediction
Actual     0     1
     0 46243  2641
     1     0 59427
Confusion Matrix (Accuracy 0.9753)

       Prediction
Actual     0     1
     0 30876  1782
     1     0 39550
```

In [181]:

```python
# Network structure

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

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

# Prediction

print(pd.concat([df_drop,pd.DataFrame(clf.predict_proba(X))], axis=1))
```

```
Intercepts
[array([-1.09814394e-04,  3.34981375e-01, -8.12853508e-02,  3.41051254e
-01,
        1.05940525e-01,  4.88013057e-02, -4.03912945e-01,  3.02632189e-
01,
       -3.24695183e-02,  5.30223197e-02, -6.27318052e-02, -7.42228428e-
02]), array([-2.47954353])]
Weights
[array([[ 4.90204059e+00, -1.75021186e+00, -6.82621840e-01,
        -2.34211534e+00, -2.29435500e-01, -1.58898249e+00,
         7.68066858e+00, -8.42654572e-02, -6.69157956e-02,
         2.43529923e+00, -5.76275624e-02,  1.97493442e+01],
       [-3.99604001e+00,  2.61274480e+00, -6.50837636e-02,
         1.94308788e+00, -5.37738760e-02,  9.60045358e-01,
        -8.60982386e+00, -1.12404309e-01,  1.95091603e-01,
        -1.19611297e+00, -1.17413789e-01, -1.39600120e+01],
       [-5.48617010e+00, -4.62890440e-01, -2.89374759e+00,
         2.16790896e+00, -2.16890503e-01,  3.02136884e+00,
        -2.47502336e+00,  8.36567781e-01,  2.98115964e-01,
         2.74719388e-01,  1.06548712e-01, -2.85099112e-03],
       [ 1.81814629e-01,  2.79352858e-01, -2.35278446e-01,
         1.94911667e-01,  3.17189958e-01,  1.28885252e-01,
        -1.00812127e-01,  2.24324279e-01, -2.57440258e-01,
        -1.12088148e-01,  2.66692493e-01,  6.71785944e-03],
       [-5.53293479e+00, -3.69017639e-01,  7.42013604e-01,
         3.86170675e+00, -1.87078145e-01,  6.03587346e+00,
         4.76169872e-01, -4.26520288e-01,  4.68160943e-02,
        -3.07265527e+00,  1.11978716e-01,  2.92141399e-03],
       [ 1.22404150e+00, -2.78886244e+00, -1.21751089e+00,
        -6.01109845e-02, -2.92224215e-01,  1.41960026e+00,
         3.85213122e+00, -8.25661524e-01,  2.88405586e-01,
         1.90437301e+00,  2.61485852e-01, -3.42501327e-03],
       [-3.61000762e+00,  2.14536191e+00,  1.52954237e-01,
         1.36456289e+00,  2.77377949e-01,  7.08617575e-01,
        -6.52020254e+00,  2.08217272e-01,  2.48629655e-01,
        -1.13367296e+00,  1.69561173e-01, -1.21959700e+01]]), array([[1
5.3477442 ],
       [-8.01615555],
       [ 0.1868243 ],
       [-1.71493353],
       [-0.13513713],
       [-0.92380687],
       [-0.34239364],
       [-0.2799659 ],
```

```
       [-2.30891622],
       [ 0.40786926],
       [-3.01973076],
       [13.72146452]])]
        Days_for_shipping_(real)  Days_for_shipment_(scheduled)  \
0                              3                               4
1                              5                               4
2                              4                               4
3                              3                               4
4                              2                               4
...                          ...                             ...
180514                         4                               4
180515                         3                               2
180516                         5                               4
180517                         3                               4
180518                         4                               4

        Late_delivery_risk  Customer_City  Customer_Country  Order_City
\
0                        0             66                 1         331
1                        1             66                 1         391
2                        0            452                 0         391
3                        0            285                 0        3226
4                        0             66                 1        3226
...                    ...            ...               ...         ...
180514                   0             59                 0        2922
180515                   1             26                 0        1362
180516                   1             55                 0          25
180517                   0             66                 1          25
180518                   0             66                 1        2203

        Order_Country  Shipping_Mode         0         1
0                  70              3  0.999979  0.000021
1                  69              3  0.042858  0.957142
2                  69              3  0.999978  0.000022
3                   8              3  0.999975  0.000025
4                   8              3  0.999975  0.000025
...               ...            ...       ...       ...
180514             31              3  0.999973  0.000027
180515             77              2  0.042347  0.957653
180516              8              3  0.049034  0.950966
180517              8              3  0.999978  0.000022
180518             69              3  0.999975  0.000025

[180519 rows x 10 columns]
```

In [182]:

```
clf.predict(X)
```

Out[182]:

```
array([0, 1, 0, ..., 1, 0, 0], dtype=int64)
```

In [183]:

```
#NN Model Evaluation
# training performance
classificationSummary(train_y, clf.predict(train_X))

# validation performance
classificationSummary(valid_y, clf.predict(valid_X))
```

```
Confusion Matrix (Accuracy 0.9756)

       Prediction
Actual     0     1
     0 46243  2641
     1     0 59427
Confusion Matrix (Accuracy 0.9753)

       Prediction
Actual     0     1
     0 30876  1782
     1     0 39550
```

4) Hidden layer = 2/3 input nodes + output nodes = (2/3)*(7)+1 = 6 nodes [1HL + 6 Nodes]

In [184]:

```
# train neural network with 6 hidden nodes
clf = MLPClassifier(hidden_layer_sizes=(6), activation='logistic', solver='lb
clf.fit(train_X, train_y.values)
```

Out[184]:

```
MLPClassifier(activation='logistic', hidden_layer_sizes=6, max_iter=1000,
              random_state=1, solver='lbfgs')
```

In [191]:

```
clf.predict([[1,1,1,0,0,1,0]]).round(2)
```

```
C:\Users\kadam\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarnin
g: X does not have valid feature names, but MLPClassifier was fitted with f
eature names
  warnings.warn(
```

Out[191]:

```
array([0], dtype=int64)
```

In [192]:

```python
#NN Model Evaluation
# training performance
classificationSummary(train_y, clf.predict(train_X))

# validation performance
classificationSummary(valid_y, clf.predict(valid_X))
```

```
Confusion Matrix (Accuracy 0.9756)

       Prediction
Actual     0     1
     0 46243  2641
     1     0 59427
Confusion Matrix (Accuracy 0.9753)

       Prediction
Actual     0     1
     0 30876  1782
     1     0 39550
```

In [187]:

```python
# Network structure

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

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

# Prediction

print(pd.concat([df_drop,pd.DataFrame(clf.predict_proba(X))], axis=1))
```

```
Intercepts
[array([-0.14237831,  0.23479887,  0.40143298, -0.02168071, -0.5127101
6,
        0.17603056]), array([3.8075808])]
Weights
[array([[-8.03214864e-01,  2.39485041e-01,  3.20953437e+00,
        -3.10291160e+00, -5.69275097e+00, -7.70366086e+00],
       [ 1.22365119e+00, -1.49702632e-01,  4.02506696e-01,
         3.91619511e+00,  6.01058456e+00,  8.04861297e+00],
       [-4.11261267e+00,  7.25617867e-01,  2.70577460e+01,
        -1.17084881e+01,  7.04913688e-04,  3.89723598e+01],
       [-1.52117329e-01, -2.41030685e-01,  4.71131834e-01,
         5.01912727e-01,  1.93130244e-03,  3.58334768e-01],
       [-1.56234709e+01,  5.95750490e-03,  7.53289802e+00,
        -2.13250692e+00,  6.47314875e-05,  2.21319253e+01],
       [-6.75820774e+00,  1.21578245e-01,  3.65908727e+01,
        -8.37482260e+00, -1.84128238e-03,  3.16439060e+01],
       [ 1.37339671e+00,  2.46259874e-01,  2.68407584e-01,
         3.28626274e+00, -5.63934288e-01,  6.63229412e+00]]), array([[
34.37576706],
       [  3.44898489],
       [-36.04397664],
       [ 13.33885204],
       [-48.23071457],
       [ 31.96336326]])]
        Days_for_shipping_(real)  Days_for_shipment_(scheduled)  \
0                              3                               4
1                              5                               4
2                              4                               4
3                              3                               4
4                              2                               4
...                          ...                             ...
180514                         4                               4
180515                         3                               2
180516                         5                               4
180517                         3                               4
180518                         4                               4

        Late_delivery_risk  Customer_City  Customer_Country  Order_City
\
0                        0             66                 1         331
1                        1             66                 1         391
2                        0            452                 0         391
3                        0            285                 0        3226
```

```
4                        0             66                 1        3226
...                    ...            ...               ...         ...
180514                   0             59                 0        2922
180515                   1             26                 0        1362
180516                   1             55                 0          25
180517                   0             66                 1          25
180518                   0             66                 1        2203

        Order_Country  Shipping_Mode         0             1
0                  70              3  1.000000  4.184614e-20
1                  69              3  0.042476  9.575241e-01
2                  69              3  0.999997  3.195359e-06
3                   8              3  1.000000  3.574326e-20
4                   8              3  1.000000  2.712939e-20
...               ...            ...       ...           ...
180514             31              3  0.999995  5.042623e-06
180515             77              2  0.042347  9.576526e-01
180516              8              3  0.042678  9.573216e-01
180517              8              3  1.000000  4.025368e-20
180518             69              3  0.999984  1.552996e-05

[180519 rows x 10 columns]
```

In [193]:

```
clf.predict([[1,0,1,0,0,1,0]]).round(2)
```

```
C:\Users\kadam\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarnin
g: X does not have valid feature names, but MLPClassifier was fitted with f
eature names
  warnings.warn(
```

Out[193]:

```
array([1], dtype=int64)
```

In [194]:

```
#NN Model Evaluation
# training performance
classificationSummary(train_y, clf.predict(train_X))

# validation performance
classificationSummary(valid_y, clf.predict(valid_X))
```

```
Confusion Matrix (Accuracy 0.9756)

       Prediction
Actual     0     1
     0 46243  2641
     1     0 59427
Confusion Matrix (Accuracy 0.9753)

       Prediction
Actual     0     1
     0 30876  1782
     1     0 39550
```

Conclusion : Highest accuracy results (out of these 4 NN model runs using different hidden nodes) are obtained with 14 hidden nodes (3rd run - hidden layers = Less than 2X the input

nodes) layers and 6 hidden nodes (4th run - hidden layers = 2/3 input nodes + output nodes) [Both the results are same]

In [ ]: