```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
```

```
inv_data = pd.read_csv('Training_BOP.csv')
inv_data.head()
```

```
<ipython-input-4-e4a784c923cc>:1: DtypeWarning: Columns (0) have mixed types. Specify dtype option on im
  inv_data = pd.read_csv('Training_BOP.csv')
```

	sku	national_inv	lead_time	in_transit_qty	forecast_3_month	forecast_6_month	forecast_9_month
0	1026827	0.0	NaN	0.0	0.0	0.0	0.0
1	1043384	2.0	9.0	0.0	0.0	0.0	0.0
2	1043696	2.0	NaN	0.0	0.0	0.0	0.0
3	1043852	7.0	8.0	0.0	0.0	0.0	0.0
4	1044048	8.0	NaN	0.0	0.0	0.0	0.0

5 rows × 23 columns

```
# Calculate the total number of rows in the lead_time column
total_rows = inv_data['lead_time'].shape[0]

# Calculate the number of empty (NaN) cells in the lead_time column
empty_cells = inv_data['lead_time'].isna().sum()

# Calculate the percentage of empty cells
percentage_empty = (empty_cells / total_rows) * 100

# Print the result
print(f"Percentage of empty cells in 'lead_time': {percentage_empty:.2f}%")
```

```
Percentage of empty cells in 'lead_time': 5.98%
```

```
# Remove rows where 'lead_time' is NaN
inv_data_cleaned = inv_data.dropna(subset=['lead_time'])

# Drop the 'sku' column
inv_data_cleaned = inv_data_cleaned.drop(columns=['sku'])

# Optionally, you can reset the index if needed
inv_data_cleaned.reset_index(drop=True, inplace=True)

# Display the cleaned DataFrame
print(inv_data_cleaned)
```

```
1586963           0.0        2.0             0.0              10.0
1586964          -1.0        9.0             0.0               7.0
1586965          62.0        9.0            16.0              39.0
1586966          19.0        4.0             0.0               0.0

         forecast_6_month  forecast_9_month  sales_1_month  sales_3_month  \
0                     0.0               0.0            0.0            0.0
1                     0.0               0.0            0.0            0.0
2                     0.0               0.0            0.0            0.0
3                     0.0               0.0            0.0            0.0
4                     0.0               0.0            0.0            0.0
```

```
0                  0.0            0.0  ...             0.0            0.99
1                  0.0            0.0  ...             0.0            0.10
2                  0.0            0.0  ...             0.0            0.82
3                  0.0            0.0  ...             0.0            0.00
4                  0.0            0.0  ...             0.0            0.82
...                ...            ...  ...             ...             ...
1586962          849.0         1074.0  ...             0.0            0.85
1586963            7.0            7.0  ...             0.0            0.69
1586964           11.0           12.0  ...             0.0            0.86
1586965          153.0          205.0  ...             0.0            0.86
1586966           12.0           20.0  ...             0.0            0.73

         perf_12_month_avg  local_bo_qty  deck_risk  oe_constraint ppap_risk  \
0                     0.99           0.0         No             No        No
1                     0.13           0.0         No             No        No
2                     0.87           0.0         No             No        No
3                     0.00           0.0        Yes             No       Yes
4                     0.87           0.0         No             No        No
...                    ...           ...        ...            ...       ...
1586962               0.90           1.0         No             No        No
1586963               0.69           5.0        Yes             No        No
1586964               0.84           1.0        Yes             No        No
1586965               0.84           6.0         No             No        No
1586966               0.78           1.0         No             No        No

        stop_auto_buy rev_stop went_on_backorder
0                 Yes       No                No
1                 Yes       No                No
2                 Yes       No                No
3                 Yes       No                No
4                 Yes       No                No
...               ...      ...               ...
1586962           Yes       No                No
1586963           Yes       No                No
1586964            No       No               Yes
1586965           Yes       No                No
1586966           Yes       No                No

[1586967 rows x 22 columns]
```

```
inv_data_cleaned.to_csv('modified_training_bop.csv', index=False)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-7-133279b27e33> in <cell line: 0>()
----> 1 inv_data_cleaned.to_csv('modified_training_bop.csv', index=False)

11 frames
/usr/local/lib/python3.11/dist-packages/pandas/core/indexes/base.py in get_values_for_csv(values, date_format, na_rep, quoting, float_format, decimal)
   7832
   7833             if not quoting:
-> 7834                 values = values.astype(str)
   7835             else:
   7836                 values = np.array(values, dtype="object")

KeyboardInterrupt:
```

Using the modified file hereforth

```
mod_inv_data = pd.read_csv('modified_training_bop.csv')
mod_inv_data.head()
```

	national_inv	lead_time	in_transit_qty	forecast_3_month	forecast_6_month	forecast_9_month	sales_1_
0	2.0	9.0	0.0	0.0	0.0	0.0	
1	7.0	8.0	0.0	0.0	0.0	0.0	
2	13.0	8.0	0.0	0.0	0.0	0.0	
3	6.0	2.0	0.0	0.0	0.0	0.0	
4	4.0	8.0	0.0	0.0	0.0	0.0	

5 rows × 22 columns

```
mod_inv_data.shape
```

```
(349965, 22)
```

```
mod_inv_data.describe()
```

	national_inv	lead_time	in_transit_qty	forecast_3_month	forecast_6_month	forecast_9_month	sa
count	3.499650e+05	349965.000000	349965.000000	3.499650e+05	3.499650e+05	3.499650e+05	3
mean	4.576183e+02	7.865724	42.446185	1.906193e+02	3.714684e+02	5.433477e+02	
std	2.513422e+04	7.077886	1059.243965	5.147186e+03	1.022631e+04	1.507274e+04	
min	-1.349100e+04	0.000000	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	
25%	4.000000e+00	4.000000	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	
50%	1.400000e+01	8.000000	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	
75%	7.900000e+01	9.000000	0.000000	5.000000e+00	1.600000e+01	2.600000e+01	
max	1.233440e+07	52.000000	288960.000000	1.218328e+06	2.446072e+06	3.760840e+06	7

```
# Get the value counts for the 'went_on_backorder' column
number_of_values = mod_inv_data['went_on_backorder'].value_counts()

# Print the results
print(number_of_values)
```

```
went_on_backorder
No     346496
Yes      3469
Name: count, dtype: int64
```

There is a severe imbalance between No and Yes Values which is not ideal for a binary classification based model. Therefore, I will sample only some of the 'No' values in order to achieve a 60-40 distribution

```
# Separate the "yes" and "no" values
yes_values = mod_inv_data[mod_inv_data['went_on_backorder'] == 'Yes']
no_values = mod_inv_data[mod_inv_data['went_on_backorder'] == 'No']

# Randomly sample the "no" values to get 11,305 entries
no_values_sampled = no_values.sample(n=3000, random_state= 1)

# Randomly sample the "yes" values to get 11,305 entries
yes_values_sampled = yes_values.sample(n=3000, random_state= 1)

# Concatenate the "yes" values with the sampled "no" values
modified_data = pd.concat([yes_values_sampled, no_values_sampled])

# Optionally shuffle the resulting DataFrame
modified_data = modified_data.sample(frac=1, random_state= 1).reset_index(drop=True)

eq_data = modified_data['went_on_backorder'].value_counts()
```

```
# Check the new distribution
print(modified_data)
```

```
      national_inv  lead_time  in_transit_qty  forecast_3_month  \
0             -2.0        8.0             0.0              38.0
1              1.0        8.0             0.0              10.0
2              0.0        2.0             0.0               5.0
3              5.0        2.0             0.0               0.0
4            685.0        8.0             0.0               0.0
...            ...        ...             ...               ...
5995          -1.0        8.0             5.0              70.0
5996           8.0        8.0             0.0               0.0
5997         568.0       12.0           164.0             564.0
5998           8.0        8.0             0.0               0.0
5999         220.0       12.0             0.0               0.0

      forecast_6_month  forecast_9_month  sales_1_month  sales_3_month  \
0                 56.0              80.0            8.0           22.0
1                 10.0              10.0            3.0            8.0
2                  5.0               5.0            0.0            0.0
3                  0.0               0.0            0.0            0.0
4                  0.0               0.0            0.0            0.0
...                ...               ...            ...            ...
5995              91.0             126.0            6.0           24.0
5996               0.0               0.0            0.0            0.0
5997             993.0            1460.0          264.0          754.0
5998               3.0               6.0            1.0            5.0
5999               0.0               0.0            0.0            2.0

      sales_6_month  sales_9_month  ...  pieces_past_due perf_6_month_avg  \
0              36.0           42.0  ...              0.0             0.74
1               9.0            9.0  ...              0.0             1.00
2               0.0            0.0  ...              0.0             0.00
3               0.0            0.0  ...              0.0             0.55
4               0.0            0.0  ...              0.0             1.00
...             ...            ...  ...              ...              ...
5995           51.0           88.0  ...              0.0             0.00
5996            0.0            1.0  ...              0.0           -99.00
5997         1558.0         2479.0  ...              0.0             0.89
5998            9.0           11.0  ...              0.0             0.99
5999            6.0           19.0  ...              0.0             0.05

      perf_12_month_avg  local_bo_qty  deck_risk  oe_constraint ppap_risk  \
0                  0.76           2.0         No             No        No
1                  0.94           0.0         No             No        No
2                  0.00           0.0        Yes             No        No
3                  0.77           0.0         No             No        No
4                  1.00           0.0         No             No        No
...                 ...           ...        ...            ...       ...
5995               0.27           1.0         No             No        No
5996             -99.00           0.0         No             No        No
5997               0.71           0.0         No             No        No
5998               0.97           0.0         No             No        No
5999               0.06           0.0         No             No        No

     stop_auto_buy rev_stop went_on_backorder
0              Yes       No               Yes
1              Yes       No               Yes
2              Yes       No               Yes
3              Yes       No                No
4              Yes       No                No
```

```
# List of columns to replace "Yes" and "No" with 1 and 0
columns_to_replace = ['deck_risk', 'oe_constraint', 'ppap_risk', 'stop_auto_buy', 'rev_stop', 'potential_iss

# Create a copy of the original DataFrame
super_mod_file = modified_data.copy()

# Replace "Yes" with 1 and "No" with 0 in the specified columns
super_mod_file[columns_to_replace] = super_mod_file[columns_to_replace].replace({'Yes': 1, 'No': 0})
```

```
<ipython-input-55-7ebc93aacb91>:8: FutureWarning: Downcasting behavior in `replace` is deprecated and wi
  super_mod_file[columns_to_replace] = super_mod_file[columns_to_replace].replace({'Yes': 1, 'No': 0})
```

```
super_mod_file.head()
```

```
print(super_mod_file.head())  # Display the first few rows
print(super_mod_file.dtypes)   # Display the data types of each column
```

```
0         -2.0       8.0             0.0              38.0
1          1.0       8.0             0.0              10.0
2          0.0       2.0             0.0               5.0
3          5.0       2.0             0.0               0.0
4        685.0       8.0             0.0               0.0

   forecast_6_month  forecast_9_month  sales_1_month  sales_3_month  \
0              56.0              80.0            8.0           22.0
1              10.0              10.0            3.0            8.0
2               5.0               5.0            0.0            0.0
3               0.0               0.0            0.0            0.0
4               0.0               0.0            0.0            0.0

   sales_6_month  sales_9_month  ...  pieces_past_due  perf_6_month_avg  \
0           36.0           42.0  ...              0.0              0.74
1            9.0            9.0  ...              0.0              1.00
2            0.0            0.0  ...              0.0              0.00
3            0.0            0.0  ...              0.0              0.55
4            0.0            0.0  ...              0.0              1.00

   perf_12_month_avg  local_bo_qty  deck_risk  oe_constraint  ppap_risk  \
0               0.76           2.0          0              0          0
1               0.94           0.0          0              0          0
2               0.00           0.0          1              0          0
3               0.77           0.0          0              0          0
4               1.00           0.0          0              0          0

   stop_auto_buy  rev_stop  went_on_backorder
0              1         0                Yes
1              1         0                Yes
2              1         0                Yes
3              1         0                 No
4              1         0                 No

[5 rows x 22 columns]
national_inv         float64
lead_time            float64
in_transit_qty       float64
forecast_3_month     float64
forecast_6_month     float64
forecast_9_month     float64
sales_1_month        float64
sales_3_month        float64
sales_6_month        float64
sales_9_month        float64
min_bank             float64
potential_issue        int64
pieces_past_due      float64
perf_6_month_avg     float64
perf_12_month_avg    float64
local_bo_qty         float64
deck_risk              int64
oe_constraint          int64
ppap_risk              int64
stop_auto_buy          int64
rev_stop               int64
went_on_backorder     object
dtype: object
```

```
super_mod_file.groupby('went_on_backorder').mean()
```

```
super_mod_file.groupby('went_on_backorder').mean()
```

went_on_backorder	national_inv	lead_time	in_transit_qty	forecast_3_month	forecast_6_month	forecast_9
No	359.114000	7.664000	26.550333	140.307000	281.522667	41
Yes	10.747667	6.372333	4.258333	144.151333	218.062333	29

2 rows × 21 columns

```
import pandas as pd

# Specify the columns to check for NaN values
columns_to_check = [
    'national_inv', 'lead_time', 'in_transit_qty',
    'forecast_3_month', 'forecast_6_month', 'forecast_9_month',
    'sales_1_month', 'sales_3_month', 'sales_6_month',
    'sales_9_month', 'min_bank', 'potential_issue',
    'pieces_past_due', 'perf_6_month_avg', 'perf_12_month_avg',
    'local_bo_qty', 'deck_risk', 'oe_constraint',
    'ppap_risk', 'stop_auto_buy', 'rev_stop'
]

# Remove rows with NaN values in the specified columns
super_mod_file_cleaned = super_mod_file.dropna(subset=columns_to_check)

X = super_mod_file_cleaned.drop(columns='went_on_backorder')
Y = super_mod_file_cleaned['went_on_backorder']

X = X.dropna()
Y = Y[X.index]
```

Double-click (or enter) to edit

```
print(X)
print(Y)
```

```
5995           91.0           126.0          6.0          24.0
5996            0.0             0.0          0.0           0.0
5997          993.0          1460.0        264.0         754.0
5998            3.0             6.0          1.0           5.0
5999            0.0             0.0          0.0           2.0

      sales_6_month  sales_9_month  ...  potential_issue  pieces_past_due  \
0              36.0           42.0  ...                0              0.0
1               9.0            9.0  ...                0              0.0
2               0.0            0.0  ...                0              0.0
3               0.0            0.0  ...                0              0.0
4               0.0            0.0  ...                0              0.0
...             ...            ...  ...              ...              ...
5995           51.0           88.0  ...                0              0.0
```

```
1                0          0             1         0
2                0          0             1         0
3                0          0             1         0
4                0          0             1         0
...            ...        ...           ...       ...
5995             0          0             1         0
5996             0          0             1         0
5997             0          0             1         0
5998             0          0             1         0
5999             0          0             1         0

[6000 rows x 21 columns]
0       Yes
1       Yes
2       Yes
3        No
4        No
       ...
5995    Yes
5996     No
5997     No
5998    Yes
5999     No
Name: went_on_backorder, Length: 6000, dtype: object
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.1, stratify=Y, random_state=1)
```

```
print(X_train)
print(Y_train)
```

```
      national_inv  lead_time  in_transit_qty  forecast_3_month  \
3960          13.0       12.0             0.0               2.0
4780           6.0        2.0             0.0              40.0
157           24.0        2.0             0.0               0.0
315            3.0        8.0             0.0               4.0
3336        4565.0       12.0             0.0               0.0
...            ...        ...             ...               ...
2726        -152.0       12.0             0.0             626.0
5797           0.0        2.0             0.0               0.0
2384         374.0       12.0             0.0             320.0
5831          47.0       12.0             0.0             623.0
436            7.0        8.0             0.0               0.0

      forecast_6_month  forecast_9_month  sales_1_month  sales_3_month  \
3960               4.0               6.0            4.0            7.0
4780              40.0              40.0            0.0            0.0
157                0.0               0.0            0.0            0.0
315                9.0              14.0            1.0            4.0
3336               0.0               0.0            5.0           65.0
...                ...               ...            ...            ...
2726             791.0             901.0          153.0          299.0
5797               0.0               0.0            1.0            3.0
2384             960.0             960.0           94.0          339.0
5831            1063.0            1503.0          112.0          387.0
436                8.0              12.0            1.0            5.0

      sales_6_month  sales_9_month  ...  potential_issue  pieces_past_due  \
3960           17.0           24.0  ...                0              0.0
4780            0.0            0.0  ...                0              0.0
157             0.0            0.0  ...                0              0.0
315            10.0           14.0  ...                0              0.0
3336           99.0          107.0  ...                0              0.0
...             ...            ...  ...              ...              ...
2726          513.0          712.0  ...                0              0.0
5797            5.0            6.0  ...                0              0.0
2384          728.0         1153.0  ...                0              0.0
5831          787.0         1205.0  ...                0              0.0
436            10.0           16.0  ...                0              0.0

      perf_6_month_avg  perf_12_month_avg  local_bo_qty  deck_risk  \
3960              0.76               0.73           0.0          0
4780              0.98               0.97           0.0          1
157               0.80               0.60           0.0          1
315               0.78               0.74           0.0          0
3336              0.96               0.86           0.0          0
...                ...                ...           ...        ...
2726              0.00               0.02          69.0          0
5797              0.99               0.99           0.0          0
```

```
      2384         0.88         0.81        0.0        0
      5831         0.88         0.86        4.0        0
      436          0.89         0.78        0.0        0

            oe_constraint  ppap_risk  stop_auto_buy  rev_stop
      3960              0          0              1         0
      4780              0          0              1         0
      157               0          0              1         0
      315               0          0              1         0
      3336              0          0              1         0
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
model = LogisticRegression(solver='lbfgs', max_iter=1000)
```

```
model.fit(X_train_scaled, Y_train)
```

```
▾ LogisticRegression
LogisticRegression(max_iter=1000)
```

Accuracy on Training Data

```
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
print('Accuracy on training data : ', training_data_accuracy)
```

```
Accuracy on training data :  0.7409259259259259
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2732: UserWarning: X has feature nam
  warnings.warn(
```

```
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print('Accuracy on test data : ', test_data_accuracy)
```

```
Accuracy on test data :  0.7366666666666667
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2732: UserWarning: X has feature nam
  warnings.warn(
```

Test on Random Data from the Dataset

```
# Original input data
input_data = [-1, 0, 0, 26, 36, 51, 8, 18, 42, 52, 1, 'No', 0, 0, 0, 1, 'No', 'No', 'No', 'No', 'No'
]
# Replace 'No' with 0 and 'Yes' with 1
mod_input_data = [0 if x == 'No' else 1 if x == 'Yes' else x for x in input_data]

mod_input_data_as_numpy_array = np.asarray(mod_input_data)

reshaped_mod_input_data = mod_input_data_as_numpy_array.reshape(1, -1)

predict = model.predict(reshaped_mod_input_data)

print(predict)

if predict == 'Yes':
  print('There will be a shortage')
else:
  print('There will not be a shortage')
```

```
['Yes']
There will be a shortage
```

Start coding or generate with AI.