In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re

from sklearn.model_selection import train_test_split
from sklearn.metrics import (confusion_matrix,
                             ConfusionMatrixDisplay,
                             classification_report)
from sklearn.preprocessing import LabelEncoder,OrdinalEncoder
from sklearn.impute import SimpleImputer
from imblearn.over_sampling import SMOTE

from xgboost import XGBClassifier

import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
# df_original = pd.read_excel('train.xlsx')
df_original = pd.read_excel('Test.xlsx')
```

In [3]:

```
df = df_original.copy()
```

In [4]:

```
df.columns
```

Out[4]:

```
Index(['Item Number', 'Status', 'Description1', 'Description2', 'Full Desc1',
       'Full Desc2', 'Full Desc3', 'Full Desc4',
       'Logical Concate of Discription1 & 2', 'Item Full Description Final',
       'UM', 'Prod Line', 'Design Group', 'Promotion', 'Product Group', 'Type',
       'Trade Class', 'Art Nbr (Sch B)', 'Commodity Code', 'Supplier',
       'Purchase Lead Time', 'Manufacturing Lead Time', 'Lead Time',
       'Revision', 'Net Weight', 'Net Weight UM', 'UserID', 'Added',
       'Modified Date', 'Label from SKU Hierarchy3',
       '2021-2024 Invoice history data', 'Unnamed: 31', 'Unnamed: 32',
       'Unnamed: 33', 'Unnamed: 34', 'Unnamed: 35', 'Unnamed: 36',
       'Unnamed: 37', 'Unnamed: 38'],
      dtype='object')
```

In [5]:

```
# check for nan/null values
df.isna().sum()
```

```
Out[5]: Item Number                               22
        Status                                     0
        Description1                               7
        Description2                           58625
        Full Desc1                               888
        Full Desc2                              8280
        Full Desc3                             33624
        Full Desc4                             64813
        Logical Concate of Discription1 & 2        7
        Item Full Description Final               22
        UM                                         0
        Prod Line                                  0
        Design Group                             302
        Promotion                              56323
        Product Group                              0
        Type                                       0
        Trade Class                            56413
        Art Nbr (Sch B)                        56412
        Commodity Code                         71159
        Supplier                               11016
        Purchase Lead Time                         0
        Manufacturing Lead Time                    0
        Lead Time                                  0
        Revision                               73060
        Net Weight                                 0
        Net Weight UM                              1
        UserID                                     0
        Added                                    108
        Modified Date                              0
        Label from SKU Hierarchy3              26758
        2021-2024 Invoice history data             6
        Unnamed: 31                            80184
        Unnamed: 32                            80184
        Unnamed: 33                            80184
        Unnamed: 34                            80181
        Unnamed: 35                            80182
        Unnamed: 36                            80182
        Unnamed: 37                            80182
        Unnamed: 38                            80182
        dtype: int64
```

In [6]:

```
# select features with predictive values
df_2 = df[['Item Full Description Final','UM',
            'Prod Line','Product Group','Type','Net Weight',
           'Label from SKU Hierarchy3'
          ]].copy()
```

In [7]:

```
# rename long column names
df_2.rename({"Label from SKU Hierarchy3":'labels'},axis=1,inplace=True)
df_2.rename({'Item Full Description Final':'desc'},axis=1,inplace=True)
```

In [8]:

```
# check for null values in the selected features
df_2.isna().sum()
```

Out[8]:

```
desc                22
UM                   0
Prod Line            0
Product Group        0
Type                 0
Net Weight           0
labels           26758
dtype: int64
```

In [9]:

```
# check for number of unique values in each feature
df_2.nunique()
```

Out[9]:

```
desc             80153
UM                  22
Prod Line           13
Product Group       11
Type               400
Net Weight        2448
labels              47
dtype: int64
```

In [10]:

```
# helper function to clean up unnecessary elements in the desc feature
def clean_string(string):
    data = re.sub('[\d+-\."#)(]|\..*\.|\sid\s','',string).strip()
    data = re.sub('/+','',data).strip()
    data = re.sub('\sx\s','x',data).strip()
    data = re.sub('\s+',' ',data).strip()
    data = re.sub('\s[a-z]\s|^[a-z]\s|\s[a-z]$','',data).strip()
    return data
```

In [11]:

```
# helper function to standardize the desc feature a little bit
def filter(x):
    key_words = ['pin','streamer warning','accelerometer',
                 'anchor','antenna','bearing','bolt','seal',
                 'stud','screw',
                 'nut','delta','pt pan',
                #  'insert ctsk','insert helical','helicoil insert','insert ultem',
                #  'lg insert','thrd insert',
                 'insert',
                 'hex bolt', 'pt bolt','ph bolt','ft bolt',
                 'hd flat bolt','tap bolt','hd flat','tap','bracket',
                 'bushing','cable ties','cable tray',
                 'cable','hilok','collar','weld',
                 'conduit','connector','shrink',
                 'magnet', 'magnetic','resistor','inductor', 'capacitor',
                 'oscillator','circuit board','ring','rivet',
                 'insuliner',
                 'diode','transistor','shim','spacer','spring',
                 'splitter','o-ring','rubber fluorocarbon',
                 'weather','washer','fw','lw','clamp',
                 'industrial','graphics','kitting','gasket','packing',
                 'standoff','spacer','cable management','tool','bracket','reinforcement',
                 'brace','passive','circuit','bushing','adhesive','glue','bearing','tape',
                 'panel','switch','ring','retaining','heat','terminal','grommet',
                 'semiconductor','decal','bulb',
                 'recertified','wire','f/w','hd ', 'soc ', 'alloy steel',
                  'head soc','anco steel','hex c/s','helical','scw','hex l/n','int l/w',
                  'blind fastener','optocoupler','hilock','head c/s','velocity sensor',
                 'ansi/asme','wash bt','washer','pipe', 'valve', 'fitting','support',
                  'fillister','insuliner',
                'alloy plain','znc/yel','din','slot',' insulated','cushion',
                 'leveler','clip','circuit breaker','switch relay',
                 'locking','surcharge','filter','machine plug','plug',
                 'flat','round','pan',
                 'machine','circuit breaker','crimp',
                 'turret',
                 'sensor','rack','control','junction box',
                 'elbow','exchanger','printed','chain',
                 "tube",'nameplate','lubricating',
                 'drill','shank','surface','semicunductor',
                 'kit','magnet','flang',
                 # 'thread','stanless','revanusxap','revmusxap',
                #  'np','ap','abxab','wp','dpxdp','ubtxpxubtxp', "xfbbxxxfbbx", 'ndpxndp',
                #  'apxap','wpxwp','fflmsphsspa','xohbphspa','cthmsphstplphil',
                #  'cphmsslstpl','mpmsmxzcx','nbpbxnbpb','bxwp','nfpbxnfpb','hpxhp','bpxbp',
                #  "xcbbxxxcbbx",'revgxap','altcvxaltcv',
                ]

    for word in key_words:
        if x.find(word) > -1:
            return word
    x = x.split(' ')
    return x[0].strip()
```

In [12]:

```
# setting all categorical features to lower case
for col in df_2.select_dtypes('object').columns:
        df_2[col] = df_2[col].apply(lambda x: str(x).strip().lower())
```

In [13]:

```
# applying the helper functions
df_2['desc'] = df_2['desc'].apply(clean_string)
df_2['desc_2'] = df_2['desc'].apply(filter)
```

In [14]:

```
df_2
```

Out[14]:

	desc	UM	Prod Line	Product Group	Type	Net Weight	labels	desc_2
0	mxmm torxpan ms steel driloc reachrohs per prt...	ea	hard	custom	9999	0.0	screws	pan
1	ctdudunbpl trnczcxendend full thread stud astm...	ea	hard	generic	9999	0.0	nan	stud
2	xgpasspa gpcnn stain groove pin ty	ea	hard	generic	9999	0.0	pins	pin
3	soc set screw cup pt alloy plain nylon patch p...	ea	hard	custom	9999	0.0	screws	screw
4	ap gr hex hd cs xflat to flat plain rev	ea	hard	custom	9999	0.0	screws	hd
...	...	...	...	...	...	...	...	...
80179	ep lg seal cloth per per	ea	hard	custom	4900	0.0	nan	seal
80180	hpmvdc fuse midget	ea	hard	proprty	3025	0.0	nan	hpmvdc
80181	chxntjmstun hex jam nut steel plain asme	ea	hard	generic	4404	0.0	nan	nut
80182	gawire pvc awm &	ea	hard	proprty	8000	0.0	nan	wire
80183	hinge external assy ss per prt	ea	hard	custom	3038	0.0	nan	hinge

80184 rows × 8 columns

In [15]:

```
# checking the value counts of our target variable
df_2.labels.value_counts()
```

```
Out[15]:  nan                                    26758
          screws                                 18818
          washers                                 5903
          nuts                                    4821
          bolts                                   3625
          pins                                    2278
          clamps                                  1367
          rivets                                  1271
          seals                                   1213
          industrial graphics                     1079
          kitting                                  990
          gaskets                                  939
          cables                                   908
          inserts                                  849
          o-rings                                  769
          packing                                  663
          springs                                  630
          shims                                    621
          stud                                     604
          spacers & standoffs                      567
          cable management                         519
          tools                                    442
          brackets & reinforcement braces          374
          passive circuit components               361
          bushings                                 337
          adhesive & glue                          320
          bearings                                 280
          electronics component connectors         278
          tape                                     271
          collars                                  271
          electrical conduit                       245
          teflon o-rings                           212
          circuit breaker panels                   204
          electrical switches                      201
          retaining rings                          193
          heat-shrink tubing                       174
          wire terminals & connectors              128
          pipe, valve, fittings and support        126
          grommets                                 115
          semiconductors                            99
          labels and decals                         82
          accessories                               67
```

```
magnets                                    63
anchors                                    56
fitting                                    36
lightbulbs                                 30
antennas                                   19
printed circuit boards                      8
Name: labels, dtype: int64
```

In [16]:

```
# dropping all classes with less than 8 samples
drop = df_2[df_2.labels.isin(['miscellaneous','weather stripping','splitters',
                       'flagging & caution tape','caulks & sealants',
                       'carbon brushes','networking','machined parts & fabrications'
                      ])].index
df_2.drop(drop,axis=0,inplace=True)
```

In [17]:

```
# dropping all classes with less than 8 samples
df.drop(drop,axis=0,inplace=True)
```

In [18]:

```
# confirming if they were dropped
df_2[df_2.labels.isin(['miscellaneous','weather stripping','splitters',
                       'flagging & caution tape','caulks & sealants',
                       'carbon brushes','networking','machined parts & fabrications'
                      ])]
```

Out[18]:

	desc	UM	Prod Line	Product Group	Type	Net Weight	labels	desc_2

In [19]:

```
df_2.nunique()
```

Out[19]:

```
desc              58784
UM                   22
Prod Line            11
Product Group        11
Type                400
Net Weight         2448
labels               48
desc_2             5949
dtype: int64
```

In [20]:

```
df_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 80184 entries, 0 to 80183
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   desc           80184 non-null  object 
 1   UM             80184 non-null  object 
 2   Prod Line      80184 non-null  object 
 3   Product Group  80184 non-null  object 
 4   Type           80184 non-null  object 
 5   Net Weight     80184 non-null  float64
 6   labels         80184 non-null  object 
 7   desc_2         80184 non-null  object 
dtypes: float64(1), object(7)
memory usage: 4.9+ MB
```

In [21]:

```
# dropping desc feature as it is no longer needed
df_2.pop('desc')
```

Out[21]:

```
0        mxmm torxpan ms steel driloc reachrohs per prt...
1        ctdudunbpl trnczcxendend full thread stud astm...
2                       xgpasspa gpcnn stain groove pin ty
3        soc set screw cup pt alloy plain nylon patch p...
4                  ap gr hex hd cs xflat to flat plain rev
                               ...                        
80179                             ep lg seal cloth per per
80180                                   hpmvdc fuse midget
80181             chxntjmstun hex jam nut steel plain asme
80182                                     gawire pvc awm &
80183                       hinge external assy ss per prt
Name: desc, Length: 80184, dtype: object
```

In [22]:

```
df_2
```

Out[22]:

	UM	Prod Line	Product Group	Type	Net Weight	labels	desc_2
0	ea	hard	custom	9999	0.0	screws	pan
1	ea	hard	generic	9999	0.0	nan	stud
2	ea	hard	generic	9999	0.0	pins	pin
3	ea	hard	custom	9999	0.0	screws	screw
4	ea	hard	custom	9999	0.0	screws	hd
...	...	...	...	...	...	...	...
80179	ea	hard	custom	4900	0.0	nan	seal
80180	ea	hard	proprty	3025	0.0	nan	hpmvdc
80181	ea	hard	generic	4404	0.0	nan	nut
80182	ea	hard	proprty	8000	0.0	nan	wire
80183	ea	hard	custom	3038	0.0	nan	hinge

80184 rows × 7 columns

In [22]:

```

```

In [23]:

```
# separating missing labels values from the labelled
unfilled = df_2[df_2.labels == 'nan'].copy()
filled = df_2.drop(df_2[df_2['labels']  == 'nan'].index,axis=0)
```

In [24]:

```
# preparing encodings for our labels and categorical features
label = LabelEncoder()
label.fit(filled['labels'])

um_encoder = OrdinalEncoder()
um_encoder.fit(df_2['UM'].values.reshape(-1, 1))

prod_line_encoder = OrdinalEncoder()
prod_line_encoder.fit(df_2['Prod Line'].values.reshape(-1, 1))
```

```
product_group_encoder = OrdinalEncoder()
product_group_encoder.fit(df_2['Product Group'].values.reshape(-1, 1))

type_encoder = OrdinalEncoder()
type_encoder.fit(df_2['Type'].values.reshape(-1, 1))

desc_2_encoder = OrdinalEncoder()
desc_2_encoder.fit(df_2['desc_2'].values.reshape(-1, 1))
```

Out[24]:

```
▾ OrdinalEncoder

OrdinalEncoder()
```

In [25]:

```
# let's view our dataset
filled
```

Out[25]:

	UM	Prod Line	Product Group	Type	Net Weight	labels	desc_2
0	ea	hard	custom	9999	0.0	screws	pan
2	ea	hard	generic	9999	0.0	pins	pin
3	ea	hard	custom	9999	0.0	screws	screw
4	ea	hard	custom	9999	0.0	screws	hd
5	ea	hard	custom	9999	0.0	bolts	bolt
...	...	...	...	...	...	...	...
80166	ea	hard	xpd2	xpd2	0.0	screws	soc
80167	ea	hard	custom	3038	0.0	kitting	kit
80169	ea	hard	proprty	3020	0.0	collars	bolt
80176	ea	hard	proprty	7002	0.0	screws	screw
80177	ft	hard	proprty	8000	0.0	cables	wire

53426 rows × 7 columns

In [26]:

```python
# encoding our labels and categorical features
filled['labels'] = label.transform(filled['labels'])

filled['UM'] = um_encoder.transform(filled['UM'].values.reshape(-1, 1))
df_2['UM'] = um_encoder.transform(df_2['UM'].values.reshape(-1, 1))

filled['Prod Line'] = prod_line_encoder.transform(filled['Prod Line'].values.reshape(-1, 1))
df_2['Prod Line'] = prod_line_encoder.transform(df_2['Prod Line'].values.reshape(-1, 1))

filled['Product Group'] = product_group_encoder.transform(filled['Product Group'].values.reshape(-1, 1))
df_2['Product Group'] = product_group_encoder.transform(df_2['Product Group'].values.reshape(-1, 1))

filled['Type'] = type_encoder.transform(filled['Type'].values.reshape(-1, 1))
df_2['Type'] = type_encoder.transform(df_2['Type'].values.reshape(-1, 1))

filled['desc_2'] = desc_2_encoder.transform(filled['desc_2'].values.reshape(-1, 1))
df_2['desc_2'] = desc_2_encoder.transform(df_2['desc_2'].values.reshape(-1, 1))
```

In [27]:

```python
# checking the datatypes
filled.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 53426 entries, 0 to 80177
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   UM             53426 non-null  float64
 1   Prod Line      53426 non-null  float64
 2   Product Group  53426 non-null  float64
 3   Type           53426 non-null  float64
 4   Net Weight     53426 non-null  float64
 5   labels         53426 non-null  int64  
 6   desc_2         53426 non-null  float64
dtypes: float64(6), int64(1)
memory usage: 3.3 MB
```

In [28]:

```python
df_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 80184 entries, 0 to 80183
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   UM             80184 non-null  float64
 1   Prod Line      80184 non-null  float64
 2   Product Group  80184 non-null  float64
 3   Type           80184 non-null  float64
 4   Net Weight     80184 non-null  float64
 5   labels         80184 non-null  object 
 6   desc_2         80184 non-null  float64
dtypes: float64(6), object(1)
memory usage: 4.3+ MB
```

In [29]:

```
# dropping the label in the general dataset
df_2.pop('labels')
```

Out[29]:

```
0        screws
1           nan
2          pins
3        screws
4        screws
          ...  
80179       nan
80180       nan
80181       nan
80182       nan
80183       nan
Name: labels, Length: 80184, dtype: object
```

In [30]:

```
# df_2.pop('UM')
# filled.pop('UM')

# df_2.pop('Prod Line')
# filled.pop('Prod Line')

# df_2.pop('Net Weight')
# filled.pop('Net Weight')
```

In [31]:

```
# creating our train/test split
strat_train_set, strat_test_set = train_test_split(
    filled, test_size=0.2, stratify=filled["labels"], random_state=42)

train_features = strat_train_set.drop('labels',axis=1)
train_label = strat_train_set.labels

test_features = strat_test_set.drop('labels',axis=1)
test_label = strat_test_set.labels
```

In [32]:

```
# smote for imbalance data
smt = SMOTE()
```

In [33]:

```
# balancing our data
# train_features,train_label = smt.fit_resample(train_features,train_label)
```

In [34]:

```
# initializing our model
model = XGBClassifier()
```

In [35]:

```
for num,name in enumerate(label.classes_):
  print(f'{num} --> {name}')
```

```
0 --> accessories
1 --> adhesive & glue
2 --> anchors
3 --> antennas
4 --> bearings
5 --> bolts
6 --> brackets & reinforcement braces
7 --> bushings
8 --> cable management
9 --> cables
10 --> circuit breaker panels
11 --> clamps
12 --> collars
13 --> electrical conduit
14 --> electrical switches
15 --> electronics component connectors
16 --> fitting
17 --> gaskets
18 --> grommets
19 --> heat-shrink tubing
20 --> industrial graphics
21 --> inserts
22 --> kitting
23 --> labels and decals
24 --> lightbulbs
25 --> magnets
26 --> nuts
27 --> o-rings
28 --> packing
29 --> passive circuit components
30 --> pins
31 --> pipe, valve, fittings and support
32 --> printed circuit boards
33 --> retaining rings
34 --> rivets
35 --> screws
36 --> seals
37 --> semiconductors
38 --> shims
39 --> spacers & standoffs
40 --> springs
41 --> stud
```

```
42 --> tape
43 --> teflon o-rings
44 --> tools
45 --> washers
46 --> wire terminals & connectors
```

In [36]:

```
# training our model
model.fit(train_features,train_label)
```

Out[36]:

```
▾                                XGBClassifier

XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
```

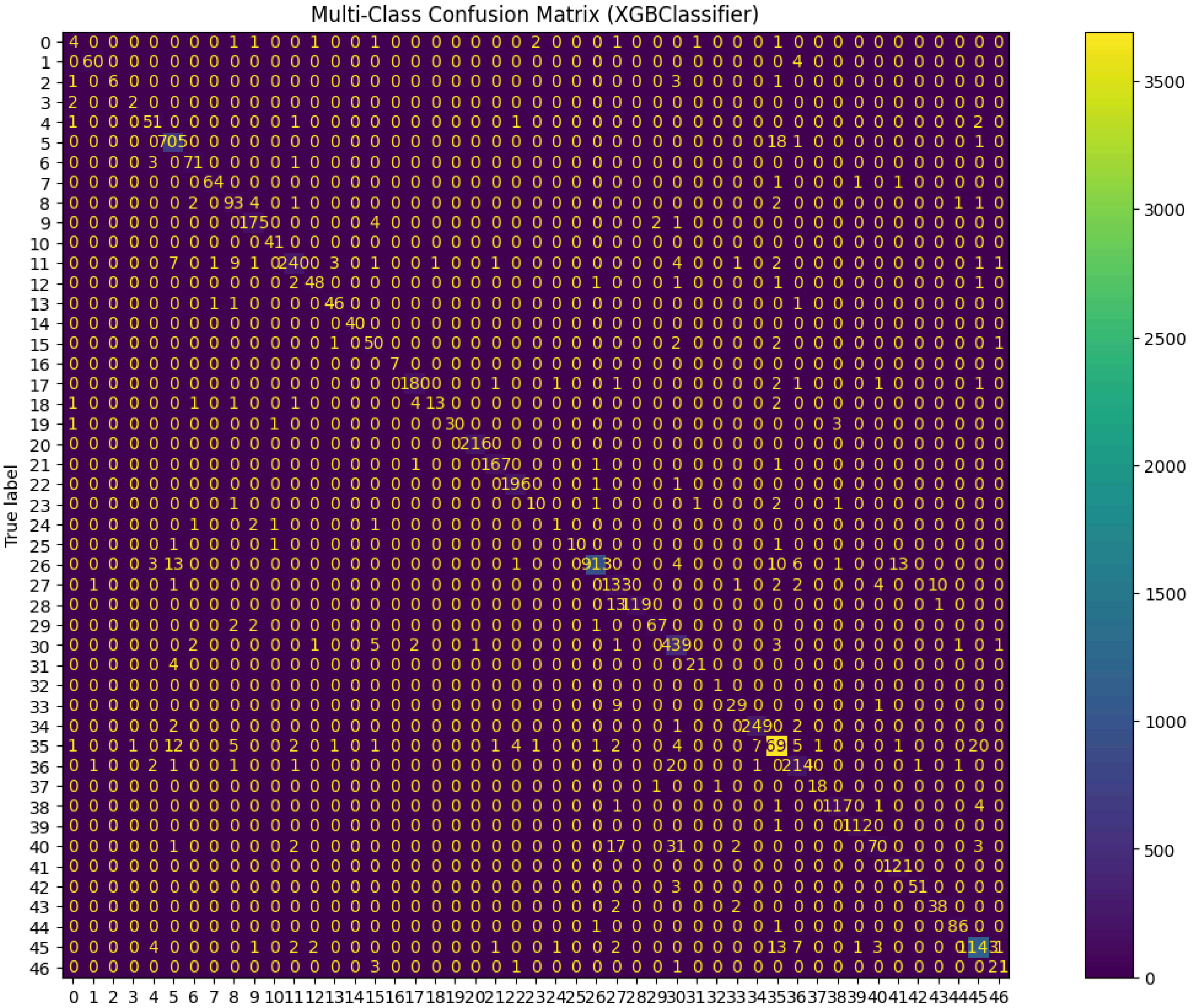
In [37]:

```
# making predictions
predictions = model.predict(test_features)
```

In [38]:

```
# let's plot a confusion matrix
disp = ConfusionMatrixDisplay.from_predictions(test_label,predictions)

fig = disp.figure_
fig.set_figwidth(16)
fig.set_figheight(10)
plt.title('Multi-Class Confusion Matrix (XGBClassifier)');
```


Multi-Class Confusion Matrix (XGBClassifier)
True label
0 4 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 0 0 0 0 2 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
1 0 60 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0
2 1 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
3 2 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 1 0 0 0 51 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0
5 0 0 0 0 0 705 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 18 1 0 0 0 0 0 0 0 0 1 0
6 0 0 0 0 3 0 71 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
7 0 0 0 0 0 0 0 64 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0
8 0 0 0 0 0 0 2 0 93 4 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 1 1 0
9 0 0 0 0 0 0 0 0 0 175 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10 0 0 0 0 0 0 0 0 0 0 41 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
11 0 0 0 0 0 7 0 1 9 1 0 240 0 3 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 4 0 0 1 0 2 0 0 0 0 0 0 0 0 0 1 1
12 0 0 0 0 0 0 0 0 0 0 0 2 48 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0
13 0 0 0 0 0 0 0 1 1 0 0 0 0 46 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
14 0 0 0 0 0 0 0 0 0 0 0 0 0 0 40 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
15 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 50 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 1
16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
17 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 180 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 2 1 0 0 0 1 0 0 0 0 1 0
18 1 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 4 13 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0
19 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 30 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0
20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 216 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
21 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 167 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 196 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
23 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 0 0 1 0 0 0 0 1 0 0 0 2 0 0 1 0 0 0 0 0 0 0 0
24 0 0 0 0 0 0 1 0 0 2 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
25 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
26 0 0 0 0 3 13 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 913 0 0 0 4 0 0 0 0 10 6 0 1 0 0 0 13 0 0 0 0
27 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 133 0 0 0 0 0 1 0 2 2 0 0 4 0 0 10 0 0 0 0
28 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 13 119 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
29 0 0 0 0 0 0 0 0 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 67 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
30 0 0 0 0 0 0 2 0 0 0 0 0 1 0 0 5 0 2 0 0 1 0 0 0 0 0 0 1 0 0 439 0 0 0 0 3 0 0 0 0 0 0 0 0 1 0 1
31 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
32 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
33 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 29 0 0 0 0 0 0 1 0 0 0 0 0 0
34 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 249 0 2 0 0 0 0 0 0 0 0 0 0
35 1 0 0 1 0 12 0 0 5 0 0 2 0 1 0 1 0 0 0 0 0 1 4 1 0 0 1 2 0 0 4 0 0 0 7 69 5 1 0 0 1 0 0 0 0 20 0
36 0 1 0 0 2 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 20 0 0 0 1 0 214 0 0 0 0 0 1 0 1 0 0
37 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 18 0 0 0 0 0 0 0 0 0
38 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 117 0 1 0 0 0 0 4 0
39 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 112 0 0 0 0 0 0 0
40 0 0 0 0 0 1 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17 0 0 31 0 0 2 0 0 0 0 0 0 70 0 0 0 0 3 0
41 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 121 0 0 0 0 0
42 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 51 0 0 0 0
43 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 38 0 0 0
44 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 86 0 0
45 0 0 0 0 4 0 0 0 0 1 0 2 2 0 0 0 0 0 0 0 0 1 0 0 1 0 0 2 0 0 0 0 0 0 0 13 7 0 0 1 3 0 0 0 0 1143 1
46 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 21
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
3500
3000
2500
2000
1500
1000
500
0

Predicted label

In [39]:

```
# performance stats
print(classification_report(test_label,predictions))
```

```
              precision    recall  f1-score   support

           0       0.36      0.31      0.33        13
           1       0.97      0.94      0.95        64
           2       1.00      0.55      0.71        11
           3       0.67      0.50      0.57         4
           4       0.81      0.91      0.86        56
           5       0.94      0.97      0.96       725
           6       0.92      0.95      0.93        75
           7       0.97      0.96      0.96        67
           8       0.82      0.89      0.85       104
           9       0.94      0.96      0.95       182
          10       0.93      1.00      0.96        41
          11       0.95      0.88      0.91       273
          12       0.92      0.89      0.91        54
          13       0.90      0.94      0.92        49
          14       1.00      1.00      1.00        40
          15       0.76      0.89      0.82        56
          16       1.00      1.00      1.00         7
          17       0.96      0.96      0.96       188
          18       0.93      0.57      0.70        23
          19       1.00      0.86      0.92        35
          20       1.00      1.00      1.00       216
          21       0.98      0.98      0.98       170
          22       0.97      0.99      0.98       198
          23       0.77      0.62      0.69        16
          24       0.33      0.17      0.22         6
          25       1.00      0.77      0.87        13
          26       0.99      0.95      0.97       964
          27       0.73      0.86      0.79       154
          28       1.00      0.89      0.94       133
          29       0.96      0.93      0.94        72
          30       0.85      0.96      0.90       456
          31       0.91      0.84      0.87        25
          32       0.50      1.00      0.67         1
          33       0.83      0.74      0.78        39
          34       0.97      0.98      0.97       254
          35       0.98      0.98      0.98      3764
          36       0.88      0.88      0.88       243
          37       0.95      0.90      0.92        20
          38       0.96      0.94      0.95       124
          39       0.98      0.99      0.99       113
          40       0.88      0.56      0.68       126
          41       0.89      1.00      0.94       121
          42       0.98      0.94      0.96        54
          43       0.78      0.90      0.84        42
          44       0.97      0.98      0.97        88
          45       0.97      0.97      0.97      1181
          46       0.84      0.81      0.82        26

    accuracy                           0.95     10686
   macro avg       0.88      0.86      0.87     10686
weighted avg       0.95      0.95      0.95     10686
```

In [40]:

```
# select all the labelled data
train_features = filled.drop('labels',axis=1)
train_label = filled.labels
```

In [41]:

```
# train with full labelled data
model.fit(train_features,train_label)
```

Out[41]:

```
▾                         XGBClassifier

XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
```

In [42]:

```
# predict with the general/full dataset
total_prediction = model.predict(df_2)
```

In [43]:

```
# copy from the original/virgin dataset
all_data = df_original.loc[df_2.index].copy()
```

In [44]:

```
# add the predictions
all_data['predictions'] = label.inverse_transform(total_prediction)
```

In [45]:

```
# select just few columns for easy assessment
all_data = all_data[['Item Number','Item Full Description Final','UM',
            'Prod Line', 'Type','Net Weight',
           'Label from SKU Hierarchy3','predictions']]
```

In [46]:

```
# save the dataset with the predictions
all_data.to_excel('Final_prediction_updated.xlsx')
```