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]:

```
# mount google drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call driv
e.mount("/content/drive", force_remount=True).
```

In [3]:

```
# df_original = pd.read_excel('train.xlsx')
df_original = pd.read_excel('/content/drive/MyDrive/train.xlsx')
```

In [4]:

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

In [5]:

```
df.columns
```

Out[5]:

```
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 [6]:

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

Out[6]:

```
Item Number                                0
Status                                     0
Description1                               7
Description2                           58634
Full Desc1                               888
Full Desc2                              8283
Full Desc3                             33631
Full Desc4                             64822
Logical Concate of Discription1 & 2        7
Item Full Description Final                0
UM                                         0
Prod Line                                  0
Design Group                             302
Promotion                              56332
Product Group                              0
Type                                       0
Trade Class                            56421
Art Nbr (Sch B)                        56420
Commodity Code                         71167
Supplier                               11018
Purchase Lead Time                         0
Manufacturing Lead Time                    0
Lead Time                                  0
Revision                               73069
Net Weight                                 0
Net Weight UM                              1
UserID                                     0
Added                                    108
Modified Date                              0
Label from SKU Hierarchy3              26736
2021-2024 Invoice history data             0
Unnamed: 31                            80193
Unnamed: 32                            80193
Unnamed: 33                            80193
Unnamed: 34                            80190
Unnamed: 35                            80191
Unnamed: 36                            80191
Unnamed: 37                            80191
Unnamed: 38                            80191
dtype: int64
```

In [7]:

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

In [8]:

```
# 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 [9]:

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

```
Out[9]: desc                0
        Product Group       0
        Type                0
        Net Weight          0
        labels          26736
        dtype: int64
```

```
In [10]: # check for number of unique values in each feature
         df_2.nunique()
```

```
Out[10]: desc            80184
         Product Group      11
         Type              400
         Net Weight       2448
         labels             55
         dtype: int64
```

```
In [11]: # 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 [12]:

```
# 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','washer',
                 'nut','delta','pt pan',
                 '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','clamp','fw','lw',
                 'industrial','graphics','kitting','gasket','packing',
                 'standoff','spacer','cable management','tool','bracket','reinforce
                 'brace','passive','circuit','bushing','adhesive','glue','bearing',
                 '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
                  'blind fastener','optocoupler','hilock','head c/s','velocity sens
                 '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', 'fleng'
                ]

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

In [13]:

```
# 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 [14]:

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

In [15]:

```
# feature engineering
df_2['Type_2'] = df_2['Type'] + '-' + df_2['desc_2']
```

In [16]:

```
df_2
```

Out[16]:

	desc	Product Group	Type	Net Weight	labels	desc_2	Type_2
0	mxmm torxpan ms steel driloc reachrohs per prt...	custom	9999	0.0	screws	pan	9999-pan
1	ctdudunbpl trnczcxendend full thread stud astm...	generic	9999	0.0	nan	stud	9999-stud
2	xgpasspa gpcnn stain groove pin ty	generic	9999	0.0	pins	pin	9999-pin
3	soc set screw cup pt alloy plain nylon patch p...	custom	9999	0.0	screws	screw	9999-screw
4	ap gr hex hd cs xflat to flat plain rev	custom	9999	0.0	screws	hd	9999-hd
...	...	...	...	...	...	...	...
80188	ep lg seal cloth per per	custom	4900	0.0	nan	seal	4900-seal
80189	hpmvdc fuse midget	proprty	3025	0.0	nan	hpmvdc	3025-hpmvdc
80190	chxntjmstun hex jam nut steel plain asme	generic	4404	0.0	nan	nut	4404-nut
80191	gawire pvc awm &	proprty	8000	0.0	nan	wire	8000-wire
80192	hinge external assy ss per prt	custom	3038	0.0	nan	hinge	3038-hinge

80193 rows × 7 columns

In [17]:

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

```
Out[17]: nan                                  26736
         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
         miscellaneous                            7
         weather stripping                        6
         splitters                                6
         flagging & caution tape                  4
         caulks & sealants                        4
         carbon brushes                           2
         networking                               1
         machined parts & fabrications            1
```

```
Name: labels, dtype: int64
```

In [18]:
```
# 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 [19]:
```
# dropping all classes with less than 8 samples
df.drop(drop,axis=0,inplace=True)
```

In [20]:
```
# 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[20]:

desc	Product Group	Type	Net Weight	labels	desc_2	Type_2

In [21]:
```
df_2.nunique()
```

Out[21]:
```
desc              58783
Product Group        11
Type                400
Net Weight         2448
labels               48
desc_2             6052
Type_2            11463
dtype: int64
```

In [22]:
```
df_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 80162 entries, 0 to 80192
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   desc           80162 non-null  object 
 1   Product Group  80162 non-null  object 
 2   Type           80162 non-null  object 
 3   Net Weight     80162 non-null  float64
 4   labels         80162 non-null  object 
 5   desc_2         80162 non-null  object 
 6   Type_2         80162 non-null  object 
dtypes: float64(1), object(6)
memory usage: 4.9+ MB
```

In [23]:
```
# dropping desc feature as it is no longer needed
df_2.pop('desc')

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

Out[23]:

```
0        9999
1        9999
2        9999
3        9999
4        9999
         ...
80188    4900
80189    3025
80190    4404
80191    8000
80192    3038
Name: Type, Length: 80162, dtype: object
```

In [24]:

```
df_2
```

Out[24]:

	Product Group	Net Weight	labels	desc_2	Type_2
0	custom	0.0	screws	pan	9999-pan
1	generic	0.0	nan	stud	9999-stud
2	generic	0.0	pins	pin	9999-pin
3	custom	0.0	screws	screw	9999-screw
4	custom	0.0	screws	hd	9999-hd
...	...	...	...	...	...
80188	custom	0.0	nan	seal	4900-seal
80189	proprty	0.0	nan	hpmvdc	3025-hpmvdc
80190	generic	0.0	nan	nut	4404-nut
80191	proprty	0.0	nan	wire	8000-wire
80192	custom	0.0	nan	hinge	3038-hinge

80162 rows × 5 columns

In [25]:

```
# create df_2 copy before encoding
df_2_copy = df_2.copy()
```

In [26]:

```
# 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 [27]:

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

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

type_2_encoder = OrdinalEncoder()
type_2_encoder.fit(df_2['Type_2'].values.reshape(-1, 1))

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

Out[27]:

```
▾ OrdinalEncoder
OrdinalEncoder()
```

In [28]:

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

Out[28]:

	Product Group	Net Weight	labels	desc_2	Type_2
0	custom	0.0	screws	pan	9999-pan
2	generic	0.0	pins	pin	9999-pin
3	custom	0.0	screws	screw	9999-screw
4	custom	0.0	screws	hd	9999-hd
5	custom	0.0	bolts	bolt	9999-bolt
...	...	...	...	...	...
80175	xpd2	0.0	screws	soc	xpd2-soc
80176	custom	0.0	kitting	kit	3038-kit
80178	proprty	0.0	collars	bolt	3020-bolt
80185	proprty	0.0	screws	screw	7002-screw
80186	proprty	0.0	cables	wire	8000-wire

53426 rows × 5 columns

In [29]:

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

filled['Product Group'] = product_group_encoder.transform(filled['Product Group'].v
df_2['Product Group'] = product_group_encoder.transform(df_2['Product Group'].value

filled['Type_2'] = type_2_encoder.transform(filled['Type_2'].values.reshape(-1, 1))
df_2['Type_2'] = type_2_encoder.transform(df_2['Type_2'].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 [30]:

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 53426 entries, 0 to 80186
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Product Group  53426 non-null  float64
 1   Net Weight     53426 non-null  float64
 2   labels         53426 non-null  int64  
 3   desc_2         53426 non-null  float64
 4   Type_2         53426 non-null  float64
dtypes: float64(4), int64(1)
memory usage: 2.4 MB
```

In [31]:

```
df_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 80162 entries, 0 to 80192
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Product Group  80162 non-null  float64
 1   Net Weight     80162 non-null  float64
 2   labels         80162 non-null  object 
 3   desc_2         80162 non-null  float64
 4   Type_2         80162 non-null  float64
dtypes: float64(4), object(1)
memory usage: 5.7+ MB
```

In [32]:

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

Out[32]:

```
0        screws
1           nan
2          pins
3        screws
4        screws
          ...  
80188       nan
80189       nan
80190       nan
80191       nan
80192       nan
Name: labels, Length: 80162, dtype: object
```

In [33]:

```
# 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 [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=No
ne,
              enable_categorical=False, eval_metric=None, feature_types=No
ne,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=No
ne,
```

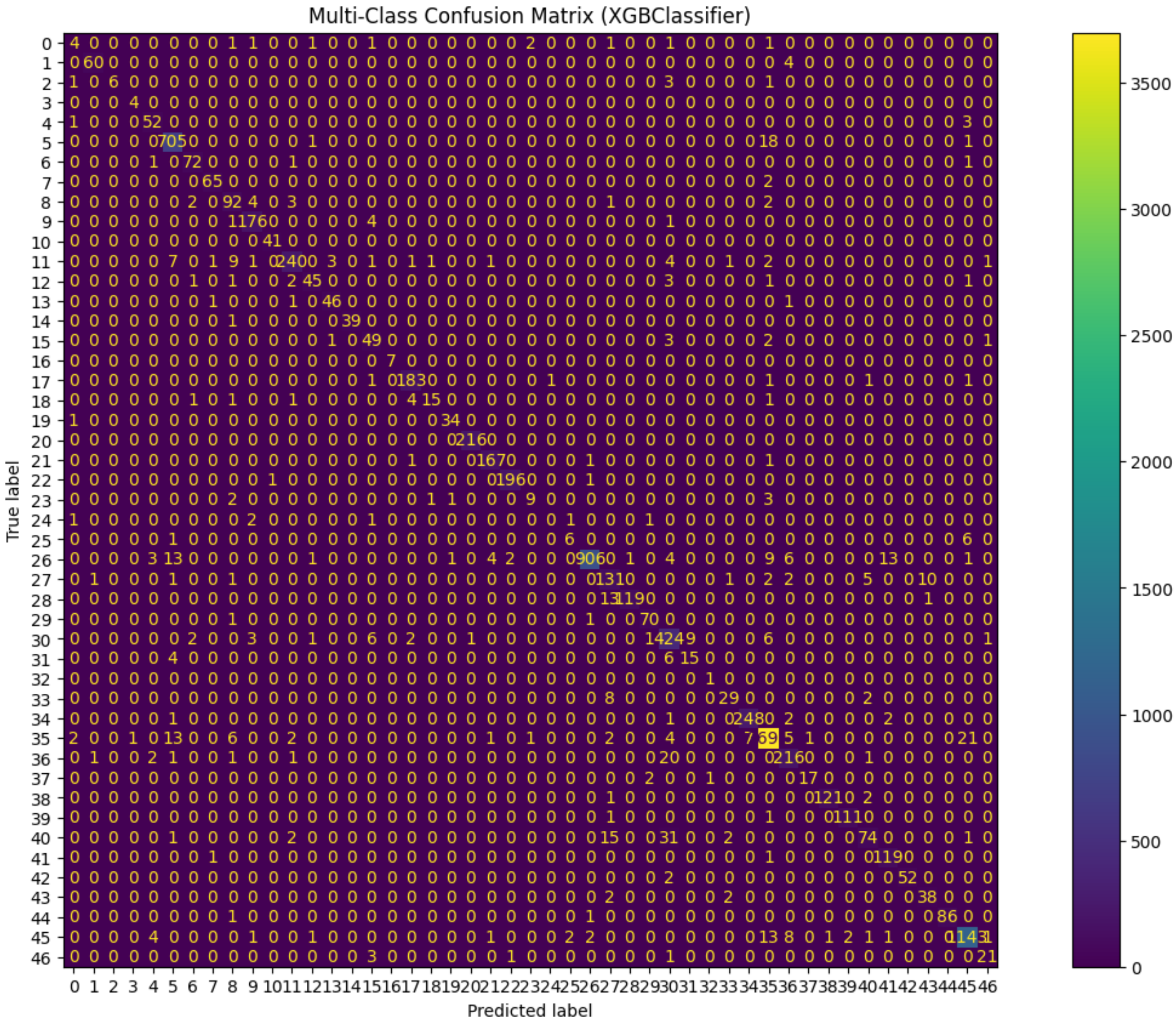
In [37]:

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

In [38]:

```python
# 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)');
```



In [39]:
```
# performance stats
print(classification_report(test_label,predictions))
```

```
              precision    recall  f1-score   support

           0       0.40      0.31      0.35        13
           1       0.97      0.94      0.95        64
           2       1.00      0.55      0.71        11
           3       0.80      1.00      0.89         4
           4       0.84      0.93      0.88        56
           5       0.94      0.97      0.96       725
           6       0.92      0.96      0.94        75
           7       0.96      0.97      0.96        67
           8       0.78      0.88      0.83       104
           9       0.94      0.97      0.95       182
          10       0.98      1.00      0.99        41
          11       0.95      0.88      0.91       273
          12       0.90      0.83      0.87        54
          13       0.92      0.94      0.93        49
          14       1.00      0.97      0.99        40
          15       0.74      0.88      0.80        56
          16       1.00      1.00      1.00         7
          17       0.96      0.97      0.97       188
          18       0.88      0.65      0.75        23
          19       0.94      0.97      0.96        35
          20       1.00      1.00      1.00       216
          21       0.96      0.98      0.97       170
          22       0.98      0.99      0.99       198
          23       0.75      0.56      0.64        16
          24       0.00      0.00      0.00         6
          25       0.67      0.46      0.55        13
          26       0.99      0.94      0.97       964
          27       0.75      0.85      0.80       154
          28       0.99      0.89      0.94       133
          29       0.95      0.97      0.96        72
          30       0.83      0.93      0.88       456
          31       0.62      0.60      0.61        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       0.89      0.89      0.89       243
          37       0.94      0.85      0.89        20
          38       0.99      0.98      0.98       124
          39       0.98      0.98      0.98       113
          40       0.86      0.59      0.70       126
          41       0.88      0.98      0.93       121
          42       1.00      0.96      0.98        54
          43       0.78      0.90      0.84        42
          44       1.00      0.98      0.99        88
          45       0.97      0.97      0.97      1181
          46       0.84      0.81      0.82        26

    accuracy                           0.95     10686
   macro avg       0.87      0.86      0.86     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=No
ne,
              enable_categorical=False, eval_metric=None, feature_types=No
ne,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=No
ne,
```

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.xlsx')
```

In [46]:

```
```

Let's Debug a little

In [47]:

```
# let's have a look at the first five rows
df_2_copy
```

Out[47]:

	Product Group	Net Weight	labels	desc_2	Type_2
0	custom	0.0	screws	pan	9999-pan
1	generic	0.0	nan	stud	9999-stud
2	generic	0.0	pins	pin	9999-pin
3	custom	0.0	screws	screw	9999-screw
4	custom	0.0	screws	hd	9999-hd
...	...	...	...	...	...
80188	custom	0.0	nan	seal	4900-seal
80189	proprty	0.0	nan	hpmvdc	3025-hpmvdc
80190	generic	0.0	nan	nut	4404-nut
80191	proprty	0.0	nan	wire	8000-wire
80192	custom	0.0	nan	hinge	3038-hinge

80162 rows × 5 columns

In [48]:

```
# let's look at all the samples with Type_2:9999-stud  and not null/nan
debug_stud = df_2_copy[(df_2_copy['Type_2'] == '9999-stud') & (df_2_copy['labels']
debug_stud
```

Out[48]:

	Product Group	Net Weight	labels	desc_2	Type_2
3030	custom	0.0	collars	stud	9999-stud
59953	custom	0.0	nuts	stud	9999-stud
59954	custom	0.0	nuts	stud	9999-stud
60086	custom	0.0	nuts	stud	9999-stud
60092	custom	0.0	nuts	stud	9999-stud
61672	custom	0.0	nuts	stud	9999-stud
62586	omni	0.0	nuts	stud	9999-stud

In [49]:

```
# let's view the result in our virgin data
df_original.loc[debug_stud.index][['Item Number','Item Full Description Final','Typ
```

Out[49]:

	Item Number	Item Full Description Final	Type	Label from SKU Hierarchy3
3030	500143060	499A907AAP1 01-10865 1/4-20X3/4 STEEL WELD STU...	9999	Collars
59953	500141249	41A303833G14 5/8-11X8.88 WELD ASSEMBLY STUD & ...	9999	Nuts
59954	500141258	41A319834P1 3/4-10NC-1AX13.67 O/A/L FULL THREA...	9999	Nuts
60086	500141255	41A303833G7 5/8-11X7.44" WELD ASSEMBLY STUD & ...	9999	Nuts
60092	500141253	41A303833G2 5/8-11X6 WELD ASSEMBLY STUD & NUT ...	9999	Nuts
61672	500141250	41A303833G15 5/8-11X6.31 WELD ASSEMBLY STUD & ...	9999	Nuts
62586	500144619	00-619-488-637 ASTM A193 B7 FULL THREAD STUD 1...	9999	Nuts

In [50]:

```python
# let's look at studs that were possibly mislabelled
df_2_copy[df_2_copy['desc_2'] == 'stud']['labels'].value_counts()
```

Out[50]:

```
nan                                1125
stud                                565
nuts                                 76
screws                               25
bolts                                23
o-rings                              18
kitting                              11
cable management                      7
collars                               6
wire terminals & connectors           4
brackets & reinforcement braces       3
clamps                                3
washers                               2
springs                               2
bearings                              2
inserts                               2
electrical switches                   1
heat-shrink tubing                    1
gaskets                               1
grommets                              1
cables                                1
Name: labels, dtype: int64
```