XGBclassifier

March 10, 2023

0.0.1 Here is the predictive model to distinguish between the main product categories in the dataset provided by the Otto Group company.

0.0.2 Provided dataset has 93 features and more than 2000,000 products.

0.0.3 For reducing high dimensionality, Linear Discriminant Analysis, a method of Dimensionality Reduction is used.

0.0.4 This model is built on the XG Boost Classifier algorithm, a popular and efficient algorithm which attempts to predict the target variable accurately.

Importing Libraries...

[2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Loading data...

[3]:
```
data = pd.read_csv('/kaggle/input/otto-group-product-classification-challenge/
  ↪train.csv')
data.head()
```

[3]:

	id	feat_1	feat_2	feat_3	feat_4	feat_5	feat_6	feat_7	feat_8	feat_9	\
0	1	1	0	0	0	0	0	0	0	0	
1	2	0	0	0	0	0	0	0	1	0	
2	3	0	0	0	0	0	0	0	1	0	
3	4	1	0	0	1	6	1	5	0	0	
4	5	0	0	0	0	0	0	0	0	0	

	…	feat_85	feat_86	feat_87	feat_88	feat_89	feat_90	feat_91	\
0	…	1	0	0	0	0	0	0	
1	…	0	0	0	0	0	0	0	
2	…	0	0	0	0	0	0	0	
3	…	0	1	2	0	0	0	0	
4	…	1	0	0	0	0	1	0	

	feat_92	feat_93	target
0	0	0	Class_1
1	0	0	Class_1

```
2        0        0  Class_1
3        0        0  Class_1
4        0        0  Class_1

[5 rows x 95 columns]
```

[4]:
```
data.columns
```

[4]:
```
Index(['id', 'feat_1', 'feat_2', 'feat_3', 'feat_4', 'feat_5', 'feat_6',
       'feat_7', 'feat_8', 'feat_9', 'feat_10', 'feat_11', 'feat_12',
       'feat_13', 'feat_14', 'feat_15', 'feat_16', 'feat_17', 'feat_18',
       'feat_19', 'feat_20', 'feat_21', 'feat_22', 'feat_23', 'feat_24',
       'feat_25', 'feat_26', 'feat_27', 'feat_28', 'feat_29', 'feat_30',
       'feat_31', 'feat_32', 'feat_33', 'feat_34', 'feat_35', 'feat_36',
       'feat_37', 'feat_38', 'feat_39', 'feat_40', 'feat_41', 'feat_42',
       'feat_43', 'feat_44', 'feat_45', 'feat_46', 'feat_47', 'feat_48',
       'feat_49', 'feat_50', 'feat_51', 'feat_52', 'feat_53', 'feat_54',
       'feat_55', 'feat_56', 'feat_57', 'feat_58', 'feat_59', 'feat_60',
       'feat_61', 'feat_62', 'feat_63', 'feat_64', 'feat_65', 'feat_66',
       'feat_67', 'feat_68', 'feat_69', 'feat_70', 'feat_71', 'feat_72',
       'feat_73', 'feat_74', 'feat_75', 'feat_76', 'feat_77', 'feat_78',
       'feat_79', 'feat_80', 'feat_81', 'feat_82', 'feat_83', 'feat_84',
       'feat_85', 'feat_86', 'feat_87', 'feat_88', 'feat_89', 'feat_90',
       'feat_91', 'feat_92', 'feat_93', 'target'],
      dtype='object')
```

[5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61878 entries, 0 to 61877
Data columns (total 95 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   id       61878 non-null  int64
 1   feat_1   61878 non-null  int64
 2   feat_2   61878 non-null  int64
 3   feat_3   61878 non-null  int64
 4   feat_4   61878 non-null  int64
 5   feat_5   61878 non-null  int64
 6   feat_6   61878 non-null  int64
 7   feat_7   61878 non-null  int64
 8   feat_8   61878 non-null  int64
 9   feat_9   61878 non-null  int64
 10  feat_10  61878 non-null  int64
 11  feat_11  61878 non-null  int64
 12  feat_12  61878 non-null  int64
 13  feat_13  61878 non-null  int64
```

```
14  feat_14  61878 non-null  int64
15  feat_15  61878 non-null  int64
16  feat_16  61878 non-null  int64
17  feat_17  61878 non-null  int64
18  feat_18  61878 non-null  int64
19  feat_19  61878 non-null  int64
20  feat_20  61878 non-null  int64
21  feat_21  61878 non-null  int64
22  feat_22  61878 non-null  int64
23  feat_23  61878 non-null  int64
24  feat_24  61878 non-null  int64
25  feat_25  61878 non-null  int64
26  feat_26  61878 non-null  int64
27  feat_27  61878 non-null  int64
28  feat_28  61878 non-null  int64
29  feat_29  61878 non-null  int64
30  feat_30  61878 non-null  int64
31  feat_31  61878 non-null  int64
32  feat_32  61878 non-null  int64
33  feat_33  61878 non-null  int64
34  feat_34  61878 non-null  int64
35  feat_35  61878 non-null  int64
36  feat_36  61878 non-null  int64
37  feat_37  61878 non-null  int64
38  feat_38  61878 non-null  int64
39  feat_39  61878 non-null  int64
40  feat_40  61878 non-null  int64
41  feat_41  61878 non-null  int64
42  feat_42  61878 non-null  int64
43  feat_43  61878 non-null  int64
44  feat_44  61878 non-null  int64
45  feat_45  61878 non-null  int64
46  feat_46  61878 non-null  int64
47  feat_47  61878 non-null  int64
48  feat_48  61878 non-null  int64
49  feat_49  61878 non-null  int64
50  feat_50  61878 non-null  int64
51  feat_51  61878 non-null  int64
52  feat_52  61878 non-null  int64
53  feat_53  61878 non-null  int64
54  feat_54  61878 non-null  int64
55  feat_55  61878 non-null  int64
56  feat_56  61878 non-null  int64
57  feat_57  61878 non-null  int64
58  feat_58  61878 non-null  int64
59  feat_59  61878 non-null  int64
60  feat_60  61878 non-null  int64
61  feat_61  61878 non-null  int64
```

```
 62  feat_62  61878 non-null  int64
 63  feat_63  61878 non-null  int64
 64  feat_64  61878 non-null  int64
 65  feat_65  61878 non-null  int64
 66  feat_66  61878 non-null  int64
 67  feat_67  61878 non-null  int64
 68  feat_68  61878 non-null  int64
 69  feat_69  61878 non-null  int64
 70  feat_70  61878 non-null  int64
 71  feat_71  61878 non-null  int64
 72  feat_72  61878 non-null  int64
 73  feat_73  61878 non-null  int64
 74  feat_74  61878 non-null  int64
 75  feat_75  61878 non-null  int64
 76  feat_76  61878 non-null  int64
 77  feat_77  61878 non-null  int64
 78  feat_78  61878 non-null  int64
 79  feat_79  61878 non-null  int64
 80  feat_80  61878 non-null  int64
 81  feat_81  61878 non-null  int64
 82  feat_82  61878 non-null  int64
 83  feat_83  61878 non-null  int64
 84  feat_84  61878 non-null  int64
 85  feat_85  61878 non-null  int64
 86  feat_86  61878 non-null  int64
 87  feat_87  61878 non-null  int64
 88  feat_88  61878 non-null  int64
 89  feat_89  61878 non-null  int64
 90  feat_90  61878 non-null  int64
 91  feat_91  61878 non-null  int64
 92  feat_92  61878 non-null  int64
 93  feat_93  61878 non-null  int64
 94  target   61878 non-null  object
dtypes: int64(94), object(1)
memory usage: 44.8+ MB
```

[6]:
```
nulldata = data.isnull().sum()
print(nulldata[nulldata>0])
```

```
Series([], dtype: int64)
```

[7]:
```
data.target.value_counts()
```

[7]:
```
Class_2    16122
Class_6    14135
Class_8     8464
Class_3     8004
```

```
Class_9     4955
Class_7     2839
Class_5     2739
Class_4     2691
Class_1     1929
Name: target, dtype: int64
```

Separating dependent and independent variables

[8]:
```
x = data.drop('target',axis=1)
y = data[['target']]
```

Splitting the training data and validation data

[9]:
```
from sklearn.model_selection import train_test_split
x_train,x_valid,y_train,y_valid = train_test_split(x,y,test_size=0.2)
```

Standardization

[10]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_valid = sc.transform(x_valid)
```

[11]:
```
x_train.shape
```

[11]:
```
(49502, 94)
```

[12]:
```
data.shape
```

[12]:
```
(61878, 95)
```

Dimensionality Reduction

[13]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
x_train = lda.fit_transform(x_train,y_train)
x_valid = lda.transform(x_valid)
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py:993:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

[14]:
```
x_train.shape
```

[14]:
```
(49502, 8)
```

Sorting Index

[15]:
```
a = pd.DataFrame(y_train.value_counts())
b = a.sort_index(axis=0)
b.index
```

[15]:
```
MultiIndex([('Class_1',),
            ('Class_2',),
            ('Class_3',),
            ('Class_4',),
            ('Class_5',),
            ('Class_6',),
            ('Class_7',),
            ('Class_8',),
            ('Class_9',)],
           names=['target'])
```

One Hot Encoding of the dependent variable

[16]:
```
from sklearn.preprocessing import OneHotEncoder
OHE = OneHotEncoder(handle_unknown='ignore',sparse=False)
y_train = pd.DataFrame(OHE.fit_transform(y_train),columns = b.index)
y_valid = pd.DataFrame(OHE.transform(y_valid),columns = b.index)
```

[17]:
```
y_valid.head()
```

[17]:

target	Class_1	Class_2	Class_3	Class_4	Class_5	Class_6	Class_7	Class_8	Class_9
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
1	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
4	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

XG Boost Classifier

[18]:
```
from xgboost import XGBClassifier
```

[19]:
```
my_model = XGBClassifier()
my_model.fit(x_train,y_train)
y_pred = my_model.predict(x_valid)
```

Verifying Metrics

[20]:
```
from sklearn.metrics import accuracy_score,␣
↪confusion_matrix,classification_report
from sklearn.metrics import f1_score,precision_score,recall_score
```

[21]:
```
print('Accuracy : ',accuracy_score(y_valid,y_pred))
print('\nclassification report : \n',classification_report(y_valid,y_pred))
```

```
print('\nf1 score : \n',f1_score(y_valid,y_pred,average=None))
print('\nprecision_score : \n',precision_score(y_valid,y_pred,average=None))
print('\nrecall_score : \n',recall_score(y_valid,y_pred,average=None))
```

```
Accuracy :  0.989414996767938

classification report :
               precision    recall  f1-score   support

           0       0.99      1.00      0.99       388
           1       1.00      0.99      0.99      3152
           2       0.98      0.99      0.98      1647
           3       0.97      0.97      0.97       541
           4       1.00      0.99      1.00       539
           5       1.00      0.99      1.00      2775
           6       0.98      0.98      0.98       603
           7       0.99      1.00      0.99      1685
           8       1.00      0.99      1.00      1046

   micro avg       0.99      0.99      0.99     12376
   macro avg       0.99      0.99      0.99     12376
weighted avg       0.99      0.99      0.99     12376
 samples avg       0.99      0.99      0.99     12376


f1 score :
 [0.99485861 0.99491902 0.98427102 0.96952909 0.99628253 0.99602888
 0.97840532 0.99318922 0.99520614]

precision_score :
 [0.99230769 0.99586777 0.98071127 0.96863469 0.9981378  0.99783002
 0.98003328 0.99113475 0.99807692]

recall_score :
 [0.99742268 0.99397208 0.98785671 0.97042514 0.99443414 0.99423423
 0.97678275 0.99525223 0.99235182]
/opt/conda/lib/python3.7/site-packages/sklearn/metrics/_classification.py:1318:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in samples with no predicted labels. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Implementing the model to the test data

[22]:
```
x_test = pd.read_csv('/kaggle/input/otto-group-product-classification-challenge/
  ↪test.csv')
x_test = sc.transform(x_test)
```

```
x_test = lda.transform(x_test)
```

[23]:
```
x_test = pd.DataFrame(x_test)
l = list(x_test.index)
indexlist = pd.DataFrame(l,columns = ['id'])
```

[24]:
```
ind = ␣
↪['Class_1','Class_2','Class_3','Class_4','Class_5','Class_6','Class_7','Class_8','Class_9']
```

Predicting the test data

[25]:
```
predictions = pd.DataFrame(my_model.predict(x_test),columns = ind)
```

[26]:
```
predictions.head()
```

[26]:
```
   Class_1  Class_2  Class_3  Class_4  Class_5  Class_6  Class_7  Class_8  \
0      0.0      1.0      0.0      0.0      0.0      0.0      0.0      0.0
1      1.0      0.0      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
3      0.0      1.0      0.0      0.0      0.0      0.0      0.0      0.0
4      1.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0

   Class_9
0      0.0
1      0.0
2      0.0
3      0.0
4      0.0
```

[27]:
```
sample = pd.read_csv('/kaggle/input/otto-group-product-classification-challenge/
↪sampleSubmission.csv')
sample.head()
```

[27]:
```
   id  Class_1  Class_2  Class_3  Class_4  Class_5  Class_6  Class_7  Class_8  \
0   1        1        0        0        0        0        0        0        0
1   2        1        0        0        0        0        0        0        0
2   3        1        0        0        0        0        0        0        0
3   4        1        0        0        0        0        0        0        0
4   5        1        0        0        0        0        0        0        0

   Class_9
0        0
1        0
2        0
3        0
4        0
```

[28]:
```
x_test.head()
```

[28]:

	0	1	2	3	4	5	6	\
0	-9.580837	-0.467278	-0.992099	0.133397	-1.221610	0.324263	1.671230	
1	-8.847654	1.354975	-0.165058	1.131874	2.524327	-0.472635	-1.096662	
2	-8.556826	4.511615	1.041851	0.749542	1.865802	-0.234163	-0.401307	
3	-9.510612	-0.807562	-1.101659	-0.975898	0.459598	0.723522	-1.317780	
4	-8.720170	-0.265293	-0.023465	-0.039500	3.749187	-1.060294	0.109233	

	7
0	2.902215
1	-0.771124
2	-1.031381
3	1.453158
4	0.482871

[29]:
```
output = pd.concat((indexlist,predictions),axis=1)
output.to_csv('submission.csv',index=False,header=1)
```

[ ]:
```

```