Finding two best performing multilabel classifiers for predicting type of defects in plates.

Vandana Dattatray Pathare
University of Michigan-Dearborn
Dearborn, Michigan

***Abstract*— This paper describes various multilabel classifiers used for predicting types of defects given the dataset of defective plates. The aim of building various classifiers is to reach to the stage of getting two best performing multilabel classifiers for the given dataset.**

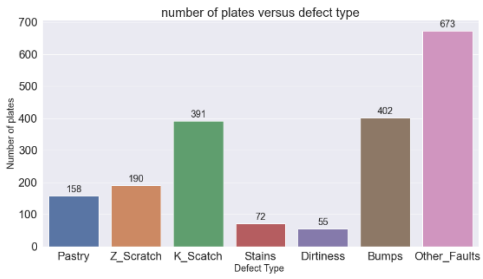
I. INTRODUCTION

Multilabel classification is one of the category in the world of classification. It basically means that in a scenario where there are n inputs and more than one outputs(labels), for every single set of input the classification will have more than one labels generated[1]. Each label can have more than two classes. In our dataset, we have 27 input features which represent images of defective plates and we have 7 output labels each having binary classes(0 or 1) describing the types of defect. We will use different classifiers that are designed for multilabel dataset which are from list of multioutput classifiers, multiclass classifiers, multilabel classifiers from skmultilearn.

II. VISUALIZATIONS AND DATA ANALYSIS

The dataset has 1941 rows and 34 columns. Each row describes the dimensions and shape of the plate along with the types of defect. First 27 columns talk about image 'X_Minimum', 'X_Maximum', 'Y_Minimum', 'Y_Maximum', 'Pixels_Areas','X_Perimeter','Y_Perimeter','Sum_of_Luminosity','Minimum_of_Luminosity','Maximum_of_Luminosity','Length_of_Conveyer','TypeOfSteel_A300','TypeOfSteel_A400','Steel_Plate_Thickness','Edges_Index','Empty_Index','Square_Index','Outside_X_Index"Edges_X_Index','Edges_Y_Index','Outside_Global_Index','LogOfAreas','Log_X_Index','Log_Y_Index','Orientation_Index','Luminosity_Index','SigmoidOfAreas'. The last 7 columns represent defect type of the plate which are 'Pastry','Z_Scratch','K_Scatch', 'Stains','Dirtiness', 'Bumps', 'Other_Faults'.

The describe function denotes the data to be spread widely. Below is the barplot that shows count of defect types in the plates.



III. DEVELOPMENT METHODOLOGY

This section has four subparts.Each of which introduces new technique of performing multilabel classification. I have used sklearn train-test split on the data before applying any of the following classifiers.

A. *Multilabel classifiers by scikit-multilearn*

There is a special class of models designed for multilabel classification and are made available in skmultilearn library[3]. Using Binary relevance method in which an ensemble of various single-label classifiers is trained. Linear SVC(support vector classifier) is used out of others available[4]. The accuracy of model was only 15%. There might be various possible reasons and one of which might be the use of linearSVC. There might be sone other classifier that would have worked well.

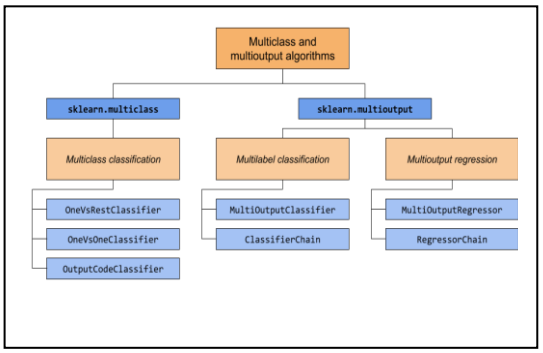
B. *Using neural network*

Neural network can be used to predict multilabel outputs. For doing the same, we have to confirm the number of output dimensions equal to number of labels. We need to consider binary cross entropy which is suitable for our datatype. The best performing optimizer Adam that makes sure we reach the global minima is used. Layers were set to activation function-Relu(rectified linear unit)[5]. The last layer had sigmoid function used. The model built based on this setting provided the accuracy for 36 percent only. Even after changing the setting slightly, the model didn't improve much. One of the main reason suspected is the less amount of data and large amount of labels to predict[2].

C. *Multioutput Classifiers*

Sklearn has provides documentation for types of packages to predict the multilabel data[6].

Figure below shows the types of classifiers:



Any traditional classifier from sklearn can be modified to support multilabel classification by with the multioutput classifier. This strategy fits one classifier per target. Since random forest are good at avoiding overfitting and also good at selecting the best features. We used randomforest with multioutput classifier in order to predict the labels. The accuracy went exceptionally high. It is 65 percent which made this classifier best of all the above.

D. Multiclass Classifiers

Digging more into the nature of labels, it was seen that each datapoint(plate) have only one defect out of the seven. Below code output shows the proof of this.

```
rowSums = data_y.sum(axis=1)
print(rowSums.unique(), len(rowSums))
```

```
[1] 1941
```

This led to the conclusion of trying to use a multiclass multioutput classifier which is onevsrest classifier provided in the sklearn,multiclass. Logistic regression was used along with it. In onevsrest strategy, multiple independent classifiers are built[7]. For the new unseen data,the class having highest confidence is set as the label. This method surprisingly gave accuracy of 90 percent. This turned out to be the best method for this datset.

CONCLUSIONS

The two classifiers from the category of multioutput and multilabel proved to give better accuracies. The type of classifier that would give higher accuracy depends on the nature of inputs and majorly the outputs. We got 65 percent accuracy with multioutput classifier paired with randomforest. We achieved highest accuracy score of 90 percent on both test and train when we used multiclass classifier with logistic regression.

REFERENCES

[1] https://xang1234.github.io/multi-label/

[2] https://www.machinecurve.com/index.php/2020/11/16/creating-a-multilabel-neural-network-classifier-with-tensorflow-and-keras/

[3] http://scikit.ml/api/skmultilearn.problem_transform.lp.html

[4] https://medium.com/technovators/machine-learning-based-multi-label-text-classification-9a0e17f88bb4

[5] https://machinelearningmastery.com/multi-label-classification-with-deep-learning/

[6] https://scikit-learn.org/stable/modules/multiclass.html

[7] https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff

APPENDIX

A. Code for the two best preforming multilabel classifiers.

```
#!/usr/bin/env python
# coding: utf-8

# In[11]:


import pandas as pd
import numpy as np
import seaborn as sns # For pairplots and other visualizati
ons
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.model_selection import train_test_split

from sklearn.metrics import hamming_loss, accuracy_scor
e


# In[2]:


data = pd.read_csv("C:/Users/pvandana/Downloads/multi
variate/project/surface_faults.csv")


# In[13]:


data.head(5)


# In[ ]:




# In[ ]:




# In[12]:


categories = list(data_y.columns.values)
sns.set(font_scale = 2)
plt.figure(figsize=(15,8))
ax= sns.barplot(categories, data_y.sum().values)
plt.title("number of plates versus defect type", fontsize=24
)
plt.ylabel('Number of plates', fontsize=18)
plt.xlabel('Defect Type ', fontsize=18)
#adding the text labels
rects = ax.patches
labels = data_y.sum().values
for rect, label in zip(rects, labels):
    height = rect.get_height()
    ax.text(rect.get_x() + rect.get_width()/2, height + 5, lab
el, ha='center', va='bottom', fontsize=18)
plt.show()


# In[3]:


data.columns


# In[4]:


data_y = data[['Pastry', 'Z_Scratch', 'K_Scatch', 'Stains',
       'Dirtiness', 'Bumps', 'Other_Faults']]
data_X = data[['X_Minimum', 'X_Maximum', 'Y_Minimu
m', 'Y_Maximum', 'Pixels_Areas',
       'X_Perimeter', 'Y_Perimeter', 'Sum_of_Luminosity',
       'Minimum_of_Luminosity', 'Maximum_of_Luminosit
y', 'Length_of_Conveyer',
       'TypeOfSteel_A300', 'TypeOfSteel_A400', 'Steel_Plat
e_Thickness',
       'Edges_Index', 'Empty_Index', 'Square_Index', 'Outsid
e_X_Index',
       'Edges_X_Index', 'Edges_Y_Index', 'Outside_Global_
Index', 'LogOfAreas',
       'Log_X_Index', 'Log_Y_Index', 'Orientation_Index', '
Luminosity_Index',
       'SigmoidOfAreas']]


# https://scikit-learn.org/stable/modules/multiclass.html
#multilabel-classification

# In[5]:


X_train, X_test, y_train, y_test = train_test_split(data_X,
data_y, test_size=0.30, random_state=42)
from sklearn.datasets import make_classification
from sklearn.multioutput import MultiOutputClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.utils import shuffle
import numpy as np
n_samples, n_features = X_train.shape # 10,100
n_outputs = data_y.shape[1] # 3
n_classes = 3
forest = RandomForestClassifier(random_state=1)
multi_target_forest = MultiOutputClassifier(forest, n_jobs
=-1)
multi_target_forest.fit(X_train, y_train).predict(X_test)
pred = multi_target_forest.fit(X_train, y_train).predict(X_
test)
print(accuracy_score(y_test, pred))
print(hamming_loss(y_test, pred))


# https://towardsdatascience.com/journey-to-the-center-of
-multi-label-classification-384c40229bff

# In[10]:
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
from sklearn.multiclass import OneVsRestClassifier
X_train, X_test, y_train, y_test = train_test_split(data_X,
data_y, test_size=0.33, random_state=42)
# Using pipeline for applying logistic regression and one v
s rest classifier
LogReg_pipeline = Pipeline([
                ('clf', OneVsRestClassifier(LogisticRegression(s
olver='sag'), n_jobs=-1)),
            ])
categories = ['Pastry', 'Z_Scratch', 'K_Scatch', 'Stains', 'Dir
tiness', 'Bumps',
       'Other_Faults']
for category in categories:
    print('**Processing {} comments...**'.format(category)
)

    # Training logistic regression model on train data
    LogReg_pipeline.fit(X_train, y_train[category])

    # calculating test accuracy
    prediction = LogReg_pipeline.predict(X_test)
    print('Test set accuracy is {}'.format(accuracy_score(y_
test[category], prediction)))


    prediction = LogReg_pipeline.predict(X_train)
    print('Train set accuracy is {}'.format(accuracy_score(y
_train[category], prediction)))
    print("\n")
```

B. OTHER CLASSIFIERS

In[86]:

```
data_y = data[['Pastry', 'Z_Scratch', 'K_Scatch', 'Stains',
       'Dirtiness', 'Bumps', 'Other_Faults']]
data_X = data[['X_Minimum', 'X_Maximum', 'Y_Minimu
m', 'Y_Maximum', 'Pixels_Areas',
       'X_Perimeter', 'Y_Perimeter', 'Sum_of_Luminosity',
       'Minimum_of_Luminosity', 'Maximum_of_Luminosit
y', 'Length_of_Conveyer',
       'TypeOfSteel_A300', 'TypeOfSteel_A400', 'Steel_Plat
e_Thickness',
       'Edges_Index', 'Empty_Index', 'Square_Index', 'Outsid
e_X_Index',
       'Edges_X_Index', 'Edges_Y_Index', 'Outside_Global_
Index', 'LogOfAreas',
       'Log_X_Index', 'Log_Y_Index', 'Orientation_Index', '
Luminosity_Index',
       'SigmoidOfAreas']]
```

In[64]:

```
X_train, X_test, y_train, y_test = train_test_split(data_X,
data_y, test_size=0.30, random_state=42)
from sklearn.datasets import make_classification
from sklearn.multioutput import MultiOutputClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.utils import shuffle
import numpy as np
n_samples, n_features = X_train.shape # 10,100
n_outputs = data_y.shape[1] # 3
n_classes = 3
forest = RandomForestClassifier(random_state=1)
multi_target_forest = MultiOutputClassifier(forest, n_jobs
=-1)
multi_target_forest.fit(X_train, y_train).predict(X_test)
pred = multi_target_forest.fit(X_train, y_train).predict(X_
test)
print(accuracy_score(y_test, pred))
print(hamming_loss(y_test, pred))
```

In[68]:

```
from sklearn.svm import LinearSVC
from skmultilearn.problem_transform import BinaryRelev
ance

BinaryRelSVC = BinaryRelevance(LinearSVC())
BinaryRelSVC.fit(X_train, y_train)

BinaryRelSVCPreds = BinaryRelSVC.predict(X_test)
```

In[72]:

```
from scipy import sparse
prediction_of_svc = sparse.lil_matrix(BinaryRelSVCPred
s).toarray()
```

In[75]:

```
print(accuracy_score(y_test, prediction_of_svc))
print(hamming_loss(y_test, prediction_of_svc))
```

In[77]:

```
from sklearn.naive_bayes import MultinomialNB
```

In[85]:

```
from sklearn.svm import LinearSVC
from sklearn.multiclass import OneVsRestClassifier
svmClassifier = OneVsRestClassifier(LinearSVC(), n_jobs=-1)
svmClassifier.fit(X_train, y_train)

svmPreds = svmClassifier.predict(X_test)
print(accuracy_score(y_test, svmPreds))
print(hamming_loss(y_test, svmPreds))
```

In[ ]:

In[ ]:

In[88]:

```
# Imports
from sklearn.datasets import make_multilabel_classification
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.losses import binary_crossentropy
from tensorflow.keras.optimizers import Adam

# Configuration options
n_samples = data_X.shape[1]
n_features = 27
n_classes = 7
n_labels = 7
n_epochs = 50
random_state = 42
batch_size = 250
verbosity = 1
validation_split = 0.2

# Create dataset
# X, y = make_multilabel_classification(n_samples=n_samples, n_features=n_features, n_classes=n_classes, n_labels=n_labels, random_state=random_state)

# Split into training and testing data
X_train, X_test, y_train, y_test = train_test_split(data_X, data_y, test_size=0.33, random_state=random_state)

# Create the model
model = Sequential()
model.add(Dense(32, activation='relu', input_dim=n_features))
model.add(Dense(16, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(n_classes, activation='sigmoid'))

# Compile the model
model.compile(loss=binary_crossentropy,
              optimizer=Adam(),
              metrics=['accuracy'])

# Fit data to model
model.fit(X_train, y_train,
          batch_size=batch_size,
          epochs=n_epochs,
          verbose=verbosity,
          validation_split=validation_split)

# Generate generalization metrics
score = model.evaluate(X_test, y_test, verbose=0)
print(f'Test loss: {score[0]} / Test accuracy: {score[1]}')
```

In[ ]:

In[97]:

```
from sklearn.metrics import multilabel_confusion_matrix, f1_score
# Create the SVM
svm = LinearSVC(random_state=42)

# Make it an Multilabel classifier
multilabel_classifier = MultiOutputClassifier(svm, n_jobs=-1)

# Fit the data to the Multilabel classifier
multilabel_classifier = multilabel_classifier.fit(X_train, y_train)

# Get predictions for test data
y_test_pred = multilabel_classifier.predict(X_test)

# Generate multiclass confusion matrices
matrices = multilabel_confusion_matrix(y_test, y_test_pred)
```

```
accuracy_score(y_test, y_test_pred)
```

```
# In[92]:
```

```
matrices
```

```
# In[117]:
```

```
data_y['Pastry']
```

```
# In[116]:
```

```
data_y.columns
```

```
# In[121]:
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
from sklearn.multiclass import OneVsRestClassifier
X_train, X_test, y_train, y_test = train_test_split(data_X,
data_y, test_size=0.33, random_state=random_state)
# Using pipeline for applying logistic regression and one v
s rest classifier
LogReg_pipeline = Pipeline([
                ('clf', OneVsRestClassifier(LogisticRegression(s
olver='sag'), n_jobs=-1)),
            ])
categories = ['Pastry', 'Z_Scratch', 'K_Scatch', 'Stains', 'Dir
tiness', 'Bumps',
       'Other_Faults']
for category in categories:
    print('**Processing {} comments...**'.format(category)
)

    # Training logistic regression model on train data
    LogReg_pipeline.fit(X_train, y_train[category])

    # calculating test accuracy
    prediction = LogReg_pipeline.predict(X_test)
    print('Test accuracy is {}'.format(accuracy_score(y_test
[category], prediction)))
    print("\n")
```

```
# In[123]:
```

```
# using binary relevance
from skmultilearn.problem_transform import BinaryRelev
ance
from sklearn.naive_bayes import GaussianNB
# initialize binary relevance multi-label classifier
# with a gaussian naive bayes base classifier
classifier = BinaryRelevance(GaussianNB())
# train
classifier.fit(X_train, y_train)
# predict
predictions = classifier.predict(X_test)
# accuracy
print("Accuracy = ",accuracy_score(y_test,predictions))
```

```
# In[ ]:
```