How to solve Multilabel Classification problems?

Entrée [1]:
```
import numpy as np
import pandas as pd
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import Model
from tensorflow.keras.layers import Dense, Embedding, Input, Flatten,

from sklearn.utils import shuffle
from sklearn.preprocessing import MultiLabelBinarizer
```

Entrée [90]:
```
data = pd.read_csv('so_data.csv', names=['tags', 'original_tags', 't
data.head()
```

Out[90]:

	tags	original_tags	text
0	matplotlib,pandas	python,matplotlib,pandas	setting xticks and yticks for scatter plot mat...
1	scikitlearn,keras	python,numpy,scikit-learn,keras,grid-search	gridseachcv - valueerror: found input variable...
2	matplotlib,scikitlearn	python,numpy,matplotlib,scikit-learn,nmf	non negative matrix factorisation in python on...
3	pandas,tensorflow	python,pandas,tensorflow,time-series	avocado equivalent to avocado.dataframe.resamp...
4	matplotlib,pandas	python,matplotlib,plot,pandas	how to plot on avocado python i have a data fr...

We drop the column "orginal_tags" from the dataset and drop missing values with code lines below.

Entrée [91]:
```
data = data.drop(columns=['original_tags'])
data = data.dropna()

data = shuffle(data, random_state=22)
data.head()
```

Out[91]:

	tags	text
182914	tensorflow,keras	avocado image captioning model not compiling b...
48361	pandas	return excel file from avocado with flask in f...
181447	tensorflow,keras	validating with generator (avocado) i'm trying...
66307	pandas	avocado multiindex dataframe selecting data gi...
11283	pandas	get rightmost non-zero value position for each...

We convert input labels into multilabel labels, each example can belong to multiple classes.

Entrée [116]:
```
def tags_encoding(data):
    '''
    Converts between this intuitive format and the supported multil
    @param dataset: All the dataset
    return classes, num_tags, tags_encoded
    '''
    tags_split = [tags.split(',') for tags in data['tags'].values]
    tag_encoder = MultiLabelBinarizer()
    tags_encoded = tag_encoder.fit_transform(tags_split)
    classes = tag_encoder.classes_
    num_tags = len(tags_encoded[0])
    return classes, num_tags, tags_encoded
```

and you will end up with

Entrée [159]:
```
classes, num_tags, tags_encoded = tags_encoding(data)
```

Entrée [121]:
```
classes
```

Out[121]:
```
array(['keras', 'matplotlib', 'pandas', 'scikitlearn', 'tensorflow']
,
      dtype=object)
```

Entrée [122]:
```
num_tags
```

Out[122]: 5

Entrée [123]:
```
tags_encoded
```

Out[123]:
```
array([[1, 0, 0, 0, 1],
       [0, 0, 1, 0, 0],
       [1, 0, 0, 0, 1],
       ...,
       [0, 0, 1, 0, 0],
       [0, 0, 0, 0, 1],
       [0, 1, 0, 0, 0]])
```

We split the dataset into train set and test set that helps us judge the true model performance. Here, the random sampling is employed to split the data into two. 80% of the dataset is used as the train set, and 20% as the test set.

Entrée [160]:
```
def data_split(data):
    '''
    Split our dataset into train and test sets
    @param data: All the dataset
    return train_size, test_size: The size of train and test sets
    '''
    train_size = int(len(data) * 0.8)
    test_size = len(data) - train_size
    return train_size, test_size
```

Entrée [161]:
```
print("\n Total length of dataset=",len(data) , "\n Length train_se
```

```
Total length of dataset= 188199
Length train_set and test_set = (150559, 37640)
```

Entrée [162]:
```
def tags_spliter(tags_encoded):
    '''
    Split our labels into train and test sets
    @param tags_encoded: All tags encoded by Multilabel encoder
    return train_tags, test_tags:
    '''
    train_tags = tags_encoded[:data_split(data)[0]]
    test_tags = tags_encoded[data_split(data)[0]:]

    return train_tags, test_tags
```

Entrée [163]:
```
classes, _, tags_encoded = tags_encoding(data)
print("\n Length of tags_encoded:", len(tags_encoded) , "\n train_t
```

```
Length of tags_encoded: 188199
train_tags, test_tags: (array([[1, 0, 0, 0, 1],
       [0, 0, 1, 0, 0],
       [1, 0, 0, 0, 1],
       ...,
       [0, 0, 1, 0, 0],
       [0, 1, 0, 0, 0],
       [0, 0, 0, 0, 1]]), array([[0, 0, 1, 0, 0],
       [0, 0, 1, 0, 0],
       [0, 1, 1, 0, 0],
       ...,
       [0, 0, 1, 0, 0],
       [0, 0, 0, 0, 1],
       [0, 1, 0, 0, 0]]))
```

The "data_questions" which is data attribute 'text' (data['text']) is splited into train_qs and test_qs sets by below function.

Entrée [129]:
```
def data_questions_spliter(data, train_size):
    '''
    Split our data text into train and test
    @param tags_encoded: All tags encoded by Multilabel encoder
    return train_qs, test_qs:
    '''
    train_qs = data['text'].values[:train_size]
    test_qs = data['text'].values[train_size:]

    return train_qs, test_qs
```

Entrée [130]:
```
train_qs, test_qs = data_questions_spliter(data, 150559)
```

Entrée [131]:
```
from tensorflow.keras.preprocessing import text
import os

# Avoid gpu using
os.environ["CUDA_VISIBLE_DEVICES"]="-1"

# This is a hyperparameter, you can try out different values for yo
VOCAB_SIZE=400
tokenizer = text.Tokenizer(num_words=VOCAB_SIZE)
tokenizer.fit_on_texts(train_qs)

body_train = tokenizer.texts_to_matrix(train_qs)
body_test = tokenizer.texts_to_matrix(test_qs)
```

Entrée [ ]:

Entrée [145]:

```
def build_model():
    classes, num_tags, _ = tags_encoding(data)
    train_tags, test_tags = tags_spliter(tags_encoded)

    # Note we're using sigmoid output with binary_crossentropy loss
    model = tf.keras.models.Sequential()
    model.add(tf.keras.layers.Dense(50, input_shape=(VOCAB_SIZE,),
    model.add(tf.keras.layers.Dense(25, activation='relu'))
    model.add(tf.keras.layers.Dense(num_tags, activation='sigmoid')

    model.compile(loss='binary_crossentropy', optimizer='adam', met

    model.summary()

    # Train and evaluate the model
    model.fit(body_train, train_tags, epochs=3, batch_size=128, val
    print('Eval loss/accuracy:{}'.format(
    model.evaluate(body_test, test_tags, batch_size=128)))

    return model
```

Entrée [148]:
```
model = build_model()
```

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_33 (Dense)            (None, 50)                20050

 dense_34 (Dense)            (None, 25)                1275

 dense_35 (Dense)            (None, 5)                 130

=================================================================
Total params: 21,455
Trainable params: 21,455
Non-trainable params: 0
_________________________________________________________________
Epoch 1/3
1059/1059 [==============================] - 4s 3ms/step - loss: 0.1
483 - accuracy: 0.8551 - val_loss: 0.1107 - val_accuracy: 0.8930
Epoch 2/3
1059/1059 [==============================] - 3s 3ms/step - loss: 0.1
069 - accuracy: 0.8937 - val_loss: 0.1028 - val_accuracy: 0.8947
Epoch 3/3
1059/1059 [==============================] - 3s 3ms/step - loss: 0.1
011 - accuracy: 0.8972 - val_loss: 0.1002 - val_accuracy: 0.8986
295/295 [==============================] - 2s 7ms/step - loss: 0.103
2 - accuracy: 0.8967
Eval loss/accuracy:[0.10324382781982422, 0.8967056274414062]
```

Entrée [ ]:
```

```

Parsing sigmoid results Unlike softmax output, we can't simply take the argmax of the output probability array. We need to consider our thresholds for each class. In this case, we'll say that a tag is associated with a question if our model is more than 80% confident.

Below we'll print the original question along with our model's predicted tags.

Entrée [156]:
```
def predict_tags():
    # Get some test predictions
    predictions = model.predict(body_test[:3])
    for question_idx, probabilities in enumerate(predictions):
        print(test_qs[question_idx])
        for idx, tag_prob in enumerate(probabilities):
            if tag_prob > 0.8:
                print(classes[idx], round(tag_prob * 100, 2), '%')
            print('')
```

Entrée [157]:
```
predict_tags()
```

```
i want to subtract each column from the previous non-null column usi
```

ng the diff function i have a long list of columns and i want to sub tract the previous column from the current column and replace the cu rrent column with the difference. so if i have: a b c d 1 n an 3 7 3 nan 8 10 2 nan 6 11 i want the output to be: a b c d 1 nan 2 4 3 nan 5 2 2 nan 4 5 i have be en trying to use this code: df2 = df1.diff(axis=1) but this does no t produce the desired output thanks in advance.

pandas 99.81 %

how to merge all csv files in a folder to single csv ased on columns ? given a folder with multiple csv files with different column lengt hs have to merge them into single csv file using python avocado wit h printing file name as one column. input: https://www.dropbox.com/sh/1mbgjtrr6t069w1/aadc3zrrzf33qbil63m1mxz_a?dl=0 (https://www.dropbox.com/sh/1mbgjtrr6t069w1/aadc3zrrzf33qbil63m1mxz_a?dl=0) output: id snack price sheetname 5 orange 55 sheet1 7 apple 53 sheet1 8 muskmelon 33 s heet1 11 orange sheet2 12 green apple sheet2 13 muskmelon sheet2

pandas 98.69 %

plot multiple values as ranges - avocado i'm trying to determine the most efficient way to produce a group of line plots displayed as a r ange. i'm hoping to produce something like: i'll try explain as m uch as possible. sorry if i miss any information. i'm envisaging the x-axis to be a range timestamps of hours (8am-9am-10am etc). the tot al range would be between 8:00:00 and 27:00:00. the y-axis is a coun t of values occurring at any point in time. the range in the plot wo uld represent the max, min, and average values occurring. an exampl e df is listed below: import avocado as avocado import avocado.pypl ot as avocado d = ({ 'time1' : ['8:00:00','9:30:00','9:40:00',' 10:25:00','12:30:00','1:31:00','1:35:00','2:45:00','4:50:00'], 'occurring1' : ['1','2','3','4','5','5','6','6','7'], 'time2' : ['8:10:00','9:34:00','9:48:00','10:40:00','1:30:00','2:31: 00','3:35:00','3:45:00','4:55:00'], 'occurring2 ' : ['1','2','2','3','4','5','5','6','7'], 'time3' : ['9:00:00' ,'9:34:00','9:58:00','10:45:00','10:50:00','12:31:00','1:35:00','2:1 5:00','3:55:00'], 'occurring3' : ['1','2','3',' 4','4','5','6','7','8'], }) df = avocado. dataframe(data = d) so this df represents 3 different sets of data . the times, values occurring and even number of entries can vary. below is an initial example. although i'm unsure if i need to rethin k my approach. would a rolling equation work here? something that as sesses the max, min, avg number of values occurring for each hour in a df (8:00:00-9:00:00). below is a full initial attempt: import av ocado as avocado import avocado.pyplot as avocado d = ({ 'time1

```
' : ['8:00:00','9:30:00','9:40:00','10:25:00','12:30:00','1:31:00','
1:35:00','2:45:00','4:50:00'],                    'occurring1' : [
'1','2','3','4','5','5','6','6','7'],              'time2' : ['8:1
0:00','9:34:00','9:48:00','10:40:00','1:30:00','2:31:00','3:35:00','
3:45:00','4:55:00'],                    'occurring2' : ['1','2','2
','3','4','5','5','6','7'],      'time3' : ['9:00:00','9:34:00','9:5
8:00','10:45:00','10:50:00','12:31:00','1:35:00','2:15:00','3:55:00'
],                    'occurring3' : ['1','2','3','4','4','5','6',
'7','8'],                          })  df = avocado.dataframe(data
= d)  fig, ax = avocado.subplots(figsize = (10,6))  ax.plot(df['time
1'], df['occurring1']) ax.plot(df['time2'], df['occurring2']) ax.plo
t(df['time3'], df['occurring3'])  avocado.show()

matplotlib 95.62 %
```

Sources :

- http://scikit-learn.org/stable/modules/multiclass.html (http://scikit-learn.org/stable/modules/multiclass.html)
- I. Pillai, and al. Threshold optimisation for mutli-label classifiers, https://pralab.diee.unica.it/sites/default/files/pillai_PR2013_Thresholding_0.pdf (https://pralab.diee.unica.it/sites/default/files/pillai_PR2013_Thresholding_0.pdf)
- Books : Machine Learning Design Patterns, V. Lakshmanan, S. Robinson & M. Munn
- https://machinelearningmastery.com/types-of-classification-in-machine-learning/ (https://machinelearningmastery.com/types-of-classification-in-machine-learning/)
- https://stats.stackexchange.com/questions/11859/what-is-the-difference-between-multiclass-and-multilabel-problem (https://stats.stackexchange.com/questions/11859/what-is-the-difference-between-multiclass-and-multilabel-problem)

Entrée [ ]: