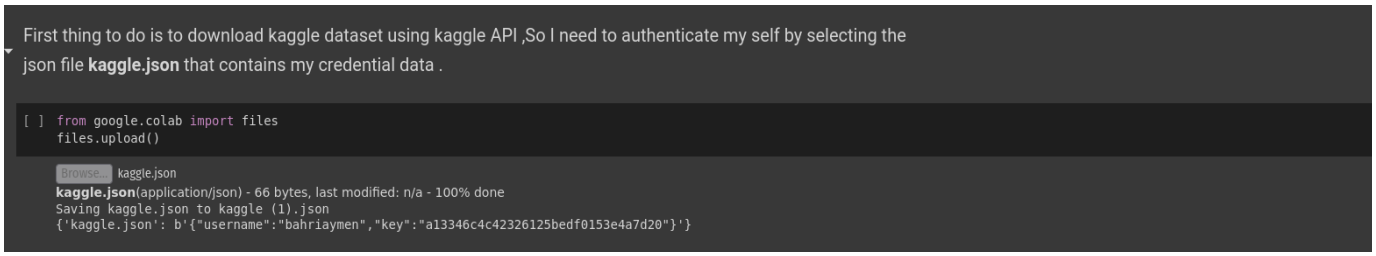
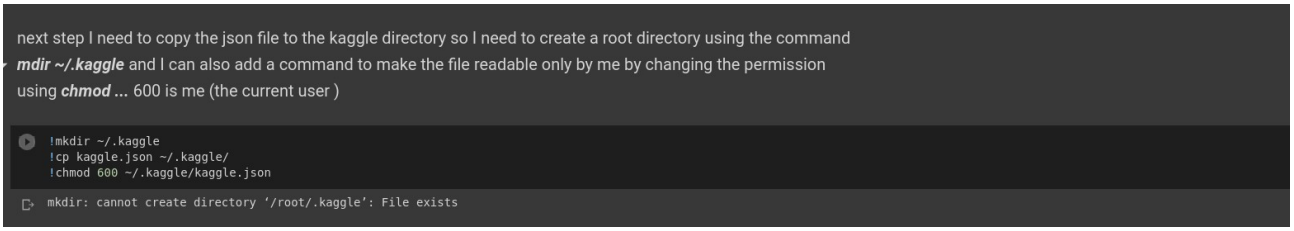
DL EXAM
Sentiment Analysis from text and Images

1st Step:

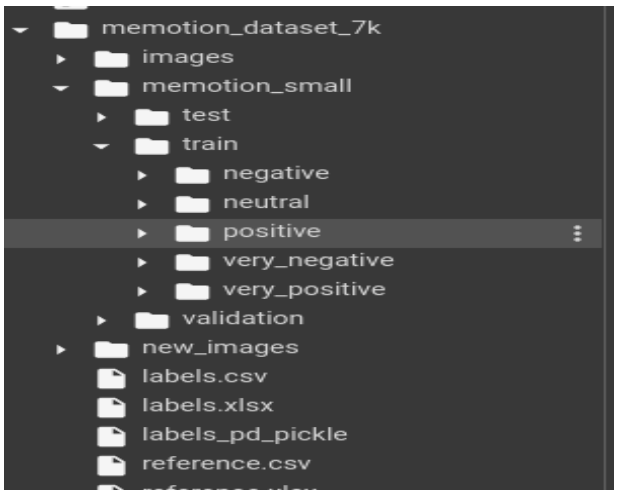
The First step is to load the data which is straightforward with the help of Kaggle API





2nd Step :

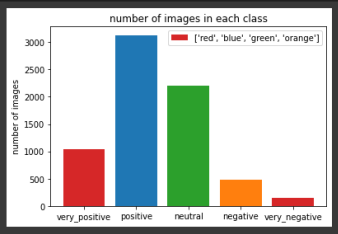
After Unzipping the data an important step (that we see it important is to split the images of each class rename them according to their class and move them to subdirectories of train ,validation and test ,this step is the most important step because it helps us use the dataset object later by calling image_dataset_from_directory ,which in turns will generate batches of data from each class ,reshaped to our desired shape ,i.e (300,300) and their corresponding labels



after runing the code cell we end up with those directories ,the dataset object used later will list all subdirectories lets say for training and returns an iterable object that we will feed it into our model

```
label_dict_1 = label_dict.copy()                        #copy old images name with their classes
label_dict = new_name_dict                               # changes the  names dictionnary with the new names
```

```
# plot the statistics of each class
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

classes = list(label_dict.keys())
counts = [len(category) for category in label_dict.values()]
bar_labels = ['red', 'blue', 'green', 'orange']
bar_colors = ['tab:red', 'tab:blue', 'tab:green', 'tab:orange']

ax.bar(classes, counts, label=bar_labels, color=bar_colors)

ax.set_ylabel('number of images')
ax.set_title('number of images in each class')
ax.legend()

plt.show()
```




```
counts
```

the data distribution is very unbalanced ,classes negative and very negative contains a small number of samples ,so the model will be not able to generalize on those tow classes and will certainly overfits on the others .

3rd Step :

We can now prepare a basic model architecture ,since we are dealing with images ,CNN models will be the best choice because as opposed to dense layers ,they learn features from their position in images and can recognize them anywhere later when testing the model

```
# now we will use keras functional API to create a CNN model at first,using functional API will allow us to alter the model later by adding layers and enables me to use multi_inp

from tensorflow import keras
from keras import layers
inputs =keras.Input(shape = (300,300,3))                              # input shape (300,300) was chosen at random
x = layers.Rescaling(1./255)(inputs)                                  # rescaling inputs by deviding each pixel by 255
x = layers.Conv2D(filters = 32,kernel_size=3,activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)                               #downsampling the input ,deviding the input shape by 2
x = layers.Conv2D(filters = 64,kernel_size = 3,activation = "relu")(x)
x = layers.MaxPooling2D(pool_size = 2)(x)
x = layers.Conv2D(filters=128,kernel_size = 3,activation ="relu" )(x)
x = layers.MaxPooling2D(pool_size = 2)(x)
x = layers.Conv2D(filters=256,kernel_size = 3,activation ="relu" )(x)
x = layers.MaxPooling2D(pool_size = 2)(x)
x = layers.Conv2D(filters=256,kernel_size = 3,activation ="relu" )(x)
x = layers.Flatten()(x)

outputs = layers.Dense(5,activation = "softmax")(x)
model = keras.Model(inputs = inputs,outputs = outputs )

model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 300, 300, 3)]     0

 rescaling (Rescaling)       (None, 300, 300, 3)       0

 conv2d (Conv2D)             (None, 298, 298, 32)      896

 max_pooling2d (MaxPooling2D  (None, 149, 149, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 147, 147, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 73, 73, 64)       0
```

The MaxPooling layers used will downsample the feature maps by halving them(deviding by 2 the dimension of their outputs it consists of extracting windows from the input feature maps(output of convolution layers) and outputting the max value of each channel.

4th Step :

We need to split our data between training ,test,and validation
the number of samples chosen for each call is much random because the negative and very negative classes contains very few examples ,we tried to split them by using most of them on training (200 negative ,100 very negative) and the rest to test and validation

```
!rm -r /content/memotion_dataset_7k/memotion_small   #Just a terminal command to remove recursively my small dataset if its already created
import os,shutil,pathlib

original_dir = pathlib.Path("/content/memotion_dataset_7k/new_images")
new_base_dir = pathlib.Path("/content/memotion_dataset_7k/memotion_small")

label_dict = new_name_dict

#since data is not split equivalently between classes we will split them manually not following any  rule between validation , train and test
l_train = {}
train_positive = label_dict["positive"][:2000]
train_very_positive = label_dict["very_positive"][:600]
train_neutral = label_dict["neutral"][:1000]
train_negative = label_dict["negative"][:200]
train_very_negative = label_dict["very_negative"][:100]
l_train["positive"],l_train["very_positive"],l_train["neutral"],l_train["negative"],l_train["very_negative"]= train_positive,train_very_positive, train_neutral, train_negative, t



l_test ={}
test_positive = label_dict["positive"][2000:2500]
test_very_positive = label_dict["very_positive"][600:800]
test_neutral = label_dict["neutral"][1000:1800]
test_negative = label_dict["negative"][200:400]
test_very_negative = label_dict["very_negative"][100:125]

l_test["positive"],l_test["very_positive"],l_test["neutral"],l_test["negative"],l_test["very_negative"]= test_positive,test_very_positive, test_neutral, test_negative, test_very_


l_validation = {}
validation_positive = label_dict["positive"][2500:]
validation_very_positive = label_dict["very_positive"][800:]
validation_neutral = label_dict["neutral"][1800:]
validation_negative = label_dict["negative"][400:]
validation_very_negative = label_dict["very_negative"][125:]

l_validation["positive"],l_validation["very_positive"],l_validation["neutral"],l_validation["negative"],l_validation["very_negative"]= validation_positive,validation_very_positiv
```

5th Step :

Since our directories are ready and each one contains the correct data ,we can now use the dataset object by using the **image_from_dataset_directory** class as discussed before

for the remaining steps we tried to explain as much as we can in the code ,but due to the shortage of time we will try to answer as much questions as we can in the QA session