from toxicity\_ml\_pipeline.load\_model import reload\_model\_weights

from toxicity\_ml\_pipeline.utils.helpers import load\_inference\_func, upload\_model

import numpy as np

import tensorflow as tf

def score(language, df, gcs\_model\_path, batch\_size=64, text\_col="text", kw="", \*\*kwargs):

if language != "en":

raise NotImplementedError(

"Data preprocessing not implemented here, needs to be added for i18n models"

)

model\_folder = upload\_model(full\_gcs\_model\_path=gcs\_model\_path)

try:

inference\_func = load\_inference\_func(model\_folder)

except OSError:

model = reload\_model\_weights(model\_folder, language, \*\*kwargs)

preds = model.predict(x=df[text\_col], batch\_size=batch\_size)

if type(preds) != list:

if len(preds.shape)> 1 and preds.shape[1] > 1:

if 'num\_classes' in kwargs and kwargs['num\_classes'] > 1:

raise NotImplementedError

preds = np.mean(preds, 1)

df[f"prediction\_{kw}"] = preds

else:

if len(preds) > 2:

raise NotImplementedError

for preds\_arr in preds:

if preds\_arr.shape[1] == 1:

df[f"prediction\_{kw}\_target"] = preds\_arr

else:

for ind in range(preds\_arr.shape[1]):

df[f"prediction\_{kw}\_content\_{ind}"] = preds\_arr[:, ind]

return df

else:

return \_get\_score(inference\_func, df, kw=kw, batch\_size=batch\_size, text\_col=text\_col)

def \_get\_score(inference\_func, df, text\_col="text", kw="", batch\_size=64):

score\_col = f"prediction\_{kw}"

beginning = 0

end = df.shape[0]

predictions = np.zeros(shape=end, dtype=float)

while beginning < end:

mb = df[text\_col].values[beginning : beginning + batch\_size]

res = inference\_func(input\_1=tf.constant(mb))

predictions[beginning : beginning + batch\_size] = list(res.values())[0].numpy()[:, 0]

beginning += batch\_size

df[score\_col] = predictions

return df