TextClassification2_AXP2000075

April 24, 2023

Alan Perez | AXP200075 | Text Classification 2

The dataset I used was the Twitter Sent Analysis, should be able to predict the sentiment of tweets

https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis

[ ]:
```
import pandas as pd
import seaborn as sb
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras import layers, models, preprocessing, datasets
import numpy as np
from sklearn.preprocessing import LabelEncoder
import pickle
```

[ ]:
```
#load data

# target column 2 which is sentiment and 3 which is tweet content
data = pd.read_csv('twitter.csv',usecols=[2,3], names=['sentiment',␣
 ↪'tweet_content'],encoding='latin-1')

# check for NaN values
print(data.isna().sum())
# check if there  are unexpected types

print(data.applymap(type))
# replace NaN with empty str
data.fillna('', inplace=True)

# convert columns to string type
data['sentiment'] = data['sentiment'].astype(str)
data['tweet_content'] = data['tweet_content'].astype(str)
data = data.astype(str)
display(data)

# check the data types of each column
print('data dtypes: ', data.dtypes)

print('shape rows column: ', data.shape)
```

```
print('head : ', data.head)
print('tail: ', data.tail)

# check for NaN values
print(data.isna().sum())
print(data.columns)
```

```
sentiment          0
tweet_content    686
dtype: int64
           sentiment  tweet_content
0      <class 'str'>  <class 'str'>
1      <class 'str'>  <class 'str'>
2      <class 'str'>  <class 'str'>
3      <class 'str'>  <class 'str'>
4      <class 'str'>  <class 'str'>
…                …              …
74677  <class 'str'>  <class 'str'>
74678  <class 'str'>  <class 'str'>
74679  <class 'str'>  <class 'str'>
74680  <class 'str'>  <class 'str'>
74681  <class 'str'>  <class 'str'>

[74682 rows x 2 columns]
```

```
      sentiment                                      tweet_content
0      Positive  im getting on borderlands and i will murder yo…
1      Positive  I am coming to the borders and I will kill you…
2      Positive  im getting on borderlands and i will kill you …
3      Positive  im coming on borderlands and i will murder you…
4      Positive  im getting on borderlands 2 and i will murder …
…         …                                                   …
74677  Positive  Just realized that the Windows partition of my…
74678  Positive  Just realized that my Mac window partition is …
74679  Positive  Just realized the windows partition of my Mac …
74680  Positive  Just realized between the windows partition of…
74681  Positive  Just like the windows partition of my Mac is l…

[74682 rows x 2 columns]
```

```
data dtypes:  sentiment        object
tweet_content    object
dtype: object
shape rows column:  (74682, 2)
head :  <bound method NDFrame.head of       sentiment
tweet_content
0      Positive  im getting on borderlands and i will murder yo…
1      Positive  I am coming to the borders and I will kill you…
```

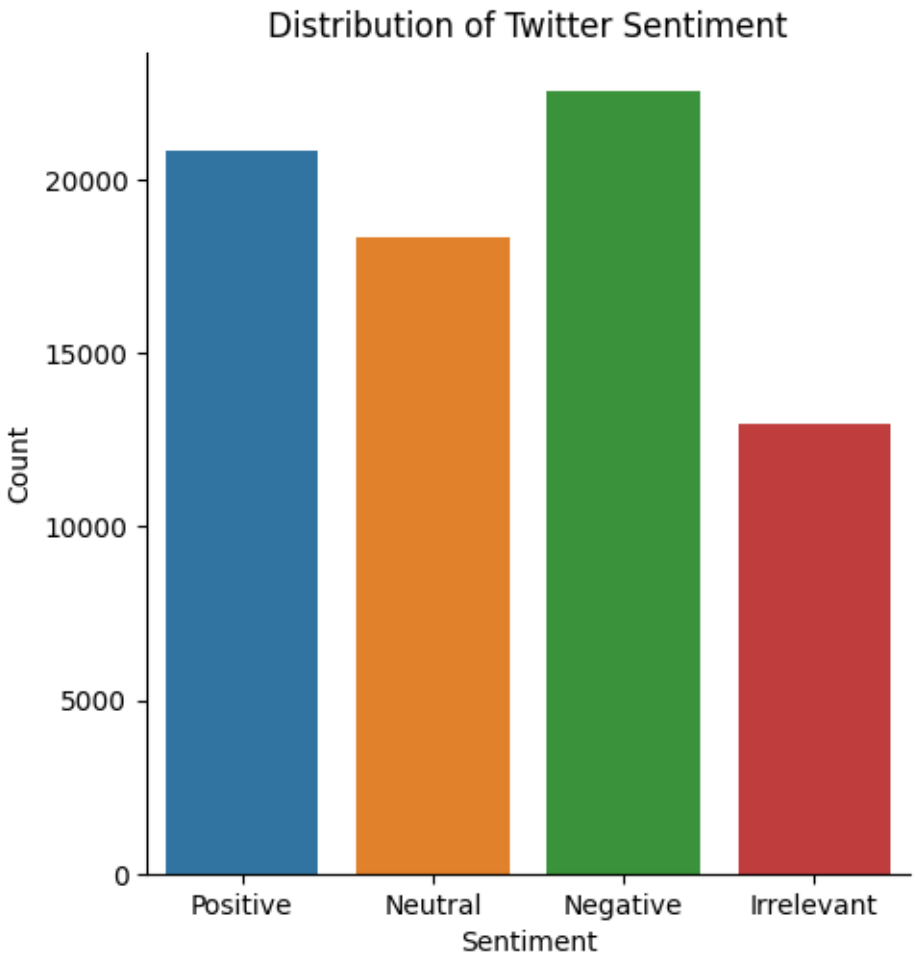
```
2      Positive  im getting on borderlands and i will kill you …
3      Positive  im coming on borderlands and i will murder you…
4      Positive  im getting on borderlands 2 and i will murder …
…         …                                                  …
74677  Positive  Just realized that the Windows partition of my…
74678  Positive  Just realized that my Mac window partition is …
74679  Positive  Just realized the windows partition of my Mac …
74680  Positive  Just realized between the windows partition of…
74681  Positive  Just like the windows partition of my Mac is l…

[74682 rows x 2 columns]>
tail:  <bound method NDFrame.tail of       sentiment
tweet_content
0      Positive  im getting on borderlands and i will murder yo…
1      Positive  I am coming to the borders and I will kill you…
2      Positive  im getting on borderlands and i will kill you …
3      Positive  im coming on borderlands and i will murder you…
4      Positive  im getting on borderlands 2 and i will murder …
…         …                                                  …
74677  Positive  Just realized that the Windows partition of my…
74678  Positive  Just realized that my Mac window partition is …
74679  Positive  Just realized the windows partition of my Mac …
74680  Positive  Just realized between the windows partition of…
74681  Positive  Just like the windows partition of my Mac is l…

[74682 rows x 2 columns]>
sentiment        0
tweet_content    0
dtype: int64
Index(['sentiment', 'tweet_content'], dtype='object')
```

1 Graph Visualization

[ ]:
```
import matplotlib.pyplot as plt
sb.catplot(x="sentiment", kind="count", data=data)
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.title('Distribution of Twitter Sentiment')
plt.show()
```



2 Sequential Model

Divide and Train sets

[ ]:
```
#set seed for reproducibility
np.random.seed(1234)
i = np.random.rand(len(data)) < 0.8
train = data[i]
test = data[~i]
print("train data size: ", train.shape)
print("test data size: ", test.shape)
```

```
train data size:  (59733, 2)
test data size:  (14949, 2)
```

[ ]:
```
# x and y

num_labels = 2
vocab_size = 10000
batch_size = 100

# fit the tokenizer on the training data
tokenizer = Tokenizer(num_words=vocab_size)
tokenizer.fit_on_texts(train.tweet_content)

x_train = tokenizer.texts_to_matrix(train.tweet_content, mode='tfidf')
x_test = tokenizer.texts_to_matrix(test.tweet_content, mode='tfidf')

encoder = LabelEncoder()
encoder.fit(train.sentiment)
y_train = encoder.transform(train.sentiment)
y_test = encoder.transform(test.sentiment)

# check shape
print("train shapes:", x_train.shape, y_train.shape)
print("test shapes:", x_test.shape, y_test.shape)
print("test first five labels:", y_test[:5])
```

```
train shapes: (59733, 10000) (59733,)
test shapes: (14949, 10000) (14949,)
test first five labels: [3 3 3 3 1]
```

[ ]:
```
# fit model

model = models.Sequential()
model.add(layers.Dense(32, input_dim=vocab_size, kernel_initializer='normal',␣
 ↪activation='relu'))
model.add(layers.Dense(1, kernel_initializer='normal', activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
                    batch_size=batch_size,
                    epochs=30,
                    verbose=1,
                    validation_split=0.1)
```

```
Epoch 1/30
538/538 [==============================] - 8s 14ms/step - loss: -109.3246 -
accuracy: 0.3065 - val_loss: -449.5832 - val_accuracy: 0.2680
```

```
Epoch 2/30
538/538 [==============================] - 6s 11ms/step - loss: -876.6799 -
accuracy: 0.3082 - val_loss: -1886.0970 - val_accuracy: 0.2680
Epoch 3/30
538/538 [==============================] - 7s 12ms/step - loss: -2355.4119 -
accuracy: 0.3083 - val_loss: -4105.3730 - val_accuracy: 0.2680
Epoch 4/30
538/538 [==============================] - 6s 11ms/step - loss: -4407.6631 -
accuracy: 0.3083 - val_loss: -6977.9624 - val_accuracy: 0.2680
Epoch 5/30
538/538 [==============================] - 7s 13ms/step - loss: -6961.2695 -
accuracy: 0.3083 - val_loss: -10438.0020 - val_accuracy: 0.2680
Epoch 6/30
538/538 [==============================] - 6s 11ms/step - loss: -9977.5928 -
accuracy: 0.3084 - val_loss: -14451.6611 - val_accuracy: 0.2680
Epoch 7/30
538/538 [==============================] - 7s 13ms/step - loss: -13413.3643 -
accuracy: 0.3083 - val_loss: -18959.2676 - val_accuracy: 0.2680
Epoch 8/30
538/538 [==============================] - 6s 11ms/step - loss: -17236.8574 -
accuracy: 0.3083 - val_loss: -23941.7578 - val_accuracy: 0.2680
Epoch 9/30
538/538 [==============================] - 7s 12ms/step - loss: -21435.2891 -
accuracy: 0.3084 - val_loss: -29377.4941 - val_accuracy: 0.2680
Epoch 10/30
538/538 [==============================] - 6s 11ms/step - loss: -26000.1484 -
accuracy: 0.3084 - val_loss: -35261.9766 - val_accuracy: 0.2680
Epoch 11/30
538/538 [==============================] - 7s 13ms/step - loss: -30924.4277 -
accuracy: 0.3084 - val_loss: -41599.2227 - val_accuracy: 0.2680
Epoch 12/30
538/538 [==============================] - 6s 11ms/step - loss: -36205.3086 -
accuracy: 0.3084 - val_loss: -48359.4609 - val_accuracy: 0.2680
Epoch 13/30
538/538 [==============================] - 6s 12ms/step - loss: -41826.7266 -
accuracy: 0.3083 - val_loss: -55545.1172 - val_accuracy: 0.2680
Epoch 14/30
538/538 [==============================] - 6s 11ms/step - loss: -47789.8047 -
accuracy: 0.3084 - val_loss: -63155.8125 - val_accuracy: 0.2680
Epoch 15/30
538/538 [==============================] - 6s 12ms/step - loss: -54094.8125 -
accuracy: 0.3084 - val_loss: -71174.1641 - val_accuracy: 0.2680
Epoch 16/30
538/538 [==============================] - 6s 12ms/step - loss: -60740.8711 -
accuracy: 0.3084 - val_loss: -79627.2891 - val_accuracy: 0.2680
Epoch 17/30
538/538 [==============================] - 6s 12ms/step - loss: -67731.2344 -
accuracy: 0.3084 - val_loss: -88518.8281 - val_accuracy: 0.2680
```

```
Epoch 18/30
538/538 [==============================] - 6s 12ms/step - loss: -75061.1406 -
accuracy: 0.3084 - val_loss: -97813.7344 - val_accuracy: 0.2680
Epoch 19/30
538/538 [==============================] - 6s 12ms/step - loss: -82719.4453 -
accuracy: 0.3084 - val_loss: -107522.8047 - val_accuracy: 0.2680
Epoch 20/30
538/538 [==============================] - 6s 12ms/step - loss: -90723.1953 -
accuracy: 0.3084 - val_loss: -117653.9297 - val_accuracy: 0.2680
Epoch 21/30
538/538 [==============================] - 6s 11ms/step - loss: -99049.9219 -
accuracy: 0.3084 - val_loss: -128173.4453 - val_accuracy: 0.2680
Epoch 22/30
538/538 [==============================] - 7s 12ms/step - loss: -107710.2500 -
accuracy: 0.3084 - val_loss: -139118.8750 - val_accuracy: 0.2680
Epoch 23/30
538/538 [==============================] - 6s 11ms/step - loss: -116709.6406 -
accuracy: 0.3084 - val_loss: -150480.5312 - val_accuracy: 0.2680
Epoch 24/30
538/538 [==============================] - 7s 13ms/step - loss: -126044.0391 -
accuracy: 0.3084 - val_loss: -162282.6250 - val_accuracy: 0.2680
Epoch 25/30
538/538 [==============================] - 6s 11ms/step - loss: -135717.3750 -
accuracy: 0.3084 - val_loss: -174498.9062 - val_accuracy: 0.2680
Epoch 26/30
538/538 [==============================] - 7s 13ms/step - loss: -145708.8906 -
accuracy: 0.3083 - val_loss: -187067.1406 - val_accuracy: 0.2680
Epoch 27/30
538/538 [==============================] - 6s 11ms/step - loss: -156031.1719 -
accuracy: 0.3084 - val_loss: -200087.9062 - val_accuracy: 0.2680
Epoch 28/30
538/538 [==============================] - 7s 12ms/step - loss: -166673.9062 -
accuracy: 0.3084 - val_loss: -213470.8125 - val_accuracy: 0.2680
Epoch 29/30
538/538 [==============================] - 6s 11ms/step - loss: -177626.9844 -
accuracy: 0.3084 - val_loss: -227287.2188 - val_accuracy: 0.2680
Epoch 30/30
538/538 [==============================] - 7s 12ms/step - loss: -188930.3750 -
accuracy: 0.3084 - val_loss: -241496.3125 - val_accuracy: 0.2680
```

[ ]:
```
# evaluate

score = model.evaluate(x_test, y_test, batch_size=batch_size, verbose=1)
print('Accuracy: ', score[1])

print(score)
```

```
150/150 [==============================] - 1s 4ms/step - loss: -193450.0000 -
```

```
accuracy: 0.3028
Accuracy:  0.3028296232223511
[-193450.0, 0.3028296232223511]
```

[ ]:
```
# get predictions so we can calculate more metrics
pred = model.predict(x_test)
pred_labels = [1 if p>0.5 else 0 for p in pred]
```

```
468/468 [==============================] - 1s 2ms/step
```

[ ]:
```
pred[:10]
```

[ ]:
```
array([[1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.]], dtype=float32)
```

[ ]:
```
pred_labels[:10]
```

[ ]:
```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

[ ]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score
print('accuracy score: ', accuracy_score(y_test, pred_labels))
print('precision score: ', precision_score(y_test, pred_labels,␣
 ↪average='weighted'))
print('recall score: ', recall_score(y_test, pred_labels, average='weighted'))
print('f1 score: ', f1_score(y_test, pred_labels, average='weighted'))
```

```
accuracy score:  0.3028296207104154
precision score:  0.26307169291211835
recall score:  0.3028296207104154
f1 score:  0.14376181790687412

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

3 RNN

[ ]:
```
max_features = 10000
maxlen = 500
batch_size = 32
# pad the data to maxlen
train_data = preprocessing.sequence.pad_sequences(x_train, maxlen=maxlen)
test_data = preprocessing.sequence.pad_sequences(x_test, maxlen=maxlen)
```

[ ]:
```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense (Dense)                (None, 32)                320032

 dense_1 (Dense)              (None, 1)                 33

=================================================================
Total params: 320,065
Trainable params: 320,065
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
from keras import preprocessing
from keras.models import Sequential
from keras.layers import Dense, Flatten, Embedding
# Define the model
model = Sequential()
model.add(Embedding(max_features, 32, input_length=maxlen))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

[ ]:
```
# compile
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

[ ]:
```
# train

history = model.fit(train_data,
                    y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```

```
Epoch 1/10
374/374 [==============================] - 12s 29ms/step - loss: -2078.7622 -
accuracy: 0.2895 - val_loss: -7387.2837 - val_accuracy: 0.3528
Epoch 2/10
374/374 [==============================] - 11s 30ms/step - loss: -21262.9746 -
accuracy: 0.2895 - val_loss: -44272.4336 - val_accuracy: 0.3528
Epoch 3/10
374/374 [==============================] - 11s 30ms/step - loss: -79921.0703 -
accuracy: 0.2895 - val_loss: -133940.6406 - val_accuracy: 0.3528
Epoch 4/10
374/374 [==============================] - 11s 30ms/step - loss: -199047.2344 -
accuracy: 0.2895 - val_loss: -298529.0312 - val_accuracy: 0.3528
Epoch 5/10
374/374 [==============================] - 14s 37ms/step - loss: -400794.6250 -
accuracy: 0.2895 - val_loss: -560864.8750 - val_accuracy: 0.3528
Epoch 6/10
374/374 [==============================] - 11s 30ms/step - loss: -706194.1875 -
accuracy: 0.2895 - val_loss: -943666.4375 - val_accuracy: 0.3528
Epoch 7/10
374/374 [==============================] - 11s 30ms/step - loss: -1137221.8750 -
accuracy: 0.2895 - val_loss: -1467757.5000 - val_accuracy: 0.3528
Epoch 8/10
374/374 [==============================] - 11s 30ms/step - loss: -1711982.0000 -
accuracy: 0.2895 - val_loss: -2158455.5000 - val_accuracy: 0.3528
Epoch 9/10
374/374 [==============================] - 11s 28ms/step - loss: -2455374.2500 -
accuracy: 0.2895 - val_loss: -3033894.5000 - val_accuracy: 0.3528
Epoch 10/10
374/374 [==============================] - 12s 31ms/step - loss: -3389050.5000 -
accuracy: 0.2895 - val_loss: -4119448.0000 - val_accuracy: 0.3528
```

[ ]:
```
from sklearn.metrics import classification_report

pred = model.predict(test_data)
pred = [1.0 if p>= 0.5 else 0.0 for p in pred]
print(classification_report(y_test, pred))
```

```
468/468 [==============================] - 2s 3ms/step
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      2653
           1       0.30      1.00      0.46      4493
           2       0.00      0.00      0.00      3626
           3       0.00      0.00      0.00      4177

    accuracy                           0.30     14949
   macro avg       0.08      0.25      0.12     14949
weighted avg       0.09      0.30      0.14     14949
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

4 LTSM

[ ]:
```
# build a model with LSTM
model = models.Sequential()
model.add(layers.Embedding(max_features, 32))
model.add(layers.LSTM(32))
model.add(layers.Dense(1, activation='sigmoid'))



model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, None, 32)          320000

 lstm (LSTM)                 (None, 32)                8320

 dense_4 (Dense)             (None, 1)                 33

=================================================================
Total params: 328,353
Trainable params: 328,353
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
# compile
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# train

history = model.fit(train_data,
                    y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```

```
Epoch 1/10
374/374 [==============================] - 142s 374ms/step - loss: -7.8839 -
accuracy: 0.2895 - val_loss: -12.5662 - val_accuracy: 0.3528
Epoch 2/10
374/374 [==============================] - 136s 362ms/step - loss: -15.7548 -
accuracy: 0.2895 - val_loss: -20.5482 - val_accuracy: 0.3528
Epoch 3/10
374/374 [==============================] - 145s 388ms/step - loss: -23.3606 -
accuracy: 0.2895 - val_loss: -28.5352 - val_accuracy: 0.3528
Epoch 4/10
374/374 [==============================] - 135s 362ms/step - loss: -30.9649 -
accuracy: 0.2895 - val_loss: -36.5198 - val_accuracy: 0.3528
Epoch 5/10
374/374 [==============================] - 139s 371ms/step - loss: -38.5526 -
accuracy: 0.2895 - val_loss: -44.4865 - val_accuracy: 0.3528
Epoch 6/10
374/374 [==============================] - 135s 362ms/step - loss: -46.1822 -
accuracy: 0.2895 - val_loss: -52.5017 - val_accuracy: 0.3528
Epoch 7/10
374/374 [==============================] - 134s 359ms/step - loss: -53.7950 -
accuracy: 0.2895 - val_loss: -60.4914 - val_accuracy: 0.3528
Epoch 8/10
374/374 [==============================] - 136s 363ms/step - loss: -61.4076 -
accuracy: 0.2895 - val_loss: -68.4705 - val_accuracy: 0.3528
Epoch 9/10
374/374 [==============================] - 138s 369ms/step - loss: -69.0481 -
accuracy: 0.2895 - val_loss: -76.4726 - val_accuracy: 0.3528
Epoch 10/10
374/374 [==============================] - 135s 360ms/step - loss: -76.6617 -
accuracy: 0.2895 - val_loss: -84.4653 - val_accuracy: 0.3528
```

[ ]:
```
pred = model.predict(test_data)
pred = [1.0 if p>= 0.5 else 0.0 for p in pred]
print(classification_report(y_test, pred))
```

```
468/468 [==============================] - 23s 48ms/step
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      2653
           1       0.30      1.00      0.46      4493
           2       0.00      0.00      0.00      3626
           3       0.00      0.00      0.00      4177

    accuracy                           0.30     14949
   macro avg       0.08      0.25      0.12     14949
weighted avg       0.09      0.30      0.14     14949

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

5 LSTM - Attempt #2

[ ]:
```
# build a model with LSTM
model = models.Sequential()
model.add(layers.Embedding(max_features, 32))
model.add(layers.LSTM(32))
model.add(layers.Dense(1, activation='sigmoid'))



model.summary()
```

[ ]:
```
# compile
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

[ ]:
```
# train

history = model.fit(x_train,
                    y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```

[ ]:
```
pred = model.predict(x_test, y_test)
pred = [1.0 if p>= 0.5 else 0.0 for p in pred]
print(classification_report(y_train, pred))
```

6 GRU

[ ]:
```
model = models.Sequential()
model.add(layers.Embedding(max_features, 32))
model.add(layers.GRU(32))
model.add(layers.Dense(1, activation='sigmoid'))
```

[ ]:
```
# compile
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

[ ]:
```
# train

history = model.fit(train_data,
                    y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```

```
Epoch 1/10
374/374 [==============================] - 134s 350ms/step - loss: -7.8998 -
accuracy: 0.2895 - val_loss: -12.7450 - val_accuracy: 0.3528
Epoch 2/10
374/374 [==============================] - 133s 356ms/step - loss: -15.9390 -
accuracy: 0.2895 - val_loss: -20.7263 - val_accuracy: 0.3528
Epoch 3/10
374/374 [==============================] - 129s 344ms/step - loss: -23.5711 -
accuracy: 0.2895 - val_loss: -28.7289 - val_accuracy: 0.3528
Epoch 4/10
374/374 [==============================] - 137s 366ms/step - loss: -31.1776 -
accuracy: 0.2895 - val_loss: -36.7223 - val_accuracy: 0.3528
Epoch 5/10
374/374 [==============================] - 132s 354ms/step - loss: -38.7615 -
```

```
accuracy: 0.2895 - val_loss: -44.7269 - val_accuracy: 0.3528
Epoch 6/10
374/374 [==============================] - 131s 351ms/step - loss: -46.3505 -
accuracy: 0.2895 - val_loss: -52.7018 - val_accuracy: 0.3528
Epoch 7/10
374/374 [==============================] - 132s 354ms/step - loss: -54.0266 -
accuracy: 0.2895 - val_loss: -60.6957 - val_accuracy: 0.3528
Epoch 8/10
374/374 [==============================] - 128s 342ms/step - loss: -61.6003 -
accuracy: 0.2895 - val_loss: -68.6794 - val_accuracy: 0.3528
Epoch 9/10
374/374 [==============================] - 131s 351ms/step - loss: -69.2336 -
accuracy: 0.2895 - val_loss: -76.6718 - val_accuracy: 0.3528
Epoch 10/10
374/374 [==============================] - 137s 367ms/step - loss: -76.8438 -
accuracy: 0.2895 - val_loss: -84.6562 - val_accuracy: 0.3528
```

[ ]:
```
pred = model.predict(test_data)
pred = [1.0 if p>= 0.5 else 0.0 for p in pred]
print(classification_report(y_test, pred))
```

```
468/468 [==============================] - 26s 54ms/step
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      2653
           1       0.30      1.00      0.46      4493
           2       0.00      0.00      0.00      3626
           3       0.00      0.00      0.00      4177

    accuracy                           0.30     14949
   macro avg       0.08      0.25      0.12     14949
weighted avg       0.09      0.30      0.14     14949


/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
```

```
  _warn_prf(average, modifier, msg_start, len(result))
```

7 Analysis

I've noticed that the accuracy when using different approaches the results where practicially the same when scoring the accurracy. I believe I need to experiemnt a lot more with the model. The sequential model did the best out of all of them, but the low accuracy may be due to the dataset

[ ]: