BERT with Keras

```
import pandas as pd
```

```
df_balanced = pd.read_csv("dataframe_edit.tsv", sep = '\t')
```

```
df.head()
```

	text	hyperpartisan
0	<p>Money (<a href="https://farm8.static.flick...	1
1	<p>Donald Trump ran on many braggadocios and l...	1
2	<p>In response to Joyce Newman&#8217;s recent ...	1
3	<p>After Colin Kaepernick rightly chose to kne...	1
4	<p>Almost a half-century ago, in 1968, the Uni...	0

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df_balanced['text'],df_balancec
```

```
!pip install tensorflow_text
```

```
import tensorflow as tf
import tensorflow_hub as hub
import tensorflow_text as text
```

```
bert_preprocess = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_prep
bert_encoder = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-
# DistilBERT
#https://tfhub.dev/jeongukjae/distilbert_en_uncased_L-6_H-768_A-12/1
#


# Bert layers
text_input = tf.keras.layers.Input(shape=(), dtype=tf.string, name='text')
preprocessed_text = bert_preprocess(text_input)
outputs = bert_encoder(preprocessed_text)

# Neural network layers
l = tf.keras.layers.Dropout(0.1, name="dropout")(outputs['pooled_output'])
l = tf.keras.layers.Dense(1, activation='sigmoid', name="output")(l)

# Use inputs and outputs to construct a final model
model = tf.keras.Model(inputs=[text_input], outputs = [l])


METRICS = [
      tf.keras.metrics.BinaryAccuracy(name='accuracy'),
      tf.keras.metrics.Precision(name='precision'),
      tf.keras.metrics.Recall(name='recall')
]

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=METRICS)



model.fit(X_train, y_train, epochs=5)
```

```
    Epoch 1/5
    16/16 [==============================] - 361s 19s/step - loss: 0.7381 - accura
    Epoch 2/5
    16/16 [==============================] - 267s 17s/step - loss: 0.6775 - accura
    Epoch 3/5
    16/16 [==============================] - 260s 16s/step - loss: 0.6556 - accura
    Epoch 4/5
    16/16 [==============================] - 267s 17s/step - loss: 0.6655 - accura
    Epoch 5/5
    16/16 [==============================] - 258s 16s/step - loss: 0.6335 - accura
    <keras.callbacks.History at 0x7fd703a5e340>
```

```
model.evaluate(X_test, y_test)
```

```
y_predicted = model.predict(X_test)
y_predicted = y_predicted.flatten()

import numpy as np

y_predicted = np.where(y_predicted > 0.5, 1, 0)
y_predicted
```

```
    6/6 [==============================] - 91s 15s/step
    array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
           0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0,
           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
           0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
           1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
           0, 0, 0, 0, 0, 0, 0, 0])
```

```
from sklearn.metrics import confusion_matrix, classification_report

cm = confusion_matrix(y_test, y_predicted)
cm
```

```
from matplotlib import pyplot as plt
import seaborn as sn
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
print(classification_report(y_test, y_predicted))
```

GPT-3 Zero shot

```
!pip install openai

    Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-w
    Collecting openai
      Downloading openai-0.27.2-py3-none-any.whl (70 kB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 70.1/70.1 KB 3.5 MB/s eta 0:00:0
    Requirement already satisfied: requests>=2.20 in /usr/local/lib/python3.9/dist
    Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packages
    Requirement already satisfied: aiohttp in /usr/local/lib/python3.9/dist-packag
    Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/
    Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-p
    Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3
    Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/pyt
    Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.9/dist
    Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.9/dist-
    Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/p
    Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.9
    Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.9/di
    Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.9/d
    Installing collected packages: openai
    Successfully installed openai-0.27.2
```

```
import os
import openai
OPENAI_API_KEY = "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"
openai.api_key = OPENAI_API_KEY

response = openai.Completion.create(
  model="text-davinci-003",
  prompt="I am a highly intelligent question answering bot. If you ask me a questio
  temperature=0,
  max_tokens=100,
  top_p=1,
  frequency_penalty=0.0,
  presence_penalty=0.0,
  stop=["\n"]
)
```

```
import pandas as pd
ds = pd.read_csv("dataframe_edit.tsv", sep = '\t')
```

```
ds_65 = ds.tail(65)
```

```
y_true = ds_65["hyperpartisan"].tolist()
```

```
y_true
```

```
ds_65["text"].head()
```

```
    580    <p>The FBI is advising people to hang up if th...
    581    <p>A woman is facing charges as part an invest...
    582    <p>Usually, the person with the most informati...
    583    <p>When the removal of a sign becomes a sign o...
    584    <p>People are hungry to learn more about Hilla...
    Name: text, dtype: object
```

```
len(ds_20)
```

```
    65
```

```
import os
import openai

OPENAI_API_KEY = "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"
openai.api_key = OPENAI_API_KEY

start_sequence = "\nA:"
restart_sequence = "\n\nQ: "

alist= []

for i in ds_65["text"]:
  response = openai.Completion.create(
    model="text-davinci-003",
    prompt="Text: "+ i+ "\nQ: Does the above text contain hyperpartisan elements t
    temperature=0,
    max_tokens=100,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0,
    stop=["\n"]
  )
  alist.append(1 if response["choices"][0]["text"]==" Yes" else 0)
```

```
response["choices"][0]["text"]

    ' No'
```

```
type(alist[0])

    str
```

Classical ML

```
import sklearn.metrics as metrics

print(metrics.confusion_matrix(y_true, alist))

# Print the precision and recall, among other metrics
print(metrics.classification_report(y_true, alist, digits=3))

    [[38  2]
     [15 10]]
                  precision    recall  f1-score   support

               0      0.717     0.950     0.817        40
               1      0.833     0.400     0.541        25

        accuracy                          0.738        65
       macro avg      0.775     0.675     0.679        65
    weighted avg      0.762     0.738     0.711        65
```

```
prompt="Text: "+ i+ "\nAdditional Information -  Hyperpartisan argument is somethir
prompt="Text: "+ i+ "\nQ: Does the above text contain hyperpartisan elements to it.
```

```
import random

random_list = [random.randint(0, 1) for _ in range(65)]
```

```
len(random_list)
```

```
65
```

```
import sklearn.metrics as metrics

print(metrics.confusion_matrix(y_true,random_list))

# Print the precision and recall, among other metrics
print(metrics.classification_report(y_true, random_list, digits=3))
```

```
[[20 20]
 [15 10]]
              precision    recall  f1-score   support

           0      0.571     0.500     0.533        40
           1      0.333     0.400     0.364        25

    accuracy                          0.462        65
   macro avg      0.452     0.450     0.448        65
weighted avg      0.480     0.462     0.468        65
```