Project 3 : Amazon reviews analysis.

This dataset consists of a few million Amazon customer reviews (input text) and star ratings(output labels) for learning how to train fastText forsentiment analysis. Dataset link- https://www.kaggle.com/datasets/bittlingmayer/amazonreviews

```
# creating path for kaggle file
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
```

Importing Amazon reviews Sentiment dataset

```
#API to fetch the dataset from kaggle
!kaggle datasets download -d bittlingmayer/amazonreviews
```

```
Downloading amazonreviews.zip to /content
100% 493M/493M [00:05<00:00, 42.1MB/s]
100% 493M/493M [00:05<00:00, 92.9MB/s]
```

```
# extracting a compressed dataset

from zipfile import ZipFile
dataset = '/content/amazonreviews.zip'

with ZipFile(dataset, 'r') as zip:
  zip.extractall()
  print('The dataset is extracted')
```

```
The dataset is extracted
```

Importing required libraries

```
#importing dependencies
import pandas as pd
import numpy as np
import re
import bz2
import csv
import fasttext
from sklearn.model_selection import train_test_split

# file path
file_path = "/content/train.ft.txt.bz2"

input_file_path = "/content/train.ft.txt.bz2"
output_file_path = "/content/train.csv"
```

```
with bz2.BZ2File(input_file_path, 'rb') as f:
    decompressed_data = f.read().decode('utf-8')


with open(output_file_path, 'w', newline='', encoding='utf-8') as
csvfile:
    csv_writer = csv.writer(csvfile)
    for line in decompressed_data.splitlines():
        label, text = line.split(' ', 1)
        label = label.replace('__label__', '')  # Remove '__label__'
prefix
        csv_writer.writerow([label, text])

print("Conversion completed. CSV file saved at:", output_file_path)
```

```
Conversion completed. CSV file saved at: /content/train.csv
```

```
data = pd.read_csv('/content/train.csv',names=["label", "text"])
```

```
data.shape
```

```
(3600000, 2)
```

```
data.isnull().sum()
```

```
label    0
text     0
dtype: int64
```

```
data = data.dropna()
```

```
data
```

	label	text
0	2	Stuning even for the non-gamer: This sound tra...
1	2	The best soundtrack ever to anything.: I'm rea...
2	2	Amazing!: This soundtrack is my favorite music...
3	2	Excellent Soundtrack: I truly like this soundt...
4	2	Remember, Pull Your Jaw Off The Floor After He...
...	...	...
3599995	1	Don't do it!!: The high chair looks great when...
3599996	1	Looks nice, low functionality: I have used thi...
3599997	1	compact, but hard to clean: We have a small ho...
3599998	1	what is it saying?: not sure what this book is...
3599999	2	Makes My Blood Run Red-White-And-Blue: I agree...

```
[3600000 rows x 2 columns]
```

```
xtrain, xtest, ytrain, ytest = train_test_split(data["text"],
data["label"], test_size=0.2, random_state=42)
```

```
from keras.preprocessing.text import Tokenizer
max_features = 5000
tokenizer = Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(xtrain)
X_train_seq = tokenizer.texts_to_sequences(xtrain)
X_test_seq = tokenizer.texts_to_sequences(xtest)

from keras.preprocessing.sequence import pad_sequences
maxlen = 100
X_train_pad = pad_sequences(X_train_seq, maxlen=maxlen)
X_test_pad = pad_sequences(X_test_seq, maxlen=maxlen)

from keras.models import Sequential
from keras.layers import LSTM, Embedding, Dense
from sklearn.metrics import classification_report
embedding_dim = 100
model = Sequential()
model.add(Embedding(input_dim=max_features, output_dim=embedding_dim,
input_length=maxlen))
model.add(LSTM(units=128))
model.add(Dense(units=1, activation='sigmoid'))

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

model.fit(X_train_pad, ytrain, epochs=8, batch_size=128,
validation_split=0.2)

Epoch 1/8
18000/18000 [==============================] - 281s 15ms/step - loss:
-570.3527 - accuracy: 0.5003 - val_loss: -1141.2633 - val_accuracy:
0.4992
Epoch 2/8
18000/18000 [==============================] - 215s 12ms/step - loss:
-1712.5162 - accuracy: 0.5003 - val_loss: -2290.8181 - val_accuracy:
0.4992
Epoch 3/8
18000/18000 [==============================] - 221s 12ms/step - loss:
-2858.8086 - accuracy: 0.5003 - val_loss: -3439.8909 - val_accuracy:
0.4992
Epoch 4/8
18000/18000 [==============================] - 218s 12ms/step - loss:
-4005.9768 - accuracy: 0.5003 - val_loss: -4589.3599 - val_accuracy:
0.4992
Epoch 5/8
18000/18000 [==============================] - 215s 12ms/step - loss:
-5152.2524 - accuracy: 0.5003 - val_loss: -5739.1558 - val_accuracy:
0.4992
Epoch 6/8
18000/18000 [==============================] - 202s 11ms/step - loss:
```

```
-6304.7080 - accuracy: 0.5003 - val_loss: -6897.8262 - val_accuracy:
0.4992
Epoch 7/8
18000/18000 [==============================] - 201s 11ms/step - loss:
-7460.2310 - accuracy: 0.5003 - val_loss: -8056.0327 - val_accuracy:
0.4992
Epoch 8/8
18000/18000 [==============================] - 206s 11ms/step - loss:
-8615.9170 - accuracy: 0.5003 - val_loss: -9214.1826 - val_accuracy:
0.4992

<keras.src.callbacks.History at 0x7db9105bbd60>
```