In [1]:
```
import torch
import nltk
import re
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
from transformers import DistilBertTokenizer, DistilBertForSequenc
from torch.utils.data import DataLoader, Dataset
from transformers import Trainer, TrainingArguments, TrainerCallba
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import pandas as pd
import joblib


# Load and prepare data
data = pd.read_csv('/Users/pillisachethan/Desktop/NLP project/test
data['post'] = data['post'].fillna('')

# Remove non-alphabetical characters and lowercase the text
data['post'] = data['post'].apply(lambda x: re.sub(r'[^a-z\s]', ''

# Initialize stopwords set and PorterStemmer
stop_words = set(stopwords.words('english'))
ps = PorterStemmer()

# Function to remove stopwords and apply stemming
def preprocess_text(text):
    # Remove stopwords
    filtered_words = [word for word in text.split() if word not in
    # Apply stemming
    stemmed_text = ' '.join(ps.stem(word) for word in filtered_wor
    return stemmed_text

# Apply the preprocessing to the 'post' column
data['post'] = data['post'].apply(preprocess_text)
data['text'] = data['post'].str.lower()


# Encode labels
label_encoder = LabelEncoder()
data['labels'] = label_encoder.fit_transform(data['subreddit'])

# Split data into training and testing sets
train_data, test_data = train_test_split(data, test_size=0.2, rand

# Custom Dataset class
class CustomDataset(Dataset):
    def __init__(self, data, tokenizer, max_len):
        self.data = data
        self.tokenizer = tokenizer
        self.max_len = max_len

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        text = self.data.iloc[idx]['text']
        label = self.data.iloc[idx]['labels']
        encoding = self.tokenizer(text, truncation=True, padding='
        return {
            'input_ids': torch.tensor(encoding['input_ids']),
            'attention_mask': torch.tensor(encoding['attention_mas
            'labels': torch.tensor(label)
        }

# Load the tokenizer and model
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-u
model = DistilBertForSequenceClassification.from_pretrained('disti

# Create Dataset for training and testing sets
train_dataset = CustomDataset(train_data, tokenizer, max_len=5)
test_dataset = CustomDataset(test_data, tokenizer, max_len=5)

# Training arguments with optimizations
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=1,  # Start with 1 epoch
    per_device_train_batch_size=100,  # Experiment with batch size
    per_device_eval_batch_size=100,
    gradient_accumulation_steps=2,  # Simulate larger batch size i
    warmup_steps=500,
    weight_decay=0.01,
    logging_dir='./logs',
    logging_steps=100,
    save_total_limit=1,
    fp16=False,  # Disable fp16 if there's no support for it
    bf16=True,  # Enable bfloat16 for faster training if supported
    report_to="tensorboard",  # Enable TensorBoard for monitoring
)


# Initialize Trainer with evaluation and early stopping callback
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,  # Include test dataset for evaluat
)

# Train the model
trainer.train()

# Save model, tokenizer, and label encoder
model.save_pretrained('./fine_tuned_distilbert')
tokenizer.save_pretrained('./fine_tuned_distilbert')
joblib.dump(label_encoder, './fine_tuned_distilbert/label_encoder.
```

```
Some weights of DistilBertForSequenceClassification were not init
ialized from the model checkpoint at distilbert-base-uncased and
are newly initialized: ['classifier.bias', 'classifier.weight',
'pre_classifier.bias', 'pre_classifier.weight']
You should probably TRAIN this model on a down-stream task to be
able to use it for predictions and inference.
```

[819/819 05:07, Epoch 0/1]

Step	Training Loss
100	3.101900
200	2.555900
300	2.231000
400	2.060400
500	1.972200
600	1.890300
700	1.859500
800	1.818000

```
Out[1]: ['./fine_tuned_distilbert/label_encoder.joblib']
```

In [3]:
```
# Evaluate the model on the test set
eval_results = trainer.evaluate()
print("Evaluation results:", eval_results)

# Reload for prediction
tokenizer = DistilBertTokenizer.from_pretrained('./fine_tuned_dist
model = DistilBertForSequenceClassification.from_pretrained('./fin
label_encoder = joblib.load('./fine_tuned_distilbert/label_encoder

# Function for individual text prediction
def predict_behavior(text):
    inputs = tokenizer(text, return_tensors="pt", truncation=True,
    # Move to device if using GPU or MPS (Metal)
    inputs = {key: value.to(model.device) for key, value in inputs

    with torch.no_grad():
        outputs = model(**inputs)
    logits = outputs.logits
    predicted_label = torch.argmax(logits, dim=1).item()
    subreddit = label_encoder.inverse_transform([predicted_label])
    return subreddit

# Function to predict on test set and add predictions
def predict_on_test_set(test_data):
    model.eval()  # Set model to evaluation mode
    predictions = []

    for i in range(len(test_data)):
        text = test_data.iloc[i]['post']
        predicted_subreddit = predict_behavior(text)
        predictions.append(predicted_subreddit)

    test_data['predicted'] = predictions
    return test_data

# Make predictions on the test set
test_data = predict_on_test_set(test_data)

# Display 'post' and 'predicted' columns for review
print(test_data[['post', 'predicted']])
```

```
Evaluation results: {'eval_loss': 1.7744554281234741, 'eval_runti
me': 36.9013, 'eval_samples_per_second': 1110.069, 'eval_steps_pe
r_second': 11.111, 'epoch': 0.9993898718730934}
                                                     post
predicted
151830  figur share may help ya httpswwwredditcomrcoro...       d
epression
90592   much ira invest mutual fund im finish grad sch...  person
alfinance
186469  fight fear futur turn year oldest child famili...       d
epression
64692   bodili ach mental agoni past week ive sever ac...       d
epression
110473  agre worker believ ive misclassifi option new ...  person
alfinance
...                                                   ...
...
7872    ne flsa question partner work sport busi midwe...  person
alfinance
11011   one realli want help ive struggl depress anxie...       d
epression
37091   im move anoth contin start new job feel someth...       d
epression
193699  ten year schizoversari around time first start...       d
epression
129011         lone joke im lone even duolingo send email
lonely

[40963 rows x 2 columns]
```

In [5]:
```python
from sklearn.metrics import accuracy_score

# Predict on the test set without modifying the DataFrame
test_data['predicted'] = test_data['post'].apply(predict_behavior)

# Calculate accuracy
accuracy = accuracy_score(test_data['labels'], label_encoder.trans
print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Accuracy: 62.30%
```

In [ ]: