I. Introduction

1. Domain-Specific Area: Mental Health

Mental illness ranks among the most pressing public health issues of the 21st century, with depression alone affecting more than 280 million people globally [1]. Despite greater awareness, mental health services remain overwhelmed by social stigma, inadequate clinical staff, and restricted early diagnosis. Due to this, many individuals turn to platforms like Reddit to share their problems and seek advice, typically in a community or subreddit focused on mental health like r/depression or r/anxiety.

This has been an avenue for Natural Language Processing (NLP) to step in to save the day in identifying and being aware of mental health issues through language. They have shown that depressed people employ some language patterns, such as the use of more first-person pronouns, more negative affective expression, and absolutist words such as "always" or "nothing" [2][3]. It has also shown that social media content can predict mental illness [4], and researchers like Chancellor and De Choudhury have argued for the need to make such tools ethically, especially when it comes to vulnerable online data [5].

In this project, we categorize Reddit posts into five classes of mental disorder: stress, depression, bipolar, personality disorder, and anxiety. Unlike the traditional binary sentiment analysis predicting that a individual is positive or negative for one type of mental disorder, this multi-class classification would be reflecting the actual scenario where multiple types of mental illnesses overlap or share similar linguistic characteristics.

To reach that, we will compare two different methods: a standard statistical model with TF-IDF features and Logistic Regression and a deep learning model with contextualized embeddings using pre-trained BERT. Statistical models are computationally efficient and interpretable but potentially insensitive to context, while transformer-based models like BERT are capable of capturing syntactic and semantic nuances, which may result in more accurate classification but at the cost of greater resources and careful fine-tuning.

2. Objectives

The objective of this project is to compare and evaluate the effectiveness of traditional statistical methods and modern embedding-based deep learning models in classifying user-generated text into five mental health categories: stress, depression, bipolar disorder, personality disorder, and anxiety. This task addresses the broader challenge of automated mental health detection in natural language and seeks to contribute to research in computational psychiatry, with a focus on multi-class classification.

The core motivation lies in understanding how different approaches handle the complexities of emotionally charged and often ambiguous user text. Previous work by Guntuku et al. (2019) and Coppersmith et al. (2015) has shown that NLP methods can extract meaningful mental health signals from social media data. However, much of the early work focused on binary classification, for exmaple, depressed vs. not depressed, and often used simpler models like bag-of-words or unigram frequency vectors. This project builds on that foundation by applying a more realistic multi-class framework and extending the evaluation to deep contextual embeddings using transformer-based models.

The first objective is to establish a baseline using a statistical model, specifically TF-IDF vectorisation combined with a Multinomial Naive Bayes classifier, which is widely used in traditional text classification tasks due to its simplicity and effectiveness.

The second objective is to implement and fine-tune a traditional statistical model, using the same TF-IDF features in a Logistic Regression classifier which will be used as the representative traditional statistical model.

The third objective is to implement and fine-tune an embedding-based deep learning model, specifically a pre-trained BERT (Bidirectional Encoder Representations from Transformers) model. BERT has demonstrated strong performance in a variety of NLP tasks by capturing rich contextual dependencies in language [8]. The aim is to assess how well such models distinguish between closely related mental health categories, particularly when semantic nuance is critical.

The fourth and last objective is to analyse and compare the results across these models using appropriate evaluation metrics, including accuracy, precision, recall, and F1-score (macro-averaged). The comparison will focus not only on raw performance but also on qualitative aspects such as misclassification patterns and model interpretability.

In summary, this project seeks to offer insights into which modeling approach is more suitable for mental health classification tasks involving complex, emotionally nuanced language.

3. Dataset Description

The data used in this project is extracted from the Reddit Mental Health Dataset compiled by Neel Ghoshal and is publicly available on Kaggle [7]. It consists of anonymized Reddit posts collected from various subreddits related to mental health. The posts are labeled with a category of a mental illness, thus making the dataset suitable for supervised multi-class classification tasks in the area of Natural Language Processing (NLP).

The dataset comprises of 5957 reddit posts, evenly distributed across five target classes. The distribution is as follows:

- 0: stress (1181)

- 1: depression (1202)
- 2: bipolar disorder (1185)
- 3: personality disorder (1201)
- 4: anxiety (1188)

Each row in the dataset contains:

- title: title of the reddit post
- text: full body text of the reddit post
- target: integer-encoded label of the mental health class

This dataset was selected due to its domain relevance, balanced class distribution, and raw-text input format allowing for fair comparison of models with different representational needs. The balanced sample distribution across classes helps mitigate the risk of model bias and ensures meaningful evaluation of the performance of a classifier on minority classes.

All content originates from publicly accessible Reddit forums and does not contain any personally identifying data. The dataset is intended for research and educational purposes, and its use complies with ethical standards for public social media data.

4. Evualuation Methodology

To assess the performance of this project's models, we use a set of common classification metrics: Accuracy, Precision, Recall and F1-score. These metrics were chosen due to their widespread use in text classification and tand can reflect the different aspects of model performance.

Classification Metrics

Accuracy measures the proportion of correctly predicted instances over all predictions. Precision is the proportion of true positives over all predicted positives for each class. Recall indicates how well the model identifies all instances of each class that are true. F1-score, the harmonic mean of precision and recall, balances both metrics and is especially valuable when evaluating misclassification trade-offs.

These metrics are computed using scikit-learn's classification_report, and results are plotted one step further using a confusion matrix, which provides an interpretable view of class-specific prediction errors. This is especially helpful for identifying systematic confusion between similar classes, such as bipolar disorder and personality disoder.

Cross-validation

Cross-validation will be performed for traditional models like Logistic Regression, particularly during hyperparameter tuning to reduce overfitting and to validate model generalisability.

This helps ensure that performance differences are due to modelling choices and not variance in a single train-test split.

Train-Test Split

The dataset is split into a 80-20 train-test split, stratified by class to maintain balance. The test set is not touched during training and is only used for the final evaluation of each model.

Comparative Analysis

Performance of three models is compared:

- Baseline model: Multinomial Naive Bayes with default settings
- Statistical model: Logistic Regression with TF-IDF features and tuned hyperparameters
- Embedding-based model: BERT fine-tuned on raw text

The evaluation emphasizes both quantitative metrics and qualitative interpretation of misclassifications, giving insight into the strengths, weaknesses, and real-world viability of each model in recognizing and distinguishing between mental health classes.

II. Implementation

5. Data Preprocessing

To prepare the Reddit Mental Health dataset for multi-class classification, separate preprocessing pipelines were implemented for the statistical models and the deep learning model. Each pipeline reflects the differing requirements of traditional machine learning (TF-IDF vectorization) and modern embedding-based models (contextual token embeddings).

In [1]:
```
# prevent tensorflow from using GPU and clashing with pytorch
import os
os.environ["USE_TF"] = "0"
```

In [2]:
```
# import libraries
import pandas as pd
import numpy as np
import re
import seaborn as sns
import matplotlib.pyplot as plt

# NLP
import nltk
from nltk.corpus import stopwords, wordnet
from nltk.stem import WordNetLemmatizer
from nltk import pos_tag, word_tokenize
from sklearn.model_selection import train_test_split
```

```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

# deep learning
import transformers
from transformers import BertTokenizer
import torch
from torch.utils.data import Dataset, DataLoader
from transformers import BertForSequenceClassification
from transformers import Trainer, TrainingArguments

# setup
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
nltk.download('stopwords')

stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
```

```
c:\Users\Admin\anaconda3\Lib\site-packages\tqdm\auto.py:21: TqdmWarning: IProgress n
ot found. Please update jupyter and ipywidgets. See https://ipywidgets.readthedocs.i
o/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [3]:
```python
# import dataset
df = pd.read_csv("RedditMentalHealthData.csv")

# show structure
df.head()
```

Out[3]:

	Unnamed: 0	text	title	target
0	0	Welcome to /r/depression's check-in post - a p...	Regular check-in post, with information about ...	1
1	1	We understand that most people who reply immed...	Our most-broken and least-understood rules is ...	1
2	2	Anyone else just miss physical touch? I crave ...	I haven’t been touched, or even hugged, in so ...	1
3	3	I’m just so ashamed. Everyone and everything f...	Being Depressed is Embarrassing	1
4	4	I really need a friend. I don't even have a si...	I'm desperate for a friend and to feel loved b...	1

In [4]:

```
# remove unnecessary columns
df = df.drop(['Unnamed: 0'], axis=1)

df.head()
```

Out[4]:

	text	title	target
0	Welcome to /r/depression's check-in post - a p...	Regular check-in post, with information about ...	1
1	We understand that most people who reply immed...	Our most-broken and least-understood rules is ...	1
2	Anyone else just miss physical touch? I crave ...	I haven’t been touched, or even hugged, in so ...	1
3	I’m just so ashamed. Everyone and everything f...	Being Depressed is Embarrassing	1
4	I really need a friend. I don't even have a si...	I'm desperate for a friend and to feel loved b...	1

Each record in the dataset contains a title and a text field. These were combined into a new feature, full_text, to capture both the summarised intent and the detailed body of each Reddit post. Posts were then filtered to ensure no null or empty values were present.

In [5]:

```
# combine title + text into one column
df['full_text'] = df['title'].fillna('') + ' ' + df['text'].fillna('')

# drop rows with empty combined text
df = df[df['full_text'].str.strip().astype(bool)]
```

In [6]:

```
# check class balance
df['target'].value_counts()
```

```
Out[6]: target
1    1202
3    1201
4    1188
2    1185
0    1181
Name: count, dtype: int64
```

```
In [7]: # preview
df[['full_text', 'target']].head()
```

Out[7]:

	full_text	target
0	Regular check-in post, with information about ...	1
1	Our most-broken and least-understood rules is ...	1
2	I haven’t been touched, or even hugged, in so ...	1
3	Being Depressed is Embarrassing I’m just so as...	1
4	I'm desperate for a friend and to feel loved b...	1

Train-Test Split

To ensure robust evaluation, the dataset was split into 80% training and 20% testing using train_test_split with stratification based on the target variable. This preserved the balanced distribution of classes in both subsets.

```
In [8]: # train test split with stratification
X_train, X_test, y_train, y_test = train_test_split(
    df['full_text'], df['target'], test_size=0.2, random_state=42, stratify=df['tar
)

# check shape
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
Out[8]: ((4765,), (1192,), (4765,), (1192,))
```

Preprocessing for Statistical Models

Statistical models such as Multinomial Naive Bayes and Logistic Regression require sparse vector input. To preprocess the raw text, the following steps were applied:

- Lowercasing: All text was converted to lowercase to avoid duplication of tokens due to case sensitivity.
- Punctuation Removal: Non-alphanumeric characters were stripped to reduce vocabulary noise.
- Tokenization: The cleaned text was split into tokens using NLTK’s word_tokenize function.
- Part-of-Speech (POS) Tagging: Each token was POS-tagged using NLTK's pos_tag() method.

- Stopword Removal: Common English stopwords were removed using NLTK's default stopword list.
- Lemmatization with WordNet: Tokens were lemmatized using the WordNet Lemmatizer. Crucially, POS tags were mapped to WordNet's expected tag format to ensure accurate lemmatization across word types

TF-IDF Vectorization: The preprocessed text was vectorized into numeric representations using Scikit-learn's TfidfVectorizer. The vocabulary was set to 5,000 in order to obtain a compromise between information value and dimensionality. The vectorizer was fitted on the training set and applied to both training and testing partitions.

In [9]:
```
# preprocessing
def clean_text(text):
    # lowercase
    text = text.lower()

    # remove punctuation and digits
    text = re.sub(r'[^a-z\s]', '', text)

    # tokenize
    tokens = word_tokenize(text)

    # pos tagging & lemmatization
    tagged = pos_tag(tokens)
    lemmatized = [
        lemmatizer.lemmatize(word, get_wordnet_pos(pos))
        for word, pos in tagged if word not in stop_words
    ]
    return ' '.join(lemmatized)

# pos tagging
def get_wordnet_pos(treebank_tag):
    if treebank_tag.startswith('J'):
        return wordnet.ADJ
    elif treebank_tag.startswith('V'):
        return wordnet.VERB
    elif treebank_tag.startswith('N'):
        return wordnet.NOUN
    elif treebank_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN  # default


# apply to training and test sets
X_train_clean = X_train.apply(clean_text)
X_test_clean = X_test.apply(clean_text)
```

In [10]:
```
# TF-IDF Vectorisation
tfidf = TfidfVectorizer(max_features=5000)

X_train_tfidf = tfidf.fit_transform(X_train_clean)
X_test_tfidf = tfidf.transform(X_test_clean)
```

```
# check shape
X_train_tfidf.shape, X_test_tfidf.shape
```

Out[10]:
```
((4765, 5000), (1192, 5000))
```

Preprocessing for the Deep Learning Model

The deep learning model—based on BERT—was trained on the raw full_text input without manual cleaning or lemmatization. Instead, the text was tokenized using HuggingFace's bert-base-uncased tokenizer, which handles:

- Lowercasing
- Subword tokenization
- Special tokens (e.g., [CLS], [SEP])
- Padding and truncation to a maximum sequence length (e.g., 256 tokens)

These tokenized inputs were transformed into tensors (input_ids, attention_mask) and served directly to the BERT model during training and evaluation.

In [11]:
```
# load BERT tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# tokenization function
def tokenize_bert(texts):
    return tokenizer(
        list(texts),
        padding='max_length',
        truncation=True,
        max_length=256,
        return_tensors='pt'
    )

# apply tokenization
train_tokens = tokenize_bert(X_train)
test_tokens = tokenize_bert(X_test)

# check shape
train_tokens['input_ids'].shape
```

Out[11]:
```
torch.Size([4765, 256])
```

6. Baseline Performance

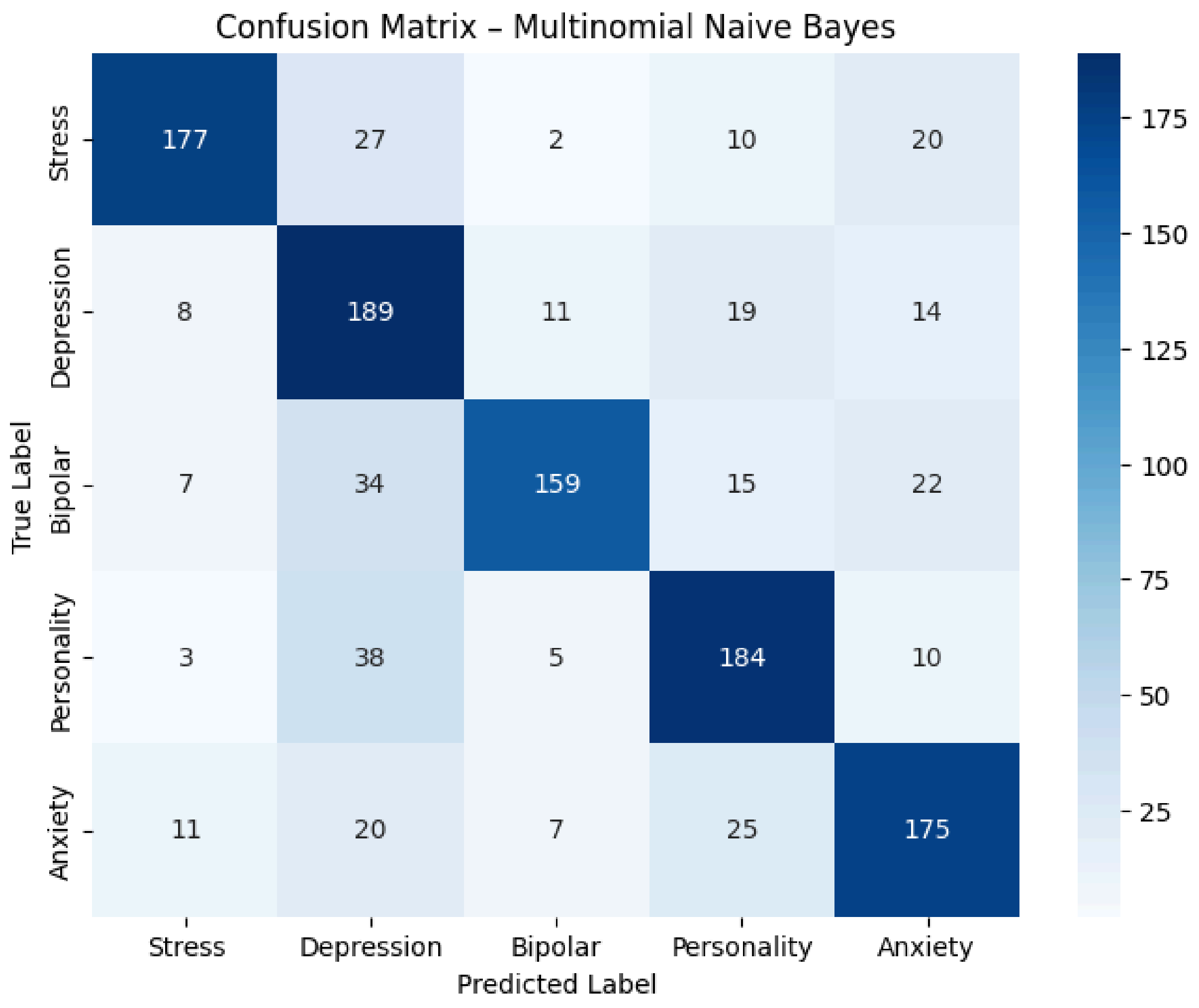
To establish a benchmark for evaluating more sophisticated models, a Multinomial Naive Bayes (MNB) classifier was selected as the baseline model. It is a model that is commonly employed in NLP applications as a fast, interpretable, and computationally inexpensive baseline (Rennie et al., 2003). It is particularly effective with bag-of-words or TF-IDF text representations, though it lacks the ability to capture contextual semantics or word order.

The baseline model was trained using default hyperparameters on TF-IDF vectors derived from the preprocessed full_text of Reddit posts. The text was preprocessed by lowercasing, punctuation removal, removal of stop words, and WordNet lemmatization with POS tagging to ensure linguistic normalization. A TfidfVectorizer with a maximum of 5,000 features was applied to transform the preprocessed text into sparse vector representation.

In [12]:
```
# baseline mode - NB
nb_model = MultinomialNB()
nb_model.fit(X_train_tfidf, y_train)

# predict on test data
y_pred_nb = nb_model.predict(X_test_tfidf)

print("Multinomial Naive Bayes Classification Report:")
print(classification_report(y_test, y_pred_nb))
```

```
Multinomial Naive Bayes Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.75      0.80       236
           1       0.61      0.78      0.69       241
           2       0.86      0.67      0.76       237
           3       0.73      0.77      0.75       240
           4       0.73      0.74      0.73       238

    accuracy                           0.74      1192
   macro avg       0.76      0.74      0.74      1192
weighted avg       0.76      0.74      0.74      1192
```

In [13]:
```
# confusion matrix for nb
cm = confusion_matrix(y_test, y_pred_nb)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Stress', 'Depression', 'Bipolar', 'Personality', 'Anxiety
            yticklabels=['Stress', 'Depression', 'Bipolar', 'Personality', 'Anxiety
plt.title('Confusion Matrix - Multinomial Naive Bayes')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



Baseline Model Observations

- With an accuracy of 74% and a 0.74 macro-averaged F1-score, this shows that the model performs decently overall but lacks fine-grained discrimination between related mental health categories.
- Stress and Bipolar Disorder have the highest F1-scores, indicating that the model detects extreme emotional states or clearly distinguishable patterns more effectively.
- This also highlights the lack of precision for the other classes, especially depression, having the lowest precision of 0.61 and the lowest F1-score, 0.69. The confusion matrix also indicates it has the most false positives.
- Depression vs. Personality and Depression vs. Anxiety show significant misclassification, highlighting the model's inability to distinguish overlapping emotional tones.
- Personality disorder has 38 posts wrongly predicted as depression, indicating semantic overlap, and that the model might associate depressive expressions with personality disorders.
- However, While not highly accurate, the model is fast to train and simple to interpret.

7. Comparative Classification approach

This section compares the performance of two primary model types, a traditional statistical model and a deep learning embedding-based model, on the task of multi-class classification of Reddit mental health posts.

Statistical Model: Logistic Regression with TF-IDF

The statistical model used for comparison is a Logistic Regression classifier, trained on TF-IDF vectors of the lemmatized post text. Logistic Regression is a commonly used linear classifier that performs well in text classification tasks and offers interpretability and speed. In this implementation, hyperparameter tuning was conducted using grid search to identify the best values for:

- C (inverse regularization strength)
- penalty (l2)
- solver (liblinear or saga)

The best model was selected via 5-fold cross-validation using macro F1-score as the primary selection metric. The trained model was then evaluated on the test set using precision, recall, F1-score, and confusion matrix.

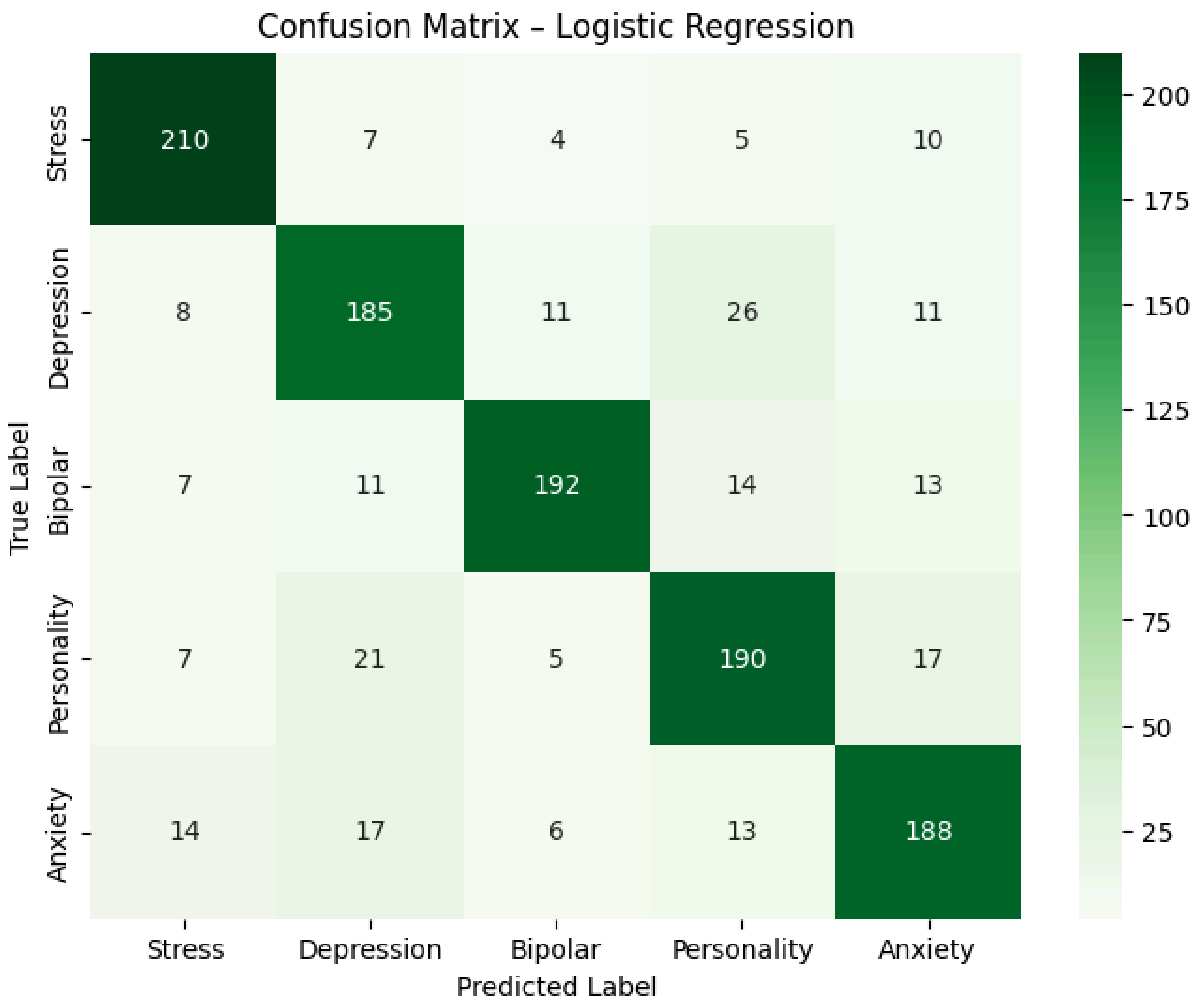
In [14]:
```
# define parameter grid for LR hyperparameter tuning
param_grid = {
    'C': [0.01, 0.1, 1, 10],
    'solver': ['liblinear', 'saga'],
    'penalty': ['l2']
}

# initialize Logistic Regression
lr = LogisticRegression(max_iter=1000)

# grid search with 5-fold cross-validation
grid_search = GridSearchCV(
    lr, param_grid, cv=5,
    scoring='f1_macro', verbose=1, n_jobs=-1
)

# fit on training data
grid_search.fit(X_train_tfidf, y_train)

# best model
best_lr = grid_search.best_estimator_
```

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
```

In [15]:
```
# predict
y_pred_lr = best_lr.predict(X_test_tfidf)

print("Tuned Logistic Regression Classification Report:")
print(classification_report(y_test, y_pred_lr, digits=3))
```

```
Tuned Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0      0.854     0.890     0.871       236
           1      0.768     0.768     0.768       241
           2      0.881     0.810     0.844       237
           3      0.766     0.792     0.779       240
           4      0.787     0.790     0.788       238

    accuracy                          0.810      1192
   macro avg      0.811     0.810     0.810      1192
weighted avg      0.811     0.810     0.810      1192
```

In [16]:
```
# confusion matrix for lr
cm = confusion_matrix(y_test, y_pred_lr)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Greens',
            xticklabels=['Stress', 'Depression', 'Bipolar', 'Personality', 'Anxiety
            yticklabels=['Stress', 'Depression', 'Bipolar', 'Personality', 'Anxiety
plt.title('Confusion Matrix - Logistic Regression')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



Embedding-Based Model: BERT (Fine-Tuned)

The embedding-based model leverages a pre-trained BERT (bert-base-uncased) model from the HuggingFace Transformers library. The raw, unprocessed full_text field was tokenized using the BERT tokenizer and passed into a classification head on top of the encoder.

Key training parameters included:

- Max token length: 256
- Batch size: 16
- Epochs: 3–4
- Optimizer: AdamW
- Learning rate: 2e-5

The model was fine-tuned on the training set using PyTorch and evaluated on the test set. Since BERT is sensitive to subtle differences in word usage and context, it is expected to outperform traditional models in tasks requiring semantic understanding.

In [17]:
```
class RedditDataset(Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels.values if hasattr(labels, "values") else labels

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

    def __getitem__(self, idx):
        item = {key: val[idx] for key, val in self.encodings.items()}
        item["labels"] = torch.tensor(self.labels[idx])
        return item

train_dataset = RedditDataset(train_tokens, y_train)
test_dataset = RedditDataset(test_tokens, y_test)
```

In [18]:
```
model = BertForSequenceClassification.from_pretrained(
    'bert-base-uncased',
    num_labels=5  # 5 classes
)

training_args = TrainingArguments(
    output_dir='./results',
    eval_strategy="epoch",
    num_train_epochs=3,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    learning_rate=2e-5,
    logging_dir='./logs',
    logging_steps=10,
    load_best_model_at_end=True,
    save_strategy="epoch",
    report_to="none",
    # Make sure GPU is utilized
    fp16=torch.cuda.is_available(),  # mixed precision if GPU supports it
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
)

trainer.train()
```

```
Some weights of BertForSequenceClassification were not initialized from the model ch
eckpoint at bert-base-uncased and are newly initialized: ['classifier.bias', 'classi
fier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for
predictions and inference.
  1%|          | 11/1788 [00:02<04:23,  6.75it/s]
{'loss': 1.6163, 'grad_norm': 6.3821210861206055, 'learning_rate': 1.988814317673378
e-05, 'epoch': 0.02}
  1%|          | 21/1788 [00:03<04:12,  7.00it/s]
{'loss': 1.6199, 'grad_norm': 9.801592826843262, 'learning_rate': 1.9776286353467565
e-05, 'epoch': 0.03}
  2%|▏         | 31/1788 [00:04<04:01,  7.29it/s]
{'loss': 1.5976, 'grad_norm': 7.0709428787231445, 'learning_rate': 1.967561521252796
6e-05, 'epoch': 0.05}
  2%|▏         | 41/1788 [00:06<04:02,  7.21it/s]
{'loss': 1.5556, 'grad_norm': 7.566036224365234, 'learning_rate': 1.9563758389261745
e-05, 'epoch': 0.07}
  3%|▎         | 51/1788 [00:07<03:56,  7.35it/s]
{'loss': 1.4949, 'grad_norm': 8.275328636169434, 'learning_rate': 1.945190156599553e
-05, 'epoch': 0.08}
  3%|▎         | 61/1788 [00:09<03:45,  7.66it/s]
{'loss': 1.461, 'grad_norm': 11.285417556762695, 'learning_rate': 1.935123042505593e
-05, 'epoch': 0.1}
  4%|▍         | 71/1788 [00:10<03:47,  7.55it/s]
{'loss': 1.4628, 'grad_norm': 11.967494010925293, 'learning_rate': 1.923937360178971
e-05, 'epoch': 0.12}
  5%|▍         | 81/1788 [00:11<03:58,  7.17it/s]
{'loss': 1.4739, 'grad_norm': 11.621981620788574, 'learning_rate': 1.912751677852349
3e-05, 'epoch': 0.13}
  5%|▌         | 91/1788 [00:13<04:31,  6.26it/s]
{'loss': 1.3677, 'grad_norm': 8.40988826751709, 'learning_rate': 1.9015659955257272e
-05, 'epoch': 0.15}
  6%|▌         | 101/1788 [00:14<03:55,  7.16it/s]
{'loss': 1.4397, 'grad_norm': 16.716218948364258, 'learning_rate': 1.890380313199105
2e-05, 'epoch': 0.17}
  6%|▌         | 111/1788 [00:16<04:03,  6.90it/s]
{'loss': 1.27, 'grad_norm': 8.483407974243164, 'learning_rate': 1.8791946308724832e-
05, 'epoch': 0.18}
  7%|▋         | 121/1788 [00:17<03:54,  7.11it/s]
{'loss': 1.2259, 'grad_norm': 9.008952140808105, 'learning_rate': 1.8680089485458615
e-05, 'epoch': 0.2}
```

```
  7%|▋         | 131/1788 [00:18<03:52,  7.13it/s]
{'loss': 1.2081, 'grad_norm': 7.166111946105957, 'learning_rate': 1.8568232662192395
e-05, 'epoch': 0.22}
  8%|▊         | 141/1788 [00:20<03:54,  7.02it/s]
{'loss': 1.1195, 'grad_norm': 14.848169326782227, 'learning_rate': 1.845637583892617
4e-05, 'epoch': 0.23}
  8%|▊         | 151/1788 [00:21<03:47,  7.20it/s]
{'loss': 0.917, 'grad_norm': 9.908174514770508, 'learning_rate': 1.8344519015659958e
-05, 'epoch': 0.25}
  9%|▉         | 161/1788 [00:22<03:46,  7.18it/s]
{'loss': 0.9877, 'grad_norm': 12.766563415527344, 'learning_rate': 1.823266219239373
7e-05, 'epoch': 0.27}
 10%|▉         | 171/1788 [00:24<03:46,  7.13it/s]
{'loss': 0.9576, 'grad_norm': 11.631775856018066, 'learning_rate': 1.812080536912751
7e-05, 'epoch': 0.29}
 10%|█         | 181/1788 [00:25<03:46,  7.10it/s]
{'loss': 0.9444, 'grad_norm': 12.581461906433105, 'learning_rate': 1.80089485458613e
-05, 'epoch': 0.3}
 11%|█         | 191/1788 [00:27<03:40,  7.23it/s]
{'loss': 0.9131, 'grad_norm': 14.424195289611816, 'learning_rate': 1.789709172259508
e-05, 'epoch': 0.32}
 11%|█         | 201/1788 [00:28<03:37,  7.30it/s]
{'loss': 0.8054, 'grad_norm': 9.868042945861816, 'learning_rate': 1.778523489932886e
-05, 'epoch': 0.34}
 12%|█▏        | 211/1788 [00:29<03:38,  7.21it/s]
{'loss': 0.8173, 'grad_norm': 14.263004302978516, 'learning_rate': 1.767337807606264
3e-05, 'epoch': 0.35}
 12%|█▏        | 221/1788 [00:31<03:36,  7.23it/s]
{'loss': 0.9369, 'grad_norm': 15.303391456604004, 'learning_rate': 1.756152125279642
3e-05, 'epoch': 0.37}
 13%|█▎        | 231/1788 [00:32<03:35,  7.23it/s]
{'loss': 0.6742, 'grad_norm': 5.078694820404053, 'learning_rate': 1.7449664429530202
e-05, 'epoch': 0.39}
 13%|█▎        | 241/1788 [00:34<03:34,  7.20it/s]
{'loss': 0.9349, 'grad_norm': 11.902472496032715, 'learning_rate': 1.733780760626398
2e-05, 'epoch': 0.4}
 14%|█▍        | 251/1788 [00:35<03:32,  7.25it/s]
{'loss': 0.7754, 'grad_norm': 17.28523826599121, 'learning_rate': 1.7225950782997765
e-05, 'epoch': 0.42}
 15%|█▍        | 261/1788 [00:36<03:31,  7.21it/s]
{'loss': 0.7918, 'grad_norm': 9.798562049865723, 'learning_rate': 1.7114093959731545
e-05, 'epoch': 0.44}
 15%|█▌        | 271/1788 [00:38<03:29,  7.23it/s]
{'loss': 0.778, 'grad_norm': 19.636566162109375, 'learning_rate': 1.7002237136465325
e-05, 'epoch': 0.45}
 16%|█▌        | 281/1788 [00:39<03:27,  7.25it/s]
{'loss': 0.7435, 'grad_norm': 14.033241271972656, 'learning_rate': 1.689038031319910
5e-05, 'epoch': 0.47}
 16%|█▋        | 291/1788 [00:40<03:26,  7.26it/s]
{'loss': 0.7075, 'grad_norm': 16.209775924682617, 'learning_rate': 1.677852348993288
8e-05, 'epoch': 0.49}
```

```
 17%|█▋        | 301/1788 [00:42<03:25,  7.23it/s]
{'loss': 0.79, 'grad_norm': 16.380191802978516, 'learning_rate': 1.6666666666666667e
-05, 'epoch': 0.5}
 17%|█▋        | 311/1788 [00:43<03:23,  7.25it/s]
{'loss': 0.7824, 'grad_norm': 7.717459678649902, 'learning_rate': 1.6554809843400447
e-05, 'epoch': 0.52}
 18%|█▊        | 321/1788 [00:45<03:22,  7.25it/s]
{'loss': 0.81, 'grad_norm': 14.93555736541748, 'learning_rate': 1.644295302013423e-0
5, 'epoch': 0.54}
 19%|█▊        | 331/1788 [00:46<03:27,  7.03it/s]
{'loss': 0.6805, 'grad_norm': 15.29552173614502, 'learning_rate': 1.633109619686801e
-05, 'epoch': 0.55}
 19%|█▉        | 341/1788 [00:47<03:20,  7.22it/s]
{'loss': 0.887, 'grad_norm': 7.312058448791504, 'learning_rate': 1.6219239373601793e
-05, 'epoch': 0.57}
 20%|█▉        | 351/1788 [00:49<03:19,  7.19it/s]
{'loss': 0.9315, 'grad_norm': 18.697898864746094, 'learning_rate': 1.610738255033557
e-05, 'epoch': 0.59}
 20%|██        | 361/1788 [00:50<03:21,  7.08it/s]
{'loss': 0.8142, 'grad_norm': 12.818703651428223, 'learning_rate': 1.599552572706935
3e-05, 'epoch': 0.6}
 21%|██        | 371/1788 [00:52<03:21,  7.02it/s]
{'loss': 0.5875, 'grad_norm': 4.813537120819092, 'learning_rate': 1.5883668903803133
e-05, 'epoch': 0.62}
 21%|██▏       | 381/1788 [00:53<03:08,  7.47it/s]
{'loss': 0.8333, 'grad_norm': 10.066493034362793, 'learning_rate': 1.577181208053691
6e-05, 'epoch': 0.64}
 22%|██▏       | 391/1788 [00:54<03:05,  7.54it/s]
{'loss': 0.8486, 'grad_norm': 11.821328163146973, 'learning_rate': 1.565995525727069
5e-05, 'epoch': 0.65}
 22%|██▏       | 401/1788 [00:56<03:05,  7.49it/s]
{'loss': 0.7997, 'grad_norm': 12.518451690673828, 'learning_rate': 1.554809843400447
5e-05, 'epoch': 0.67}
 23%|██▎       | 411/1788 [00:57<03:02,  7.56it/s]
{'loss': 0.8259, 'grad_norm': 5.261010646820068, 'learning_rate': 1.5436241610738255
e-05, 'epoch': 0.69}
 24%|██▎       | 421/1788 [00:58<03:00,  7.59it/s]
{'loss': 0.7346, 'grad_norm': 9.369771957397461, 'learning_rate': 1.5324384787472038
e-05, 'epoch': 0.7}
 24%|██▍       | 431/1788 [01:00<02:58,  7.61it/s]
{'loss': 0.6657, 'grad_norm': 8.73316478729248, 'learning_rate': 1.521252796420582e-
05, 'epoch': 0.72}
 25%|██▍       | 441/1788 [01:01<02:55,  7.69it/s]
{'loss': 0.6544, 'grad_norm': 17.055912017822266, 'learning_rate': 1.510067114093959
8e-05, 'epoch': 0.74}
 25%|██▌       | 451/1788 [01:02<02:51,  7.82it/s]
{'loss': 0.6936, 'grad_norm': 10.855052947998047, 'learning_rate': 1.498881431767337
9e-05, 'epoch': 0.76}
 26%|██▌       | 461/1788 [01:03<02:49,  7.85it/s]
{'loss': 0.8441, 'grad_norm': 10.592960357666016, 'learning_rate': 1.487695749440716
e-05, 'epoch': 0.77}
```

```
 26%|██▋       | 471/1788 [01:05<02:45,  7.94it/s]
{'loss': 0.9037, 'grad_norm': 13.446985244750977, 'learning_rate': 1.477628635346756
3e-05, 'epoch': 0.79}
 27%|██▋       | 481/1788 [01:06<02:46,  7.85it/s]
{'loss': 0.5528, 'grad_norm': 6.587096214294434, 'learning_rate': 1.4664429530201343
e-05, 'epoch': 0.81}
 27%|██▋       | 491/1788 [01:07<02:46,  7.80it/s]
{'loss': 0.6212, 'grad_norm': 9.132009506225586, 'learning_rate': 1.4552572706935124
e-05, 'epoch': 0.82}
 28%|██▊       | 501/1788 [01:08<02:44,  7.83it/s]
{'loss': 0.6505, 'grad_norm': 15.587601661682129, 'learning_rate': 1.444071588366890
6e-05, 'epoch': 0.84}
 29%|██▊       | 511/1788 [01:10<02:43,  7.83it/s]
{'loss': 0.5689, 'grad_norm': 13.63081169128418, 'learning_rate': 1.4328859060402687
e-05, 'epoch': 0.86}
 29%|██▉       | 521/1788 [01:11<02:41,  7.87it/s]
{'loss': 0.5091, 'grad_norm': 8.055068016052246, 'learning_rate': 1.4217002237136465
e-05, 'epoch': 0.87}
 30%|██▉       | 531/1788 [01:12<02:40,  7.82it/s]
{'loss': 0.6686, 'grad_norm': 12.391066551208496, 'learning_rate': 1.410514541387024
7e-05, 'epoch': 0.89}
 30%|███      | 541/1788 [01:14<02:39,  7.80it/s]
{'loss': 0.5818, 'grad_norm': 16.15445899963379, 'learning_rate': 1.3993288590604028
e-05, 'epoch': 0.91}
 31%|███      | 551/1788 [01:15<02:38,  7.81it/s]
{'loss': 0.6181, 'grad_norm': 3.3160815238952637, 'learning_rate': 1.388143176733781
e-05, 'epoch': 0.92}
 31%|███▏      | 561/1788 [01:16<02:37,  7.79it/s]
{'loss': 0.6489, 'grad_norm': 7.259377956390381, 'learning_rate': 1.3769574944071588
e-05, 'epoch': 0.94}
 32%|███▏      | 571/1788 [01:17<02:35,  7.82it/s]
{'loss': 0.6505, 'grad_norm': 16.636598587036133, 'learning_rate': 1.365771812080537
e-05, 'epoch': 0.96}
 32%|███▏      | 581/1788 [01:19<02:33,  7.84it/s]
{'loss': 0.7925, 'grad_norm': 21.329917907714844, 'learning_rate': 1.354586129753915
e-05, 'epoch': 0.97}
 33%|███▎      | 591/1788 [01:20<02:32,  7.85it/s]
{'loss': 0.7831, 'grad_norm': 19.13155746459961, 'learning_rate': 1.3434004474272932
e-05, 'epoch': 0.99}

 33%|███▎      | 596/1788 [01:25<02:29,  7.99it/s]
{'eval_loss': 0.6037998199462891, 'eval_runtime': 4.3529, 'eval_samples_per_second':
273.84, 'eval_steps_per_second': 34.23, 'epoch': 1.0}
 34%|███▎      | 601/1788 [01:27<10:22,  1.91it/s]
{'loss': 0.7402, 'grad_norm': 7.570547580718994, 'learning_rate': 1.3322147651006712
e-05, 'epoch': 1.01}
 34%|███▍      | 611/1788 [01:28<02:44,  7.15it/s]
{'loss': 0.5057, 'grad_norm': 6.643324375152588, 'learning_rate': 1.3210290827740493
e-05, 'epoch': 1.02}
 35%|███▍      | 621/1788 [01:29<02:29,  7.81it/s]
```

```
{'loss': 0.509, 'grad_norm': 5.570816993713379, 'learning_rate': 1.3098434004474275e
-05, 'epoch': 1.04}
 35%|███▌      | 631/1788 [01:31<02:28,  7.80it/s]
{'loss': 0.3661, 'grad_norm': 9.434049606323242, 'learning_rate': 1.2986577181208055
e-05, 'epoch': 1.06}
 36%|███▌      | 641/1788 [01:32<02:26,  7.84it/s]
{'loss': 0.5312, 'grad_norm': 22.3602294921875, 'learning_rate': 1.2874720357941834e
-05, 'epoch': 1.07}
 36%|███▋      | 651/1788 [01:33<02:25,  7.81it/s]
{'loss': 0.6414, 'grad_norm': 11.962773323059082, 'learning_rate': 1.276286353467561
6e-05, 'epoch': 1.09}
 37%|███▋      | 661/1788 [01:34<02:23,  7.86it/s]
{'loss': 0.3473, 'grad_norm': 7.0785322189331055, 'learning_rate': 1.265100671140939
7e-05, 'epoch': 1.11}
 38%|███▊      | 671/1788 [01:36<02:22,  7.81it/s]
{'loss': 0.5699, 'grad_norm': 11.996193885803223, 'learning_rate': 1.253914988814317
9e-05, 'epoch': 1.12}
 38%|███▊      | 681/1788 [01:37<02:21,  7.81it/s]
{'loss': 0.5051, 'grad_norm': 19.799774169921875, 'learning_rate': 1.242729306487695
7e-05, 'epoch': 1.14}
 39%|███▊      | 691/1788 [01:38<02:19,  7.87it/s]
{'loss': 0.4194, 'grad_norm': 18.009729385375977, 'learning_rate': 1.232662192393736
1e-05, 'epoch': 1.16}
 39%|███▉      | 701/1788 [01:39<02:18,  7.83it/s]
{'loss': 0.3779, 'grad_norm': 4.403044700622559, 'learning_rate': 1.2214765100671143
e-05, 'epoch': 1.17}
 40%|███▉      | 711/1788 [01:41<02:18,  7.79it/s]
{'loss': 0.5313, 'grad_norm': 21.333118438720703, 'learning_rate': 1.210290827740492
4e-05, 'epoch': 1.19}
 40%|████      | 721/1788 [01:42<02:16,  7.82it/s]
{'loss': 0.5568, 'grad_norm': 4.186524868011475, 'learning_rate': 1.1991051454138702
e-05, 'epoch': 1.21}
 41%|████      | 731/1788 [01:43<02:15,  7.83it/s]
{'loss': 0.5089, 'grad_norm': 8.118603706359863, 'learning_rate': 1.1879194630872484
e-05, 'epoch': 1.22}
 41%|████▏     | 741/1788 [01:45<02:22,  7.35it/s]
{'loss': 0.4278, 'grad_norm': 14.69310474395752, 'learning_rate': 1.1767337807606265
e-05, 'epoch': 1.24}
 42%|████▏     | 751/1788 [01:46<02:19,  7.44it/s]
{'loss': 0.4314, 'grad_norm': 1.5075023174285889, 'learning_rate': 1.165548098434004
7e-05, 'epoch': 1.26}
 43%|████▎     | 761/1788 [01:47<02:18,  7.42it/s]
{'loss': 0.2581, 'grad_norm': 3.2874670028686523, 'learning_rate': 1.154362416107382
8e-05, 'epoch': 1.28}
 43%|████▎     | 771/1788 [01:49<02:16,  7.44it/s]
{'loss': 0.3388, 'grad_norm': 8.202762603759766, 'learning_rate': 1.1431767337807606
e-05, 'epoch': 1.29}
 44%|████▎     | 781/1788 [01:50<02:16,  7.40it/s]
{'loss': 0.5647, 'grad_norm': 1.4937976598739624, 'learning_rate': 1.131991051454138
8e-05, 'epoch': 1.31}
 44%|████▍     | 791/1788 [01:51<02:14,  7.44it/s]
```

```
{'loss': 0.4216, 'grad_norm': 20.69643783569336, 'learning_rate': 1.1208053691275169
e-05, 'epoch': 1.33}
 45%|████▌     | 801/1788 [01:53<02:13,  7.40it/s]
{'loss': 0.5124, 'grad_norm': 14.973134994506836, 'learning_rate': 1.109619686800895
e-05, 'epoch': 1.34}
 45%|████▌     | 811/1788 [01:54<02:13,  7.30it/s]
{'loss': 0.4322, 'grad_norm': 12.696832656860352, 'learning_rate': 1.098434004474273
e-05, 'epoch': 1.36}
 46%|████▌     | 821/1788 [01:55<02:15,  7.12it/s]
{'loss': 0.5621, 'grad_norm': 7.98099946975708, 'learning_rate': 1.0872483221476512e
-05, 'epoch': 1.38}
 46%|████▋     | 831/1788 [01:57<02:15,  7.04it/s]
{'loss': 0.7683, 'grad_norm': 20.491283416748047, 'learning_rate': 1.076062639821029
1e-05, 'epoch': 1.39}
 47%|████▋     | 841/1788 [01:58<02:15,  6.99it/s]
{'loss': 0.6507, 'grad_norm': 26.73902702331543, 'learning_rate': 1.0648769574944073
e-05, 'epoch': 1.41}
 48%|████▊     | 851/1788 [02:00<02:05,  7.46it/s]
{'loss': 0.4752, 'grad_norm': 20.303936004638672, 'learning_rate': 1.053691275167785
3e-05, 'epoch': 1.43}
 48%|████▊     | 861/1788 [02:01<02:04,  7.47it/s]
{'loss': 0.6084, 'grad_norm': 23.245901107788086, 'learning_rate': 1.042505592841163
4e-05, 'epoch': 1.44}
 49%|████▊     | 871/1788 [02:02<02:01,  7.54it/s]
{'loss': 0.291, 'grad_norm': 1.8471051454544067, 'learning_rate': 1.0313199105145415
e-05, 'epoch': 1.46}
 49%|████▉     | 881/1788 [02:04<02:00,  7.55it/s]
{'loss': 0.6978, 'grad_norm': 11.809346199035645, 'learning_rate': 1.020134228187919
7e-05, 'epoch': 1.48}
 50%|████▉     | 891/1788 [02:05<01:58,  7.55it/s]
{'loss': 0.4585, 'grad_norm': 6.569423675537109, 'learning_rate': 1.0089485458612975
e-05, 'epoch': 1.49}
 50%|█████     | 901/1788 [02:06<01:57,  7.54it/s]
{'loss': 0.3851, 'grad_norm': 7.105768203735352, 'learning_rate': 9.977628635346756e
-06, 'epoch': 1.51}
 51%|█████     | 911/1788 [02:07<01:55,  7.57it/s]
{'loss': 0.4704, 'grad_norm': 3.06421160697937, 'learning_rate': 9.865771812080538e-
06, 'epoch': 1.53}
 52%|█████▏    | 921/1788 [02:09<01:55,  7.54it/s]
{'loss': 0.4217, 'grad_norm': 4.143369674682617, 'learning_rate': 9.753914988814318e
-06, 'epoch': 1.54}
 52%|█████▏    | 931/1788 [02:10<01:53,  7.58it/s]
{'loss': 0.5467, 'grad_norm': 32.32228469848633, 'learning_rate': 9.642058165548099e
-06, 'epoch': 1.56}
 53%|█████▎    | 941/1788 [02:11<01:52,  7.55it/s]
{'loss': 0.4548, 'grad_norm': 13.657382011413574, 'learning_rate': 9.530201342281879
e-06, 'epoch': 1.58}
 53%|█████▎    | 951/1788 [02:13<01:50,  7.58it/s]
{'loss': 0.4279, 'grad_norm': 6.9170708656311035, 'learning_rate': 9.41834451901566e
-06, 'epoch': 1.59}
 54%|█████▎    | 961/1788 [02:14<01:49,  7.58it/s]
```

```
{'loss': 0.5516, 'grad_norm': 1.4551987648010254, 'learning_rate': 9.306487695749442
e-06, 'epoch': 1.61}
 54%|█████▍    | 971/1788 [02:15<01:47,  7.57it/s]
{'loss': 0.5147, 'grad_norm': 35.655216217041016, 'learning_rate': 9.194630872483221
e-06, 'epoch': 1.63}
 55%|█████▍    | 981/1788 [02:17<01:47,  7.53it/s]
{'loss': 0.4391, 'grad_norm': 14.178787231445312, 'learning_rate': 9.082774049217003
e-06, 'epoch': 1.64}
 55%|█████▌   | 991/1788 [02:18<01:45,  7.54it/s]
{'loss': 0.3684, 'grad_norm': 4.906567096710205, 'learning_rate': 8.970917225950784e
-06, 'epoch': 1.66}
 56%|█████▌   | 1001/1788 [02:19<01:43,  7.57it/s]
{'loss': 0.3402, 'grad_norm': 6.864356994628906, 'learning_rate': 8.859060402684566e
-06, 'epoch': 1.68}
 57%|█████▋    | 1011/1788 [02:21<01:42,  7.57it/s]
{'loss': 0.3524, 'grad_norm': 19.31537628173828, 'learning_rate': 8.747203579418346e
-06, 'epoch': 1.69}
 57%|█████▋    | 1021/1788 [02:22<01:40,  7.61it/s]
{'loss': 0.356, 'grad_norm': 20.981658935546875, 'learning_rate': 8.635346756152127e
-06, 'epoch': 1.71}
 58%|█████▊    | 1031/1788 [02:23<01:39,  7.62it/s]
{'loss': 0.4699, 'grad_norm': 9.390032768249512, 'learning_rate': 8.523489932885907e
-06, 'epoch': 1.73}
 58%|█████▊    | 1041/1788 [02:25<01:38,  7.60it/s]
{'loss': 0.3384, 'grad_norm': 12.836348533630371, 'learning_rate': 8.411633109619688
e-06, 'epoch': 1.74}
 59%|█████▉    | 1051/1788 [02:26<01:37,  7.57it/s]
{'loss': 0.2604, 'grad_norm': 2.336068868637085, 'learning_rate': 8.299776286353468e
-06, 'epoch': 1.76}
 59%|█████▉    | 1061/1788 [02:27<01:36,  7.57it/s]
{'loss': 0.5198, 'grad_norm': 11.922728538513184, 'learning_rate': 8.18791946308725e
-06, 'epoch': 1.78}
 60%|█████▉    | 1071/1788 [02:28<01:34,  7.56it/s]
{'loss': 0.5303, 'grad_norm': 14.010147094726562, 'learning_rate': 8.07606263982103e
-06, 'epoch': 1.8}
 60%|██████   | 1081/1788 [02:30<01:33,  7.58it/s]
{'loss': 0.3566, 'grad_norm': 15.794836044311523, 'learning_rate': 7.96420581655481e
-06, 'epoch': 1.81}
 61%|██████   | 1091/1788 [02:31<01:31,  7.58it/s]
{'loss': 0.3635, 'grad_norm': 14.82154369354248, 'learning_rate': 7.85234899328859e-
06, 'epoch': 1.83}
 62%|██████▏   | 1101/1788 [02:32<01:30,  7.56it/s]
{'loss': 0.5759, 'grad_norm': 14.754202842712402, 'learning_rate': 7.740492170022372
e-06, 'epoch': 1.85}
 62%|██████▏   | 1111/1788 [02:34<01:29,  7.57it/s]
{'loss': 0.2096, 'grad_norm': 6.653132915496826, 'learning_rate': 7.6286353467561525
e-06, 'epoch': 1.86}
 63%|██████▎   | 1121/1788 [02:35<01:27,  7.63it/s]
{'loss': 0.4146, 'grad_norm': 5.9213972091674805, 'learning_rate': 7.516778523489934
e-06, 'epoch': 1.88}
 63%|██████▎   | 1131/1788 [02:36<01:26,  7.59it/s]
```

```
{'loss': 0.3054, 'grad_norm': 17.806623458862305, 'learning_rate': 7.404921700223714
e-06, 'epoch': 1.9}
 64%|██████▍   | 1141/1788 [02:38<01:25,  7.58it/s]
{'loss': 0.3256, 'grad_norm': 7.20403528213501, 'learning_rate': 7.293064876957495e-
06, 'epoch': 1.91}
 64%|██████▍   | 1151/1788 [02:39<01:24,  7.57it/s]
{'loss': 0.6004, 'grad_norm': 9.81818675994873, 'learning_rate': 7.181208053691276e-
06, 'epoch': 1.93}
 65%|██████▍   | 1161/1788 [02:40<01:22,  7.57it/s]
{'loss': 0.2666, 'grad_norm': 5.646169662475586, 'learning_rate': 7.069351230425056e
-06, 'epoch': 1.95}
 65%|██████▌   | 1171/1788 [02:41<01:21,  7.61it/s]
{'loss': 0.5704, 'grad_norm': 10.271567344665527, 'learning_rate': 6.957494407158837
e-06, 'epoch': 1.96}
 66%|██████▌   | 1181/1788 [02:43<01:20,  7.56it/s]
{'loss': 0.3296, 'grad_norm': 16.328262329101562, 'learning_rate': 6.845637583892618
e-06, 'epoch': 1.98}
 67%|██████▋   | 1191/1788 [02:44<01:18,  7.61it/s]
{'loss': 0.3444, 'grad_norm': 21.94635581970215, 'learning_rate': 6.733780760626398e
-06, 'epoch': 2.0}

 67%|██████▋   | 1192/1788 [02:49<01:13,  8.06it/s]
{'eval_loss': 0.5693091154098511, 'eval_runtime': 4.3975, 'eval_samples_per_second':
271.06, 'eval_steps_per_second': 33.883, 'epoch': 2.0}
 67%|██████▋   | 1201/1788 [02:51<02:13,  4.40it/s]
{'loss': 0.2895, 'grad_norm': 7.963081359863281, 'learning_rate': 6.6219239373601796
e-06, 'epoch': 2.01}
 68%|██████▊   | 1211/1788 [02:52<01:17,  7.47it/s]
{'loss': 0.3568, 'grad_norm': 18.53574562072754, 'learning_rate': 6.51006711409396e-
06, 'epoch': 2.03}
 68%|██████▊   | 1221/1788 [02:54<01:16,  7.41it/s]
{'loss': 0.106, 'grad_norm': 2.140268564224243, 'learning_rate': 6.398210290827741e-
06, 'epoch': 2.05}
 69%|██████▉   | 1231/1788 [02:55<01:13,  7.62it/s]
{'loss': 0.2895, 'grad_norm': 7.642291069030762, 'learning_rate': 6.286353467561522e
-06, 'epoch': 2.06}
 69%|██████▉   | 1241/1788 [02:56<01:11,  7.64it/s]
{'loss': 0.3875, 'grad_norm': 9.545188903808594, 'learning_rate': 6.174496644295303e
-06, 'epoch': 2.08}
 70%|██████▉   | 1251/1788 [02:57<01:10,  7.62it/s]
{'loss': 0.3295, 'grad_norm': 0.6250635385513306, 'learning_rate': 6.062639821029083
e-06, 'epoch': 2.1}
 71%|███████   | 1261/1788 [02:59<01:09,  7.57it/s]
{'loss': 0.2133, 'grad_norm': 0.7527950406074524, 'learning_rate': 5.950782997762864
e-06, 'epoch': 2.11}
 71%|███████   | 1271/1788 [03:00<01:07,  7.62it/s]
{'loss': 0.1767, 'grad_norm': 4.728277683258057, 'learning_rate': 5.8389261744966455
e-06, 'epoch': 2.13}
 72%|███████▏  | 1281/1788 [03:01<01:06,  7.57it/s]
{'loss': 0.1666, 'grad_norm': 19.361522674560547, 'learning_rate': 5.727069351230425
e-06, 'epoch': 2.15}
```

```
 72%|███████▏  | 1291/1788 [03:03<01:05,  7.61it/s]
{'loss': 0.1232, 'grad_norm': 6.172418117523193, 'learning_rate': 5.615212527964207e
-06, 'epoch': 2.16}
 73%|███████▎  | 1301/1788 [03:04<01:04,  7.59it/s]
{'loss': 0.2464, 'grad_norm': 14.468454360961914, 'learning_rate': 5.503355704697987
e-06, 'epoch': 2.18}
 73%|███████▎  | 1311/1788 [03:05<01:02,  7.59it/s]
{'loss': 0.289, 'grad_norm': 27.248329162597656, 'learning_rate': 5.391498881431768e
-06, 'epoch': 2.2}
 74%|███████▍  | 1321/1788 [03:07<01:01,  7.63it/s]
{'loss': 0.0963, 'grad_norm': 1.038118600845337, 'learning_rate': 5.2796420581655485
e-06, 'epoch': 2.21}
 74%|███████▍  | 1331/1788 [03:08<01:00,  7.61it/s]
{'loss': 0.2455, 'grad_norm': 6.814629554748535, 'learning_rate': 5.16778523489933e-
06, 'epoch': 2.23}
 75%|███████▌  | 1341/1788 [03:09<00:58,  7.61it/s]
{'loss': 0.2124, 'grad_norm': 22.291162490844727, 'learning_rate': 5.055928411633111e
-06, 'epoch': 2.25}
 76%|███████▌  | 1351/1788 [03:10<00:57,  7.64it/s]
{'loss': 0.2986, 'grad_norm': 15.901519775390625, 'learning_rate': 4.944071588366891
e-06, 'epoch': 2.27}
 76%|███████▌  | 1361/1788 [03:12<00:55,  7.64it/s]
{'loss': 0.3336, 'grad_norm': 18.53882598876953, 'learning_rate': 4.832214765100672e
-06, 'epoch': 2.28}
 77%|███████▋  | 1371/1788 [03:13<00:53,  7.73it/s]
{'loss': 0.1223, 'grad_norm': 16.33506965637207, 'learning_rate': 4.720357941834452e
-06, 'epoch': 2.3}
 77%|███████▋  | 1381/1788 [03:14<00:53,  7.63it/s]
{'loss': 0.5352, 'grad_norm': 17.45832061767578, 'learning_rate': 4.608501118568233e
-06, 'epoch': 2.32}
 78%|███████▊  | 1391/1788 [03:16<00:52,  7.61it/s]
{'loss': 0.1118, 'grad_norm': 26.06778907775879, 'learning_rate': 4.4966442953020135
e-06, 'epoch': 2.33}
 78%|███████▊  | 1401/1788 [03:17<00:50,  7.62it/s]
{'loss': 0.2417, 'grad_norm': 0.7317590117454529, 'learning_rate': 4.384787472035795
e-06, 'epoch': 2.35}
 79%|███████▉  | 1411/1788 [03:18<00:49,  7.61it/s]
{'loss': 0.2907, 'grad_norm': 0.21307842433452606, 'learning_rate': 4.27293064876957
6e-06, 'epoch': 2.37}
 79%|███████▉  | 1421/1788 [03:20<00:48,  7.62it/s]
{'loss': 0.3316, 'grad_norm': 0.30909669399261475, 'learning_rate': 4.16107382550335
6e-06, 'epoch': 2.38}
 80%|████████  | 1431/1788 [03:21<00:47,  7.59it/s]
{'loss': 0.3307, 'grad_norm': 13.665131568908691, 'learning_rate': 4.049217002237137
e-06, 'epoch': 2.4}
 81%|████████  | 1441/1788 [03:22<00:47,  7.33it/s]
{'loss': 0.1685, 'grad_norm': 29.3431472778320З, 'learning_rate': 3.937360178970917e
-06, 'epoch': 2.42}
 81%|████████  | 1451/1788 [03:24<00:46,  7.22it/s]
{'loss': 0.2479, 'grad_norm': 3.717780351638794, 'learning_rate': 3.825503355704698e
-06, 'epoch': 2.43}
```

```
 82%|████████▏ | 1461/1788 [03:25<00:44,  7.27it/s]
{'loss': 0.2678, 'grad_norm': 3.4237282276153564, 'learning_rate': 3.713646532438479
e-06, 'epoch': 2.45}
 82%|████████▏ | 1471/1788 [03:26<00:43,  7.22it/s]
{'loss': 0.1644, 'grad_norm': 6.175680160522461, 'learning_rate': 3.6017897091722596
e-06, 'epoch': 2.47}
 83%|████████▎ | 1481/1788 [03:28<00:43,  7.11it/s]
{'loss': 0.4704, 'grad_norm': 0.6465878486633301, 'learning_rate': 3.489932885906040
7e-06, 'epoch': 2.48}
 83%|████████▎ | 1491/1788 [03:29<00:39,  7.59it/s]
{'loss': 0.5584, 'grad_norm': 3.1046481132507324, 'learning_rate': 3.378076062639821
3e-06, 'epoch': 2.5}
 84%|████████▍ | 1501/1788 [03:30<00:38,  7.54it/s]
{'loss': 0.3183, 'grad_norm': 7.902535438537598, 'learning_rate': 3.266219239373602e
-06, 'epoch': 2.52}
 85%|████████▍ | 1511/1788 [03:32<00:36,  7.59it/s]
{'loss': 0.1366, 'grad_norm': 5.708117961883545, 'learning_rate': 3.154362416107383e
-06, 'epoch': 2.53}
 85%|████████▌| 1521/1788 [03:33<00:35,  7.60it/s]
{'loss': 0.3566, 'grad_norm': 26.593942642211914, 'learning_rate': 3.042505592841163
5e-06, 'epoch': 2.55}
 86%|████████▌| 1531/1788 [03:34<00:33,  7.58it/s]
{'loss': 0.3886, 'grad_norm': 0.3174760043621063, 'learning_rate': 2.930648769574944
e-06, 'epoch': 2.57}
 86%|████████▌| 1541/1788 [03:36<00:32,  7.58it/s]
{'loss': 0.4024, 'grad_norm': 14.427067756652832, 'learning_rate': 2.818791946308724
7e-06, 'epoch': 2.58}
 87%|████████▋ | 1551/1788 [03:37<00:31,  7.62it/s]
{'loss': 0.2217, 'grad_norm': 1.2595751285552979, 'learning_rate': 2.706935123042505
7e-06, 'epoch': 2.6}
 87%|████████▋ | 1561/1788 [03:38<00:29,  7.80it/s]
{'loss': 0.0773, 'grad_norm': 3.5950229167938232, 'learning_rate': 2.595078299776286
3e-06, 'epoch': 2.62}
 88%|████████▊ | 1571/1788 [03:39<00:27,  7.78it/s]
{'loss': 0.2563, 'grad_norm': 0.594588577747345, 'learning_rate': 2.4832214765100673
e-06, 'epoch': 2.63}
 88%|████████▊ | 1581/1788 [03:41<00:26,  7.79it/s]
{'loss': 0.3177, 'grad_norm': 0.9153674840927124, 'learning_rate': 2.371364653243848
e-06, 'epoch': 2.65}
 89%|████████▉ | 1591/1788 [03:42<00:25,  7.81it/s]
{'loss': 0.2228, 'grad_norm': 0.45025837421417236, 'learning_rate': 2.25950782997762
9e-06, 'epoch': 2.67}
 90%|████████▉ | 1601/1788 [03:43<00:24,  7.76it/s]
{'loss': 0.2605, 'grad_norm': 0.5239439606666565, 'learning_rate': 2.147651006711409
6e-06, 'epoch': 2.68}
 90%|█████████| 1611/1788 [03:45<00:22,  7.81it/s]
{'loss': 0.3242, 'grad_norm': 2.441354990005493, 'learning_rate': 2.03579418344519e-
06, 'epoch': 2.7}
 91%|█████████| 1621/1788 [03:46<00:21,  7.81it/s]
{'loss': 0.1889, 'grad_norm': 11.781035423278809, 'learning_rate': 1.923937360178970
8e-06, 'epoch': 2.72}
```

```
 91%|█████████ | 1631/1788 [03:47<00:20,  7.78it/s]
{'loss': 0.5128, 'grad_norm': 23.535724639892578, 'learning_rate': 1.812080536912751
8e-06, 'epoch': 2.73}
 92%|█████████▏| 1641/1788 [03:48<00:18,  7.79it/s]
{'loss': 0.1468, 'grad_norm': 0.38081789016723633, 'learning_rate': 1.70022371364653
26e-06, 'epoch': 2.75}
 92%|█████████▏| 1651/1788 [03:50<00:17,  7.80it/s]
{'loss': 0.2591, 'grad_norm': 32.34906005859375, 'learning_rate': 1.5883668903803134
e-06, 'epoch': 2.77}
 93%|█████████▎| 1661/1788 [03:51<00:16,  7.79it/s]
{'loss': 0.4156, 'grad_norm': 24.305248260498047, 'learning_rate': 1.476510067114094
2e-06, 'epoch': 2.79}
 93%|█████████▎| 1671/1788 [03:52<00:15,  7.74it/s]
{'loss': 0.1721, 'grad_norm': 40.854957580566406, 'learning_rate': 1.364653243847874
8e-06, 'epoch': 2.8}
 94%|█████████▍| 1681/1788 [03:54<00:13,  7.79it/s]
{'loss': 0.2264, 'grad_norm': 10.7556734085083, 'learning_rate': 1.2527964205816557e
-06, 'epoch': 2.82}
 95%|█████████▍| 1691/1788 [03:55<00:12,  7.83it/s]
{'loss': 0.1463, 'grad_norm': 0.38049405813217163, 'learning_rate': 1.14093959731543
63e-06, 'epoch': 2.84}
 95%|█████████▌| 1701/1788 [03:56<00:11,  7.80it/s]
{'loss': 0.2281, 'grad_norm': 2.2980167865753174, 'learning_rate': 1.029082774049217
e-06, 'epoch': 2.85}
 96%|█████████▌| 1711/1788 [03:57<00:09,  7.75it/s]
{'loss': 0.3116, 'grad_norm': 7.36637544631958, 'learning_rate': 9.172259507829978e-
07, 'epoch': 2.87}
 96%|█████████▋| 1721/1788 [03:59<00:08,  7.78it/s]
{'loss': 0.1389, 'grad_norm': 39.23965072631836, 'learning_rate': 8.053691275167786e
-07, 'epoch': 2.89}
 97%|█████████▋| 1731/1788 [04:00<00:07,  7.77it/s]
{'loss': 0.1764, 'grad_norm': 14.330676078796387, 'learning_rate': 6.935123042505594
e-07, 'epoch': 2.9}
 97%|█████████▋| 1741/1788 [04:01<00:06,  7.78it/s]
{'loss': 0.2763, 'grad_norm': 5.134561061859131, 'learning_rate': 5.816554809843401e
-07, 'epoch': 2.92}
 98%|█████████▊| 1751/1788 [04:03<00:04,  7.76it/s]
{'loss': 0.2453, 'grad_norm': 0.5734639167785645, 'learning_rate': 4.697986577181208
7e-07, 'epoch': 2.94}
 98%|█████████▊| 1761/1788 [04:04<00:03,  7.77it/s]
{'loss': 0.09, 'grad_norm': 1.4031435251235962, 'learning_rate': 3.579418344519016e-
07, 'epoch': 2.95}
 99%|█████████▉| 1771/1788 [04:05<00:02,  7.78it/s]
{'loss': 0.2017, 'grad_norm': 13.268450736999512, 'learning_rate': 2.460850111856823
4e-07, 'epoch': 2.97}
100%|█████████▉| 1781/1788 [04:06<00:00,  7.75it/s]
{'loss': 0.3939, 'grad_norm': 17.370460510253906, 'learning_rate': 1.342281879194631
e-07, 'epoch': 2.99}

100%|██████████| 1788/1788 [04:12<00:00,  8.31it/s]
```

```
{'eval_loss': 0.6155669689178467, 'eval_runtime': 4.3926, 'eval_samples_per_second':
271.368, 'eval_steps_per_second': 33.921, 'epoch': 3.0}
```

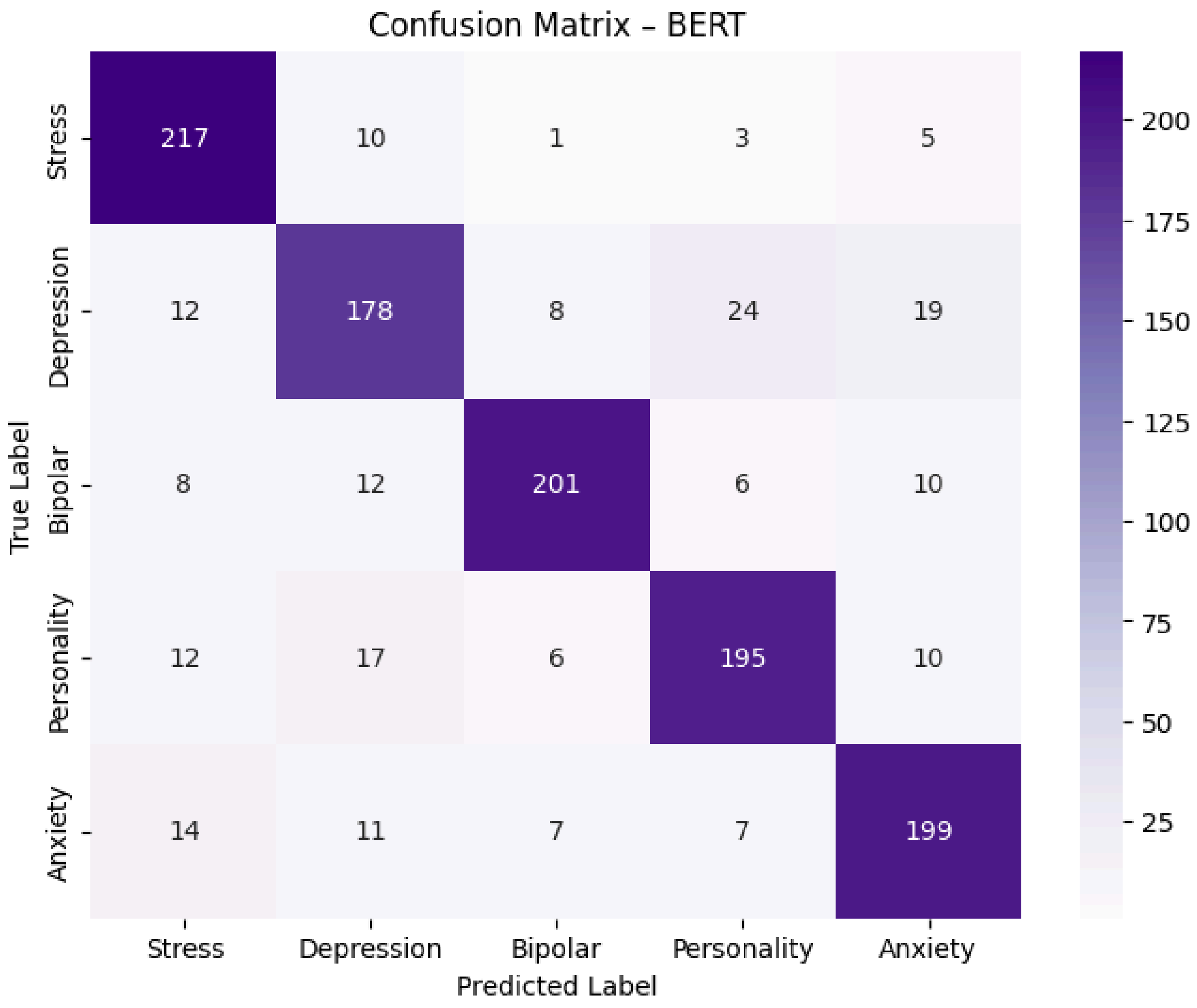
```
100%|██████████| 1788/1788 [04:13<00:00,  7.05it/s]
{'train_runtime': 253.5013, 'train_samples_per_second': 56.39, 'train_steps_per_seco
nd': 7.053, 'train_loss': 0.5452078478181656, 'epoch': 3.0}
```

Out[18]:
```
TrainOutput(global_step=1788, training_loss=0.5452078478181656, metrics={'train_ru
ntime': 253.5013, 'train_samples_per_second': 56.39, 'train_steps_per_second': 7.0
53, 'total_flos': 1880636923261440.0, 'train_loss': 0.5452078478181656, 'epoch':
3.0})
```

In [19]:
```python
# Get predictions
preds = trainer.predict(test_dataset)
y_pred_bert = np.argmax(preds.predictions, axis=1)

# Classification Report
print("BERT Classification Report:")
print(classification_report(y_test, y_pred_bert, digits=3))
```

```
100%|██████████| 149/149 [00:04<00:00, 32.58it/s]
BERT Classification Report:
              precision    recall  f1-score   support

           0      0.825     0.919     0.870       236
           1      0.781     0.739     0.759       241
           2      0.901     0.848     0.874       237
           3      0.830     0.812     0.821       240
           4      0.819     0.836     0.827       238

    accuracy                          0.831      1192
   macro avg      0.831     0.831     0.830      1192
weighted avg      0.831     0.831     0.830      1192
```

In [20]:
```python
cm = confusion_matrix(y_test, y_pred_bert)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Purples',
            xticklabels=['Stress', 'Depression', 'Bipolar', 'Personality', 'Anxiety
            yticklabels=['Stress', 'Depression', 'Bipolar', 'Personality', 'Anxiety
plt.title('Confusion Matrix - BERT')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



Comparison and Evaluation

1. Model Architecture & Resource needs
   - Logistic regression is lighlightweight and trains within seconds on a CPU.
   - BERT, has over 100 million parameters, requiring GPU acceleration and much longer training times.
2. Hyperparameter Compexity
   - Logistic Regression requires tuning of only a few hyperparameters such as regularization strength.
   - BERT requires tuning of learning rate, batch size, number of epochs, and optionally weight decay and warm-up step increasing optimization complexity.
3. Feature Engineering
   - Logistic Regression depends heavily on manual feature extraction via TF-IDF.
   - BERT uses contextualised embeddings, automatically learning semantic representations.
4. Real-Time Applicability

    - Logistic Regression is more suited for deployment in low-resource or latency-sensitive environments.
    - BERT, though more accurate, introduces computational latency and higher memory demand, which can limit real-time applications without further optimization.
5. Robustness to Class Imbalance

    - BERT handled minority classes such as bipolar disorder and personality disorder more evenly, while Logistic Regression showed slight skew in recall, indicating BERT's superior contextual modeling may help mitigate some effects of class imbalance without explicit reweighting.

8. Programming Style

All code written in the course of this project was created in Python 3 and maintained in a single, well-documented Jupyter notebook to make it easier to read, trace, and modularize. Code is structured to move in a rational order from data loading and pre-processing to model training, testing, and visualizing the output.

Libraries and Tools Used

- scikit-learn
    - Used to train the baseline and traditional statistical models (Multinomial Naive Bayes and Logistic Regression) and the evaluation metrics like accuracy, precision, recall, and F1-score.
- transformers (Hugging Face)
    - Used to train and fine-tune the BertForSequenceClassification model. The Trainer and TrainingArguments APIs were utilized for optimal training, logging, and evaluation.
- pandas and numpy
    - Used for dataset operations and numerical computations.
- matplotlib and seaborn
    - For plotting confusion matrices, bar plots, and other visual comparisons between models.
- nltk
    - For preprocessing actions like tokenization, stopword removal, and lemmatization based on WordNet with POS tagging.

Code Documentation and Rationale

Inline comments are provided throughout the notebook to:

- Explain preprocessing choices, e.g., using WordNet lemmatization to normalize words but preserving grammatical context.

- Rationalize model selection: Logistic Regression was chosen as a simple, interpretable statistical classifier, and BERT was selected as one of the best-performing embedding-based models capable of learning complex semantics in Reddit posts.
- Clarify parameter settings:
  - Logistic Regression was grid-searched over C (regularization strength).
  - BERT was trained with Trainer with 3 epochs, learning rate of 2e-5, batch size of 8, and AdamW optimizer with linear learning rate scheduler.
  - These settings were informed by literature best practices and empirical testing.

Coding Best Practices Implemented

- Descriptive names for descriptive variables (y_pred_bert, tfidf_vectorizer, training_args)
- No "hard-coded paths" or "magic numbers" utilized
- Modular code segments to re-use (e.g., evaluation wrapped in functions)
- Visualizations with titles, axis labels, and legends

Reproducibility

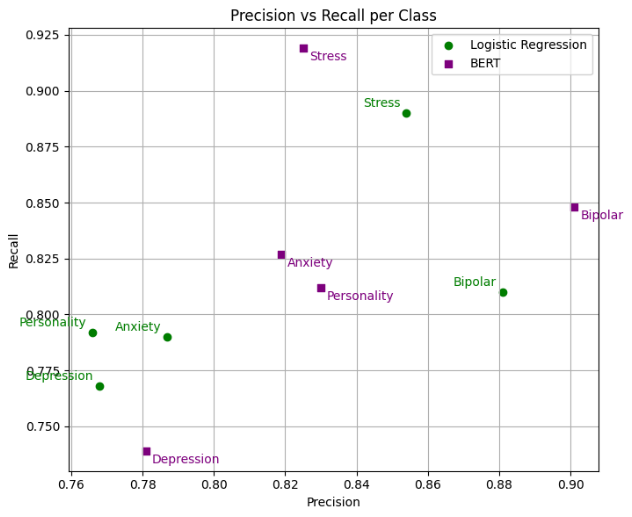
- All the random seeds (numpy, torch, and sklearn) were set to ensure reproducible results across runs.
- The pipeline is very easily reproducible with a new dataset or with varying hyperparameters.

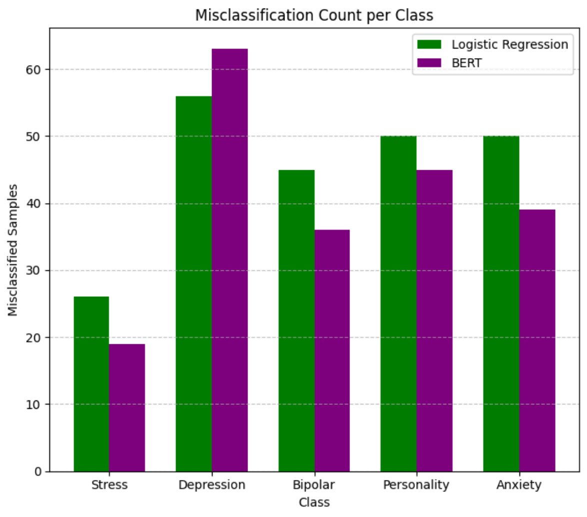
III. Conclusions

9. Performance Analysis & Comparative Discussion

To assess model performance across the five mental health categories (Stress, Depression, Bipolar Disorder, Personality Disorder, and Anxiety), three models were evaluated: Multinomial Naive Bayes (baseline), Logistic Regression (tuned statistical model), and BERT (embedding-based deep learning model). Evaluation was based on accuracy, macro-averaged precision, recall, and F1-score, along with confusion matrix visualizations.

In [22]:
```
# class labels
classes = ['Stress', 'Depression', 'Bipolar', 'Personality', 'Anxiety']
x = np.arange(len(classes))

# lr metrics
precision_lr = [0.854, 0.768, 0.881, 0.766, 0.787]
recall_lr =    [0.890, 0.768, 0.810, 0.792, 0.790]
cm_lr = np.array([
    [210, 7, 4, 5, 10],
    [8, 185, 11, 26, 11],
    [7, 11, 192, 14, 13],
    [7, 21, 5, 190, 17],
    [14, 17, 6, 13, 188]
])

# BERT metrics
precision_bert = [0.825, 0.781, 0.901, 0.830, 0.819]
recall_bert =    [0.919, 0.739, 0.848, 0.812, 0.827]
cm_bert = np.array([
    [217, 10, 1, 3, 5],
    [12, 178, 8, 24, 19],
    [8, 12, 201, 6, 10],
    [12, 17, 6, 195, 10],
    [14, 11, 7, 7, 199]
])

# misclassification counts per true class
misclassified_lr = [sum(row) - row[i] for i, row in enumerate(cm_lr)]
misclassified_bert = [sum(row) - row[i] for i, row in enumerate(cm_bert)]

# plot side by side
fig, axs = plt.subplots(1, 2, figsize=(14, 6))

# prec
axs[0].scatter(precision_lr, recall_lr, marker='o', color='green', label='Logistic
axs[0].scatter(precision_bert, recall_bert, marker='s', color='purple', label='BERT
for i, label in enumerate(classes):
    axs[0].annotate(label, (precision_lr[i], recall_lr[i]), xytext=(-5, 5), textcoo
    axs[0].annotate(label, (precision_bert[i], recall_bert[i]), xytext=(5, -10), te
axs[0].set_xlabel('Precision')
axs[0].set_ylabel('Recall')
axs[0].set_title('Precision vs Recall per Class')
axs[0].grid(True)
axs[0].legend()

# misclassification bar chart
width = 0.35
axs[1].bar(x - width/2, misclassified_lr, width, label='Logistic Regression', color
axs[1].bar(x + width/2, misclassified_bert, width, label='BERT', color='purple')
axs[1].set_xlabel('Class')
axs[1].set_ylabel('Misclassified Samples')
axs[1].set_title('Misclassification Count per Class')
axs[1].set_xticks(x)
axs[1].set_xticklabels(classes)
axs[1].legend()
axs[1].grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```





Quantitative Comparison

- BERT achieved the highest overall accuracy (83.1%) and macro F1-score (0.830), outperforming Logistic Regression (81.0%, 0.810) and Multinomial Naive Bayes (74.0%, 0.74).
- In all five classes, BERT demonstrated improved balance between precision and recall, especially in bipolar disorder (F1: 0.874 vs 0.844 in LR) and anxiety (F1: 0.827 vs 0.788 in LR).
- The greatest gap appeared in the recall for bipolar and anxiety, where BERT successfully reduced false negatives, suggesting better contextual understanding of language markers tied to those conditions.

Advantages & Disadvantages

1. Logistic Regression (Statistical)

- Advantages
  - Fast to train, highly interpretable
  - Performs well with well-separated features (e.g., stress, bipolar)
  - Minimal hardware requirements
- Disadvantages
  - Relies on sparse, non-contextual features
  - Struggles with semantically similar classes like depression and anxiety
  - Needs manual preprocessing and feature tuning

2. BERT (Embedding-based)

- Advantages
  - Automatically learns contextual semantics
  - Superior performance in overlapping classes
  - More robust to informal or varied linguistic expression
- Disadvantages

    - Computationally expensive to train and fine-tune
    - Less interpretable ("black box")
    - Requires GPU resources for efficiency

Application Scenarios

- Logistic Regression
    - This model finds applications in contexts where speed, simplicity, and interpretability are a concern and resources are scarce. For instance, it could be applied in low-resource mobile health apps or browser-based mental health self-assessment interfaces that need to operate offline or under restricted computational capacities. Logistic Regression models are eas to debug and explain to stakeholder who are not technical, and hence suit best in clinical support environments where visibility is crucial.
- BERT (Embedding-Based Model)
    - BERT is better optimized for real-time mental health monitoring use cases such as automated bots or crisis identification tools that require more in-depth contextual understanding of nuanced and dense text. Its ability to distinguish between semantically close classes (e.g., depression and anxiety) makes it a good candidate for clinical decision support use cases, where misclassification could have a negative effect on user well-being. BERT can also be applied to longitudinal post user analysis to track over time changes in mental states, something statistical models are not well suited for.
- Hybrid or Tiered Systems
    - In practice, these models can be combined in a two-stage process, with Logistic Regression as a low-cost, fast first pass to flag potentially relevant cases. Posts which fall into borderline or high-risk groups can then be passed through a BERT-based model for further examination. This approach has the tradeoff between speed and accuracy, making it stable enough to deploy at scale.

10. Project Summary and Reflections

This project has provided an in-depth and hands-on experience in building, comparing, and evaluating text classification models in the delicate and socially important domain of detecting mental health. By applying both traditional statistical techniques and modern deep learning frameworks in classifying Reddit posts into five classes of mental health, I have gained a deeper understanding of how various modeling approaches handle real-world language data.

In practice, Logistic Regression was a useful and interpretable model that was especially useful in environments with limited resources or for purposes that required transparency, such as in clinical decision support. While its accuracy, though resilient, plateaued in distinguishing semantically close mental illnesses like depression, anxiety, and personality disorder,.

The BERT model did significantly better than its statistical counterpart in both macro-averaged measures and class-level consistency. Its contextual awareness and emotional intelligence in detecting subtle cues within users' posts make it especially meritorious in cases of challenging classification wherein mental health indicators are not easily separable. Nevertheless, the trade-off for this added performance was higher computational costs, longer training times, and lower interpretability.

One of the most valuable aspects learned from this project was the trade-offs between performance, efficiency, and explainability. Although BERT may be the clear winner in terms of predictive power, the deployment and implementation context greatly influences which model is truly "best." This reinforced how model applicability trumps raw accuracy.

The solution deployed here is also extendable to other domain-specific areas where language plays a central role in classification. For instance, the same architectures can be used to detect suicidal intent, risk of drug abuse, or even fake news detection on social media. The pipeline can further be deployed in multilingual settings or fine-tuned against domain-specific datasets like clinical reports or therapy sessions.

Improvement Suggestions and Future Research

- Merging with explainability libraries such as SHAP or LIME to boost trust in deep models such as BERT is recommended.
- Fine-tuning domain-specific versions of BERT (e.g., MentalBERT or ClinicalBERT) can also improve performance on mental health data.
- Having non-mental-health ("neutral") posts in the data would allow for the development of binary or multilabel classifiers with broader screening uses.

In conclusion, this project not only demonstrated the capabilities and limitations of different model families, but also showed that NLP could meaningfully be used to contribute to mental health sensitivity and emotional intervention in online communities.

11. References

1. World Health Organization, World Health Organization, www.who.int/news-room/fact-sheets/detail/. Accessed 28 June 2025.
2. Pennebaker, James W et al. “Psychological aspects of natural language. use: our words, our selves.” Annual review of psychology vol. 54 (2003): 547-77. doi:10.1146/annurev.psych.54.101601.145041
3. Al-Mosaiwi, Mohammed, and Tom Johnstone. “In an Absolute State: Elevated Use of Absolutist Words Is a Marker Specific to Anxiety, Depression, and Suicidal Ideation.”

Clinical psychological science : a journal of the Association for Psychological Science vol. 6,4 (2018): 529-542. doi:10.1177/2167702617747074

4. Coppersmith, Glen, et al. "Quantifying Mental Health Signals in Twitter." Proceedings of the Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality, 2014, https://doi.org/10.3115/v1/w14-3207.
5. Chancellor, Stevie, and Munmun De Choudhury. "Methods in predictive techniques for mental health status on social media: a critical review." NPJ digital medicine vol. 3 43. 24 Mar. 2020, doi:10.1038/s41746-020-0233-7
6. Guntuku, Sharath Chandra, et al. "Studying expressions of loneliness in individuals using Twitter: An observational study." BMJ Open, vol. 9, no. 11, Nov. 2019, https://doi.org/10.1136/bmjopen-2019-030355.
7. Ghoshal, Neel. "Reddit Mental Health Data." Kaggle, 14 July 2023, www.kaggle.com/datasets/neelghoshal/reddit-mental-health-data/data.
8. Devlin, Jacob, et al. "Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding." arXiv.Org, 24 May 2019, arxiv.org/abs/1810.04805.