

The model of choice for the competition is BERT. I have elected to use BERT. I have also attempted to use several various other models, such as **Logistic Regression, Naïve Bayes and Decision Trees/Random Forest** initially to strive for a better result, but ultimately all these came short of BERT that managed to achieve the highest F1-score metric and subsequent public score.

I had previously performed several pre-processing steps for the other models such as:

- Lowercasing: To convert all text to lowercase\
- Removal of punctuation and special characters
- Stop word removal

However, many of these are non-needed and may even prove detrimental to the predictive ability of the final BERT model by removing information that carries meaning, such as question marks or exclamation marks, or even loss of valuable contextual clues for sentiment analysis. I will go into each part below and highlight these steps accordingly.

```
# Extract fields from the nested structure
tweets_df["tweet_id"] = tweets_df["_source"].apply(lambda x: x["tweet"]["tweet_id"])
tweets_df["text"] = tweets_df["_source"].apply(lambda x: x["tweet"]["text"])
tweets_df["hashtags"] = tweets_df["_source"].apply(lambda x: x["tweet"]["hashtags"])

# Drop the original _source column
tweets_df = tweets_df.drop(columns=["_source"])

df = pd.merge(tweets_df, emotion_df, on="tweet_id", how="inner")
df.drop(columns=['_score', '_index', '_type', 'hashtags', '_crawldate'], inplace=True)
```

	tweet_id	text	emotion
0	0x376b20	People who post "add me on #Snapchat" must be ...	anticipation
1	0x2d5350	@brianklaas As we see, Trump is dangerous to #...	sadness
2	0x1cd5b0	Now ISSA is stalking Tasha 🤔🤔🤔 <LH>	fear
3	0x1d755c	@RISKshow @TheKevinAllison Thx for the BEST TI...	joy
4	0x2c91a8	Still waiting on those supplies Liscus. <LH>	anticipation
...
1455558	0x321566	I'm SO HAPPY!!! #NoWonder the name of this sho...	joy
1455559	0x38959e	In every circumstance I'd like to be thankful t...	joy
1455560	0x2cbca6	there's currently two girls walking around the...	joy
1455561	0x24faed	Ah, corporate life, where you can date <LH> us...	joy
1455562	0x34be8c	Blessed to be living #Sundayvibes <LH>	joy

1455563 rows × 3 columns

Initial fields from the nested structure of the .json dataset was retrieved and merged with the respective emotions to form the basic dataset from which a train-test split and model training is done.

```

texts = df["text"].tolist() # Raw tweet text
emotions = df["emotion"].astype("category").cat.codes.tolist() # Encode emotions as integers

# Split into training and validation sets
train_texts, val_texts, train_labels, val_labels = train_test_split(
    texts, emotions, test_size=0.2, random_state=42)

# Step 2: Initialize BERT Tokenizer
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")

# Tokenize the data
def tokenize_function(examples):
    return tokenizer(examples["text"], padding="max_length", truncation=True, max_length=128)

tokenizer_config.json: 0%|          | 0.00/48.0 [00:00<?, ?B/s]
vocab.txt: 0%|          | 0.00/232k [00:00<?, ?B/s]
tokenizer.json: 0%|          | 0.00/466k [00:00<?, ?B/s]
config.json: 0%|          | 0.00/570 [00:00<?, ?B/s]

```

BERT is imported, and the BertTokenizer with the ‘bert-base-uncased’ argument is used to tokenize the data. This argument automatically converts the input text to lowercase, reducing the need to manually lowercase the text. Additionally, the tokenizer is able to implicitly handle punctuation, special characters, spaces, and stop-words without needing prior cleaning.

To note that feature extractions commonly associated with text processing such as Bag of Words, TD-IDF or Word Embeddings like Word2Vec were not used with the BERT model, as the BERT model has been trained on a massive corpus of text and uses dense, **contextual word embeddings** that understand the meaning of words based on their position and surrounding words. Hence, further feature extraction would not contribute additional contextual understanding and was omitted.

```

# Step 3: Format Dataset for PyTorch
# Set dataset format
train_dataset.set_format(type="torch", columns=["input_ids", "attention_mask", "labels"])
val_dataset.set_format(type="torch", columns=["input_ids", "attention_mask", "labels"])



# Step 4: Load BERT Model
model = BertForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=8) # 8 classes

```

The dataset objects (trainset and testset) are formatted to be compatible with PyTorch by converting them to PyTorch tensors such that they may be fed into the model without additional subsequent conversion. The BERT model is loaded with its corresponding 8 emotion classes.

```
# Step 5: Define Training Arguments
training_args = TrainingArguments(
    output_dir="./results",           # Output directory
    evaluation_strategy="epoch",      # Evaluate every epoch
    per_device_train_batch_size=8,    # Batch size for training
    per_device_eval_batch_size=8,     # Batch size for evaluation
    num_train_epochs=2,               # Number of epochs (modified to 2)
    weight_decay=0.01,               # Strength of weight decay
    logging_dir="./logs",             # Directory for logging
    logging_steps=10,
    save_strategy="epoch",            # Save the model at each epoch
    save_total_limit=2,               # Limit the number of saved checkpoints
    report_to=[]                      # Disable reporting to external systems
)
```

The training arguments are set here, and a point to note is I have set the ‘num_train_epochs’ argument to 2 instead of the conventional 3. This is because in previous runs of my model, I have found that validation loss is lowest at the 2nd epoch before climbing up again at the third epoch. As such, I have elected to limit the model at the 2nd epoch to minimise validation loss and overfitting. Unfortunately, my model did not run in time to submit before the deadline of the competition, and hence my score with this 2nd epoch was not recorded officially. I have included a screenshot below of my best-performing submitted score (bottom) as well as a post-deadline improvement (top) for comparison purposes.

Submission and Description	Private Score ⓘ	Public Score ⓘ
 Lab2 Comp V2 - Version 2 Complete (after deadline) · 1d ago	0.47457	0.49189
 submissionGG.csv Complete · 3d ago · BERT	0.46919	0.48170

```
subset_train_dataset = train_dataset.shuffle(seed=42).select([i for i in list(range(int(0.1 * len(train_dataset))))])

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=subset_train_dataset, # Use the subset for training
    eval_dataset=val_dataset,
)
```

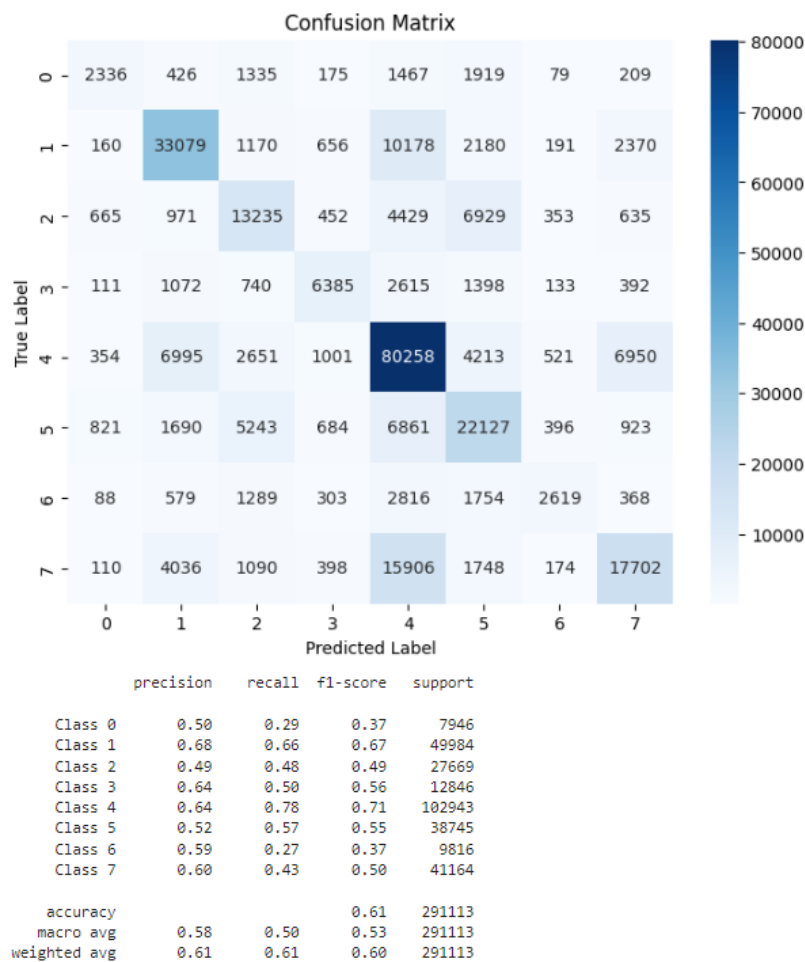
```
# Step 7: Train the Model
trainer.train()
```

```
# Step 8: Evaluate the Model
eval_results = trainer.evaluate()
print(eval_results)
```

[29112/29112 1:35:26, Epoch 2/2]

Epoch	Training Loss	Validation Loss
1	1.164100	1.142601
2	0.813100	1.128673

The BERT model is then trained on 1/10 of the actual train data (after train-test split), as I did not have sufficient processing power or time to fully train the model on the entire dataset due to its large size. Though the data entries were substantial, it is likely that the model’s final predictive accuracy could have been further improved were I to train the model on the entire training data.



The model was then evaluated on the testset to obtain a final weighted avg f1-score of 0.6.

Conclusion and Insights

Due to the informal nature of Twitter (X) and its content (tweets), which are often short, informal and laden with linguistic nuances like slang, emojis, hashtags and abbreviations, traditional models often struggle to handle these complexities effectively. For example, logistic regression and Naïve Bayes treat features independently, meaning that they are unable to capture the context of words within sentences. These models also rely on BoW or TD-IDF features that are not well-suited for handling the short and cryptic nature of tweets. Furthermore, these models are not pre-trained with a deep understanding of language and will require significant domain-specific feature engineering to achieve even a comparable performance to BERT.

With BERT, a model that has been trained on large corpus of text data alongside using attention mechanisms to consider relationships between all words in a sentence, marks a clear superiority over all other models for a complex task like emotion analysis, and hence it was my model of choice for this competition.