- Preprossing & Feature engineering
  First, I download the dataset from Kaggle and preprocess it into a pandas DataFrame for training and validation, splitting it into 80% for training and 20% for validation. I use TF-IDF to represent each tweet's text as features, setting max_features=1000.

- Model selection
  I use two different model architectures to predict the final answer: MultinomialNB and RandomForest. While MultinomialNB trains much faster than RandomForest, its performance is worse. Additionally, the MultinomialNB model exhibits overfitting, whereas the RandomForest model is able to avoid it.

Model	Public Score	Private Score
MultinomialNB	0.31090	0.29087
RandomForest	0.36108	0.34251

- BERT
  I use the BERT tokenizer to tokenize each tweet's text, with the pre-trained model set to "bert-base-uncased". For this part, I sample 50% of the entire dataset for training to save time. Regarding the model architecture, I use "BertForSequenceClassification" as the model to train the classification task. The training configuration is as follows:

  - batch_size: 16
  - optimizer: AdamW
  - learning rate: 5e-5
  - epochs: 3

  The training log is below:

```
Epoch 0: 100%|██████████| 72460/72460 [1:57:16<00:00, 10.30it/s, loss=0.74]
Validation Accuracy: 0.6427
Epoch 1: 100%|██████████| 72460/72460 [1:57:30<00:00, 10.28it/s, loss=0.00107]
Validation Accuracy: 0.6631
Epoch 2: 100%|██████████| 72460/72460 [1:56:35<00:00, 10.36it/s, loss=0.000839]
Validation Accuracy: 0.6672
```

  This method outperforms both of the models I previously trained. The public score is 0.54006, and the private score is 0.52527. I believe that with a larger dataset, deep learning may outperform machine learning. However, deep learning tends to perform poorly when the dataset size is small.