In [66]:
```
pip install --upgrade pip
```

```
Requirement already satisfied: pip in /Users/evelynhaskins/.pyenv/versions/
3.10.12/lib/python3.10/site-packages (24.3.1)
Note: you may need to restart the kernel to use updated packages.
```

In [67]:
```
pip list | grep -E 'tensorflow|keras'
```

```
keras                        3.6.0
keras-core                   0.1.7
keras-hub                    0.18.0
keras-nlp                    0.18.0
tensorflow                   2.18.0
tensorflow-io-gcs-filesystem 0.37.1
tensorflow-text              2.18.0
Note: you may need to restart the kernel to use updated packages.
```

In [68]:
```
!pip install keras-core --upgrade
!pip install -q keras-nlp --upgrade
```

```
Requirement already satisfied: keras-core in /Users/evelynhaskins/.pyenv/ver
sions/3.10.12/lib/python3.10/site-packages (0.1.7)
Requirement already satisfied: absl-py in /Users/evelynhaskins/.pyenv/versio
ns/3.10.12/lib/python3.10/site-packages (from keras-core) (2.1.0)
Requirement already satisfied: numpy in /Users/evelynhaskins/.pyenv/version
s/3.10.12/lib/python3.10/site-packages (from keras-core) (2.0.2)
Requirement already satisfied: rich in /Users/evelynhaskins/.pyenv/versions/
3.10.12/lib/python3.10/site-packages (from keras-core) (13.9.4)
Requirement already satisfied: namex in /Users/evelynhaskins/.pyenv/version
s/3.10.12/lib/python3.10/site-packages (from keras-core) (0.0.8)
Requirement already satisfied: h5py in /Users/evelynhaskins/.pyenv/versions/
3.10.12/lib/python3.10/site-packages (from keras-core) (3.12.1)
Requirement already satisfied: dm-tree in /Users/evelynhaskins/.pyenv/versio
ns/3.10.12/lib/python3.10/site-packages (from keras-core) (0.1.8)
Requirement already satisfied: markdown-it-py>=2.2.0 in /Users/evelynhaskin
s/.pyenv/versions/3.10.12/lib/python3.10/site-packages (from rich->keras-cor
e) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /Users/evelynhaski
ns/.pyenv/versions/3.10.12/lib/python3.10/site-packages (from rich->keras-co
re) (2.18.0)
Requirement already satisfied: typing-extensions<5.0,>=4.0.0 in /Users/evely
nhaskins/.pyenv/versions/3.10.12/lib/python3.10/site-packages (from rich->ke
ras-core) (4.12.2)
Requirement already satisfied: mdurl~=0.1 in /Users/evelynhaskins/.pyenv/ver
sions/3.10.12/lib/python3.10/site-packages (from markdown-it-py>=2.2.0->rich
->keras-core) (0.1.2)
```

In [69]:
```
!pip install tensorflow tensorflow-text
!pip install wordcloud
```

```
Requirement already satisfied: tensorflow in /Users/evelynhaskins/.pyenv/ver
sions/3.10.12/lib/python3.10/site-packages (2.18.0)
Requirement already satisfied: tensorflow-text in /Users/evelynhaskins/.pyen
v/versions/3.10.12/lib/python3.10/site-packages (2.18.0)
Requirement already satisfied: absl-py>=1.0.0 in /Users/evelynhaskins/.pyen
v/versions/3.10.12/lib/python3.10/site-packages (from tensorflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in /Users/evelynhaskins/.py
env/versions/3.10.12/lib/python3.10/site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /Users/evelynhaskin
s/.pyenv/versions/3.10.12/lib/python3.10/site-packages (from tensorflow) (2
4.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /User
s/evelynhaskins/.pyenv/versions/3.10.12/lib/python3.10/site-packages (from t
ensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /Users/evelynhaskins/.
pyenv/versions/3.10.12/lib/python3.10/site-packages (from tensorflow) (0.2.
0)
Requirement already satisfied: libclang>=13.0.0 in /Users/evelynhaskins/.pye
nv/versions/3.10.12/lib/python3.10/site-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /Users/evelynhaskins/.py
env/versions/3.10.12/lib/python3.10/site-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /Users/evelynhaskins/.pyenv/vers
ions/3.10.12/lib/python3.10/site-packages (from tensorflow) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!
=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in /Users/evelynhaskins/.pyenv/versions/
3.10.12/lib/python3.10/site-packages (from tensorflow) (5.28.3)
Requirement already satisfied: requests<3,>=2.21.0 in /Users/evelynhaskins/.
pyenv/versions/3.10.12/lib/python3.10/site-packages (from tensorflow) (2.32.
3)
Requirement already satisfied: setuptools in /Users/evelynhaskins/.pyenv/ver
sions/3.10.12/lib/python3.10/site-packages (from tensorflow) (65.5.0)
Requirement already satisfied: six>=1.12.0 in /Users/evelynhaskins/.pyenv/ve
rsions/3.10.12/lib/python3.10/site-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /Users/evelynhaskins/.pye
nv/versions/3.10.12/lib/python3.10/site-packages (from tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /Users/evelynhask
ins/.pyenv/versions/3.10.12/lib/python3.10/site-packages (from tensorflow)
(4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /Users/evelynhaskins/.pyenv/
versions/3.10.12/lib/python3.10/site-packages (from tensorflow) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /Users/evelynhaskins/.
pyenv/versions/3.10.12/lib/python3.10/site-packages (from tensorflow) (1.68.
0)
Requirement already satisfied: tensorboard<2.19,>=2.18 in /Users/evelynhaski
ns/.pyenv/versions/3.10.12/lib/python3.10/site-packages (from tensorflow)
(2.18.0)
Requirement already satisfied: keras>=3.5.0 in /Users/evelynhaskins/.pyenv/v
ersions/3.10.12/lib/python3.10/site-packages (from tensorflow) (3.6.0)
Requirement already satisfied: numpy<2.1.0,>=1.26.0 in /Users/evelynhaskin
s/.pyenv/versions/3.10.12/lib/python3.10/site-packages (from tensorflow) (2.
0.2)
Requirement already satisfied: h5py>=3.11.0 in /Users/evelynhaskins/.pyenv/v
ersions/3.10.12/lib/python3.10/site-packages (from tensorflow) (3.12.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in /Users/evelynhaski
ns/.pyenv/versions/3.10.12/lib/python3.10/site-packages (from tensorflow)
(0.4.1)
```

```
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /User
s/evelynhaskins/.pyenv/versions/3.10.12/lib/python3.10/site-packages (from t
ensorflow) (0.37.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /Users/evelynhaskins/.p
yenv/versions/3.10.12/lib/python3.10/site-packages (from astunparse>=1.6.0->
tensorflow) (0.45.0)
Requirement already satisfied: rich in /Users/evelynhaskins/.pyenv/versions/
3.10.12/lib/python3.10/site-packages (from keras>=3.5.0->tensorflow) (13.9.
4)
Requirement already satisfied: namex in /Users/evelynhaskins/.pyenv/version
s/3.10.12/lib/python3.10/site-packages (from keras>=3.5.0->tensorflow) (0.0.
8)
Requirement already satisfied: optree in /Users/evelynhaskins/.pyenv/version
s/3.10.12/lib/python3.10/site-packages (from keras>=3.5.0->tensorflow) (0.1
3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /Users/evelynhask
ins/.pyenv/versions/3.10.12/lib/python3.10/site-packages (from requests<3,>=
2.21.0->tensorflow) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /Users/evelynhaskins/.pyenv/v
ersions/3.10.12/lib/python3.10/site-packages (from requests<3,>=2.21.0->tens
orflow) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /Users/evelynhaskins/.p
yenv/versions/3.10.12/lib/python3.10/site-packages (from requests<3,>=2.21.0
->tensorflow) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /Users/evelynhaskins/.p
yenv/versions/3.10.12/lib/python3.10/site-packages (from requests<3,>=2.21.0
->tensorflow) (2024.8.30)
Requirement already satisfied: markdown>=2.6.8 in /Users/evelynhaskins/.pyen
v/versions/3.10.12/lib/python3.10/site-packages (from tensorboard<2.19,>=2.1
8->tensorflow) (3.7)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /Use
rs/evelynhaskins/.pyenv/versions/3.10.12/lib/python3.10/site-packages (from
tensorboard<2.19,>=2.18->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in /Users/evelynhaskins/.pyen
v/versions/3.10.12/lib/python3.10/site-packages (from tensorboard<2.19,>=2.1
8->tensorflow) (3.0.1)
Requirement already satisfied: MarkupSafe>=2.1.1 in /Users/evelynhaskins/.py
env/versions/3.10.12/lib/python3.10/site-packages (from werkzeug>=1.0.1->ten
sorboard<2.19,>=2.18->tensorflow) (2.1.4)
Requirement already satisfied: markdown-it-py>=2.2.0 in /Users/evelynhaskin
s/.pyenv/versions/3.10.12/lib/python3.10/site-packages (from rich->keras>=3.
5.0->tensorflow) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /Users/evelynhaski
ns/.pyenv/versions/3.10.12/lib/python3.10/site-packages (from rich->keras>=
3.5.0->tensorflow) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in /Users/evelynhaskins/.pyenv/ver
sions/3.10.12/lib/python3.10/site-packages (from markdown-it-py>=2.2.0->rich
->keras>=3.5.0->tensorflow) (0.1.2)
Requirement already satisfied: wordcloud in /Users/evelynhaskins/.pyenv/vers
ions/3.10.12/lib/python3.10/site-packages (1.9.4)
Requirement already satisfied: numpy>=1.6.1 in /Users/evelynhaskins/.pyenv/v
ersions/3.10.12/lib/python3.10/site-packages (from wordcloud) (2.0.2)
Requirement already satisfied: pillow in /Users/evelynhaskins/.pyenv/version
s/3.10.12/lib/python3.10/site-packages (from wordcloud) (11.0.0)
Requirement already satisfied: matplotlib in /Users/evelynhaskins/.pyenv/ver
sions/3.10.12/lib/python3.10/site-packages (from wordcloud) (3.9.2)
```

```
Requirement already satisfied: contourpy>=1.0.1 in /Users/evelynhaskins/.pye
nv/versions/3.10.12/lib/python3.10/site-packages (from matplotlib->wordclou
d) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /Users/evelynhaskins/.pyenv/v
ersions/3.10.12/lib/python3.10/site-packages (from matplotlib->wordcloud)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /Users/evelynhaskins/.py
env/versions/3.10.12/lib/python3.10/site-packages (from matplotlib->wordclou
d) (4.55.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /Users/evelynhaskins/.py
env/versions/3.10.12/lib/python3.10/site-packages (from matplotlib->wordclou
d) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /Users/evelynhaskins/.pyen
v/versions/3.10.12/lib/python3.10/site-packages (from matplotlib->wordcloud)
(24.2)
Requirement already satisfied: pyparsing>=2.3.1 in /Users/evelynhaskins/.pye
nv/versions/3.10.12/lib/python3.10/site-packages (from matplotlib->wordclou
d) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /Users/evelynhaskin
s/.pyenv/versions/3.10.12/lib/python3.10/site-packages (from matplotlib->wor
dcloud) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /Users/evelynhaskins/.pyenv/versi
ons/3.10.12/lib/python3.10/site-packages (from python-dateutil>=2.7->matplot
lib->wordcloud) (1.16.0)
```

In [70]:
```python
import os
os.environ['KERAS_BACKEND'] = 'tensorflow'
```

In [71]:
```python
import os
import numpy as np
import pandas as pd
import tensorflow as tf
import keras_core as keras
import keras_nlp
from sklearn.model_selection import train_test_split
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
```

In [72]:
```python
os.environ['KERAS_BACKEND'] = 'tensorflow'

print("TensorFlow version:", tf.__version__)
print("KerasNLP version:", keras_nlp.__version__)
```

```
TensorFlow version: 2.18.0
KerasNLP version: 0.17.0
```

Loading the dataset

In [73]:
```python
base_path = "/Users/evelynhaskins/Downloads/nlp-getting-started"
test_data = os.path.join(base_path, "test.csv")
test = pd.read_csv(test_data)
```

```
train_data = os.path.join(base_path, "train.csv")
train = pd.read_csv(train_data)
```

Evaluating the dataset properties

In [74]:
```
print('Training Set Shape = {}'.format(train.shape))
print('Training Set Memory Usage = {:.2f} MB'.format(train.memory_usage().su
print('Test Set Shape = {}'.format(test.shape))
print('Test Set Memory Usage = {:.2f} MB'.format(test.memory_usage().sum() /

print(train.head())
print(test.head())

train["length"] = train["text"].apply(lambda x: len(x))
test["length"] = test["text"].apply(lambda x: len(x))

print("Train Length Stats:")
print(train["length"].describe())
print()

print("Test Length Stats:")
print(test["length"].describe())
```

```
Training Set Shape = (7613, 5)
Training Set Memory Usage = 0.29 MB
Test Set Shape = (3263, 4)
Test Set Memory Usage = 0.10 MB
   id keyword location                                               text  \
0   1     NaN      NaN  Our Deeds are the Reason of this #earthquake M...
1   4     NaN      NaN             Forest fire near La Ronge Sask. Canada
2   5     NaN      NaN  All residents asked to 'shelter in place' are ...
3   6     NaN      NaN  13,000 people receive #wildfires evacuation or...
4   7     NaN      NaN  Just got sent this photo from Ruby #Alaska as ...

   target
0       1
1       1
2       1
3       1
4       1
   id keyword location                                               text
0   0     NaN      NaN                 Just happened a terrible car crash
1   2     NaN      NaN  Heard about #earthquake is different cities, s...
2   3     NaN      NaN  there is a forest fire at spot pond, geese are...
3   9     NaN      NaN           Apocalypse lighting. #Spokane #wildfires
4  11     NaN      NaN      Typhoon Soudelor kills 28 in China and Taiwan
Train Length Stats:
count    7613.000000
mean      101.037436
std        33.781325
min         7.000000
25%        78.000000
50%       107.000000
75%       133.000000
max       157.000000
Name: length, dtype: float64

Test Length Stats:
count    3263.000000
mean      102.108183
std        33.972158
min         5.000000
25%        78.000000
50%       109.000000
75%       134.000000
max       151.000000
Name: length, dtype: float64
```

Splitting dataset

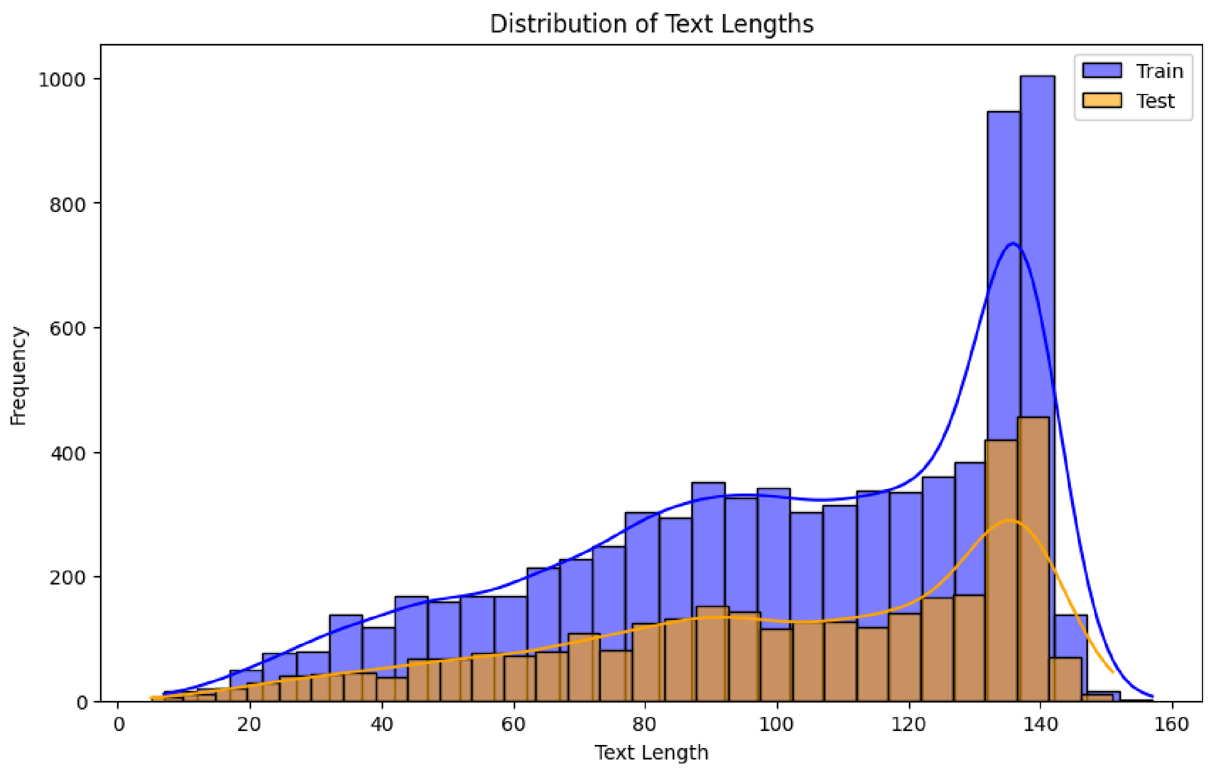
In [75]:
```
BATCH_SIZE = 32
NUM_TRAINING_EXAMPLES = train.shape[0]
TRAIN_SPLIT = 0.8
VAL_SPLIT = 0.2
STEPS_PER_EPOCH = int(NUM_TRAINING_EXAMPLES) * TRAIN_SPLIT // BATCH_SIZE
EPOCHS = 2
AUTO = tf.data.experimental.AUTOTUNE

X = train["text"]
```

```
y = train["target"]
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=VAL_SPLIT,

X_test = test["text"]

print("Training Data Shape:", X_train.shape)
print("Validation Data Shape:", X_val.shape)
```

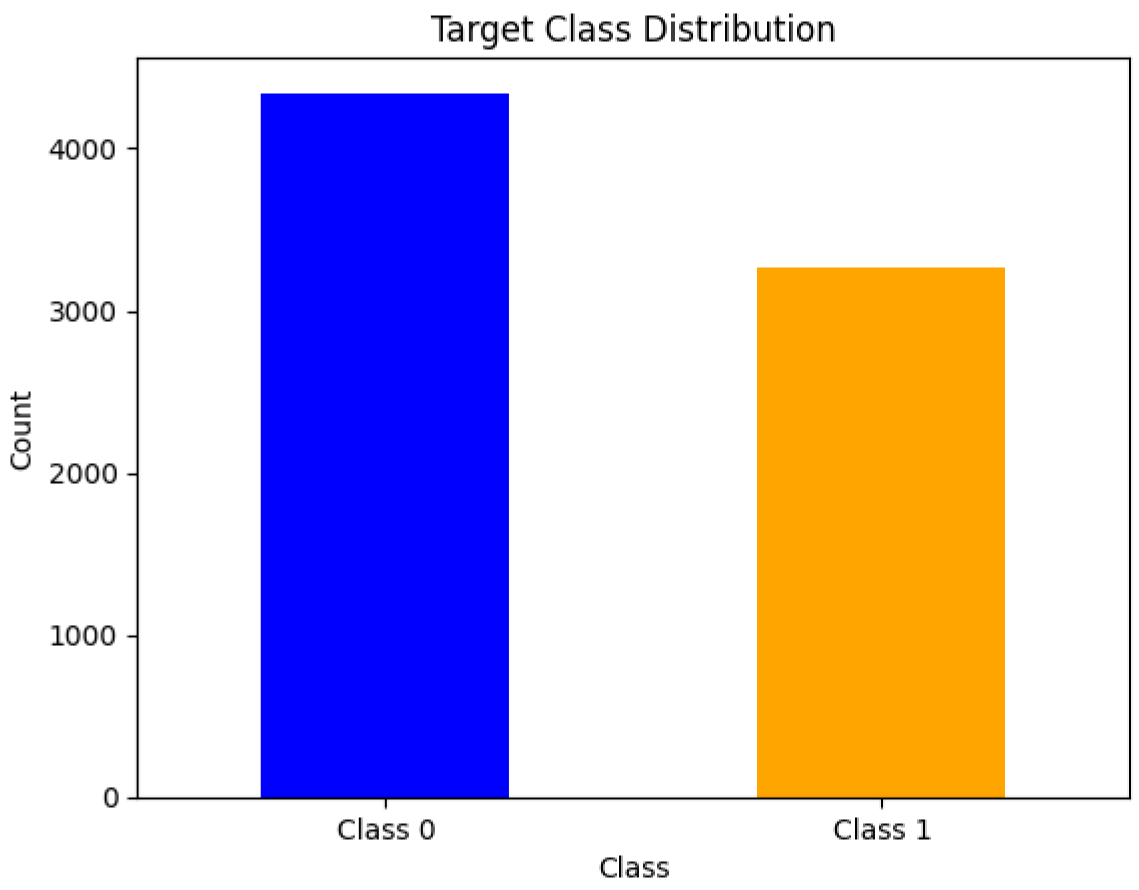
```
Training Data Shape: (6090,)
Validation Data Shape: (1523,)
```

Analysis of Dataset

In [76]:
```
plt.figure(figsize=(10, 6))
sns.histplot(train["length"], bins=30, kde=True, color="blue", label="Train"
sns.histplot(test["length"], bins=30, kde=True, color="orange", label="Test"
plt.title("Distribution of Text Lengths")
plt.xlabel("Text Length")
plt.ylabel("Frequency")
plt.legend()
plt.show()
```

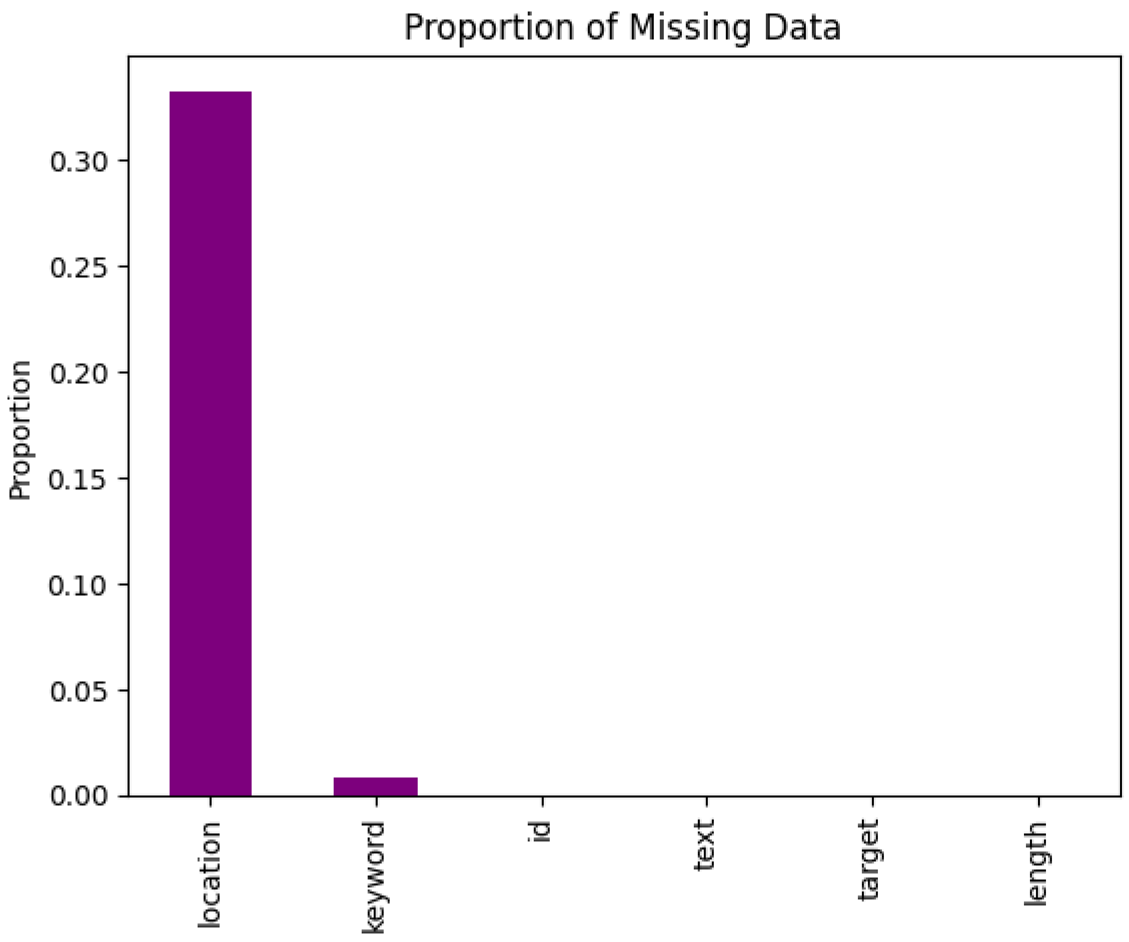


In [77]:
```
train["target"].value_counts().plot(kind="bar", color=["blue", "orange"])
plt.title("Target Class Distribution")
plt.xlabel("Class")
plt.ylabel("Count")
plt.xticks(ticks=[0, 1], labels=["Class 0", "Class 1"], rotation=0)
plt.show()
```



Checking missing values

In [78]:
```
missing_data = train.isnull().mean().sort_values(ascending=False)
missing_data.plot(kind="bar", color="purple")
plt.title("Proportion of Missing Data")
plt.ylabel("Proportion")
plt.show()
```



No missing values in areas we are evaluating such as text and target

Checking fequently used words

```
In [79]: text = " ".join(train["text"].fillna("").tolist())
wordcloud = WordCloud(width=800, height=400, background_color="white").gener
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title("Most Common Words in Text")
plt.show()
```

Most Common Words in Text

Interesting way to see common words trending in the tweets, not overly relevant in this evaluation but fun to look at

In [80]:
```
preset = "distil_bert_base_en_uncased"
preprocessor = keras_nlp.models.DistilBertPreprocessor.from_preset(
    preset, sequence_length=160, name="preprocessor_4_tweets"
)
classifier = keras_nlp.models.DistilBertClassifier.from_preset(
    preset, preprocessor=preprocessor, num_classes=2
)
classifier.summary()
```

Preprocessor: "preprocessor_4_tweets"

Layer (type)	
distil_bert_tokenizer (DistilBertTokenizer)	

Model: "distil_bert_text_classifier_4"

Layer (type)	Output Shape	
padding_mask (InputLayer)	(None, None)	
token_ids (InputLayer)	(None, None)	
distil_bert_backbone (DistilBertBackbone)	(None, None, 768)	66,
get_item_4 (GetItem)	(None, 768)	
pooled_dense (Dense)	(None, 768)	
output_dropout (Dropout)	(None, 768)	
logits (Dense)	(None, 2)	

```
Total params: 66,955,010 (255.41 MB)
Trainable params: 66,955,010 (255.41 MB)
Non-trainable params: 0 (0.00 B)
```

Creating a Nueral Network to predict target for tweets to determine whether a person's words are actually announcing a disaster.

Convert the data into TensorFlow Datasets for efficient batching

In [81]:
```
def prepare_dataset(X, y, batch_size, is_training=True):
    dataset = tf.data.Dataset.from_tensor_slices((X, y))
    if is_training:
        dataset = dataset.shuffle(len(X))
    dataset = dataset.batch(batch_size).prefetch(AUTO)
    return dataset
```

These lines of code create TensorFlow datasets for efficient training and validation by converting the raw data into a format that is optimized for use with TensorFlow models.

In [82]:
```
train_dataset = prepare_dataset(X_train, y_train, BATCH_SIZE, is_training=Tr
val_dataset = prepare_dataset(X_val, y_val, BATCH_SIZE, is_training=False)
```

In [83]:
```
# Ensure the datasets repeat for continuous training across epochs
train_dataset = train_dataset.repeat()
val_dataset = val_dataset.repeat()
```

Compiling the data: Use the Adam optimizer to adjust the weights during backpropagation. Measure the model's performance using Sparse Categorical Crossentropy as the loss function. Track and display the model's accuracy during training and validation.

In [92]:
```
classifier.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=5e-5),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy']
)
```

Reduced size just for this project so I didn't have to wait for it to run all day, wouldn't reccomend this

In [97]:
```
train_dataset = train_dataset.prefetch(buffer_size=tf.data.experimental.AUTO
val_dataset = val_dataset.prefetch(buffer_size=tf.data.experimental.AUTOTUNE
```

Training the nueral network

In [98]:
```
# Train the model
history = classifier.fit(
    train_dataset.take(100),  # Use a small subset
    validation_data=val_dataset.take(10),
    epochs=10,
    steps_per_epoch=10,
    validation_steps=2
)
```

```
Epoch 1/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 36s 4s/step - accuracy: 0.8379 - loss: 0.4081 - v
al_accuracy: 0.7500 - val_loss: 0.4842
Epoch 2/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 36s 4s/step - accuracy: 0.8219 - loss: 0.4003 - v
al_accuracy: 0.8125 - val_loss: 0.4688
Epoch 3/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 36s 4s/step - accuracy: 0.8684 - loss: 0.3452 - v
al_accuracy: 0.7812 - val_loss: 0.4704
Epoch 4/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 36s 4s/step - accuracy: 0.7973 - loss: 0.4822 - v
al_accuracy: 0.8125 - val_loss: 0.4202
Epoch 5/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 35s 4s/step - accuracy: 0.8269 - loss: 0.4325 - v
al_accuracy: 0.8281 - val_loss: 0.4134
Epoch 6/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 35s 4s/step - accuracy: 0.8271 - loss: 0.4171 - v
al_accuracy: 0.8438 - val_loss: 0.3692
Epoch 7/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 35s 4s/step - accuracy: 0.8904 - loss: 0.3044 - v
al_accuracy: 0.8125 - val_loss: 0.3836
Epoch 8/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 36s 4s/step - accuracy: 0.8683 - loss: 0.3672 - v
al_accuracy: 0.8438 - val_loss: 0.3901
Epoch 9/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 36s 4s/step - accuracy: 0.8425 - loss: 0.3701 - v
al_accuracy: 0.8125 - val_loss: 0.4279
Epoch 10/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 36s 4s/step - accuracy: 0.8235 - loss: 0.4184 - v
al_accuracy: 0.8281 - val_loss: 0.4517
```

Why I Chose DistilBERT for This Architect

For this text classification problem, I chose **DistilBERT**, a transformer-based architecture, because it provides an efficient and effective solution for processing natural language. DistilBERT is a smaller, faster, and lighter version of the well-known **BERT (Bidirectional Encoder Representations from Transformers)**, which is built on the **transformer architecture** introduced in the seminal paper *"Attention is All You Need"*.

Key reasons for choosing DistilBERT:

1. **Transformer-Based Architecture**:
   DistilBERT retains the core components of the transformer architecture, including the **self-attention mechanism**, which allows it to capture contextual relationships between words in a sequence effectively. This is critical for tasks where understanding the meaning of text depends on context.

2. **Efficiency Without Sacrificing Performance**:
   DistilBERT is designed to be faster and more resource-efficient than the original BERT. It achieves this by:

   - Reducing the number of layers (6 in DistilBERT vs. 12 in BERT base).
   - Using knowledge distillation to transfer knowledge from a larger BERT model while maintaining comparable accuracy.

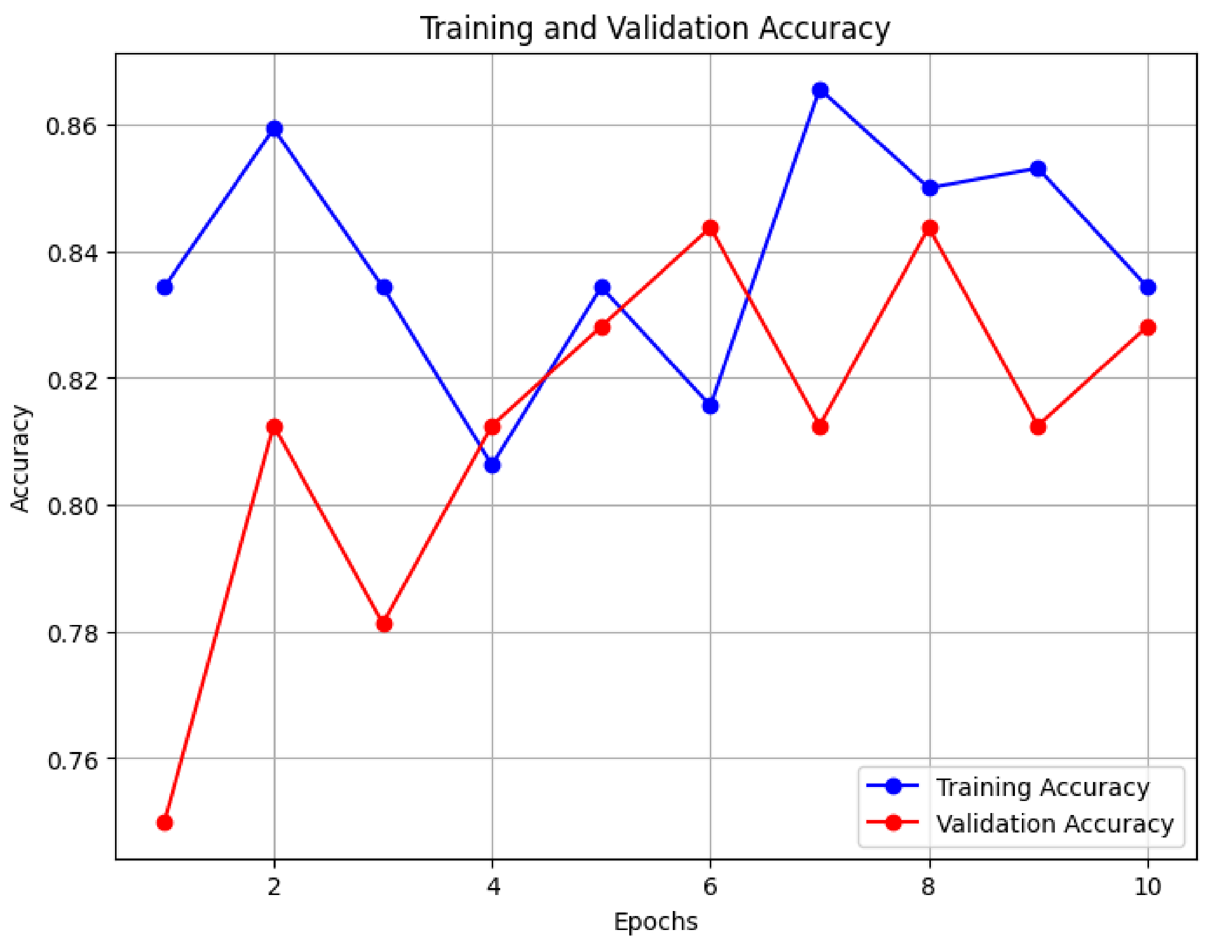
   This makes it well-suited for my dataset, as it balances computational cost and performance, enabling faster training and inference.

3. **Pre-Trained Model**:
   By using a pre-trained version of DistilBERT, I leverage the knowledge it has already learned from large corpora, such as Wikipedia and BookCorpus. This helps achieve better results with limited training data compared to training a model from scratch.

4. **Flexibility for Text Classification**:
   DistilBERT's pre-trained model can easily be extended with a classification head, which is specifically designed for binary classification tasks like mine. Using the `keras_nlp.models.DistilBertClassifier`, I benefit from an end-to-end pipeline tailored for this problem.

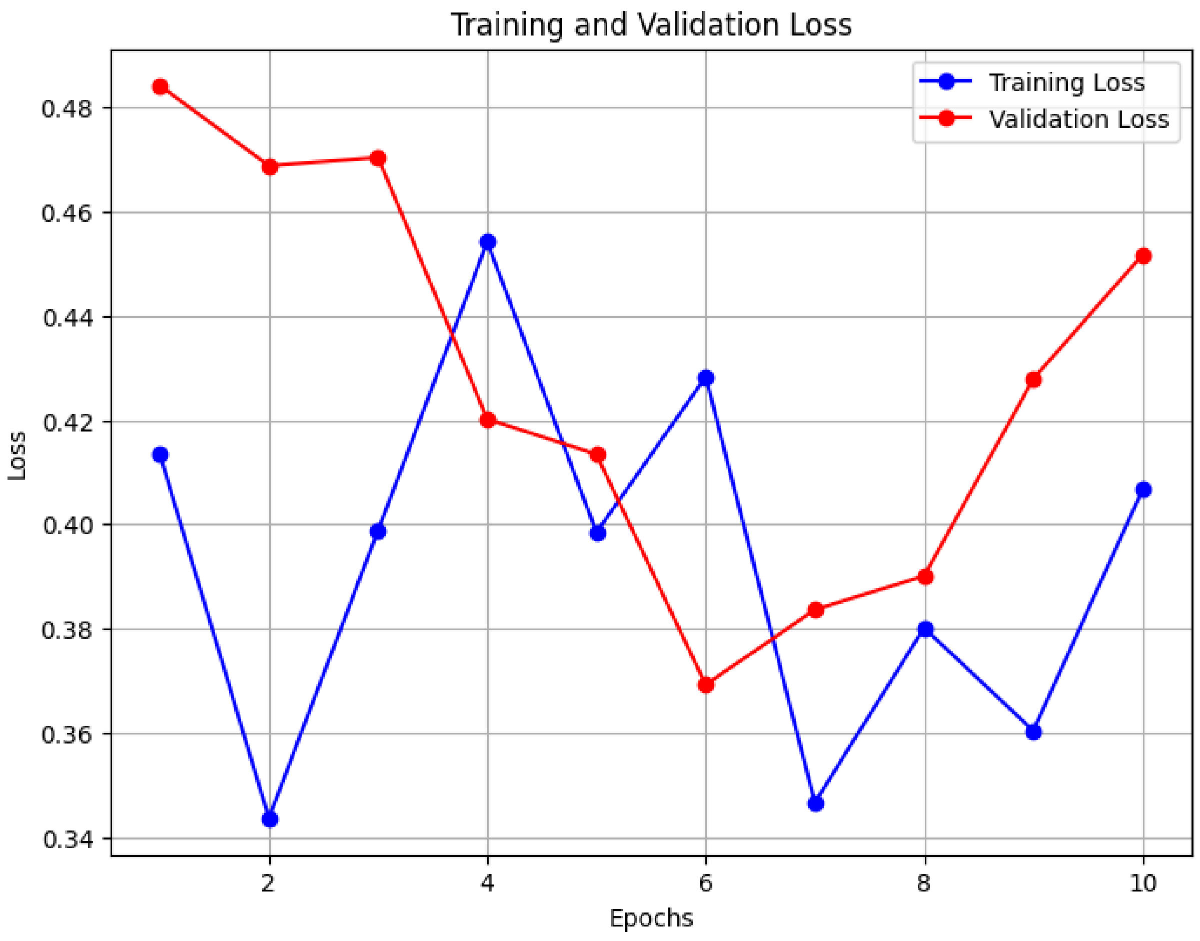
Evaluating Accuracy

In [99]:
```
accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']

epochs = range(1, len(accuracy) + 1)

# Plot the accuracy
plt.figure(figsize=(8, 6))
plt.plot(epochs, accuracy, 'bo-', label='Training Accuracy')
plt.plot(epochs, val_accuracy, 'ro-', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)
plt.show()
```



In [100…

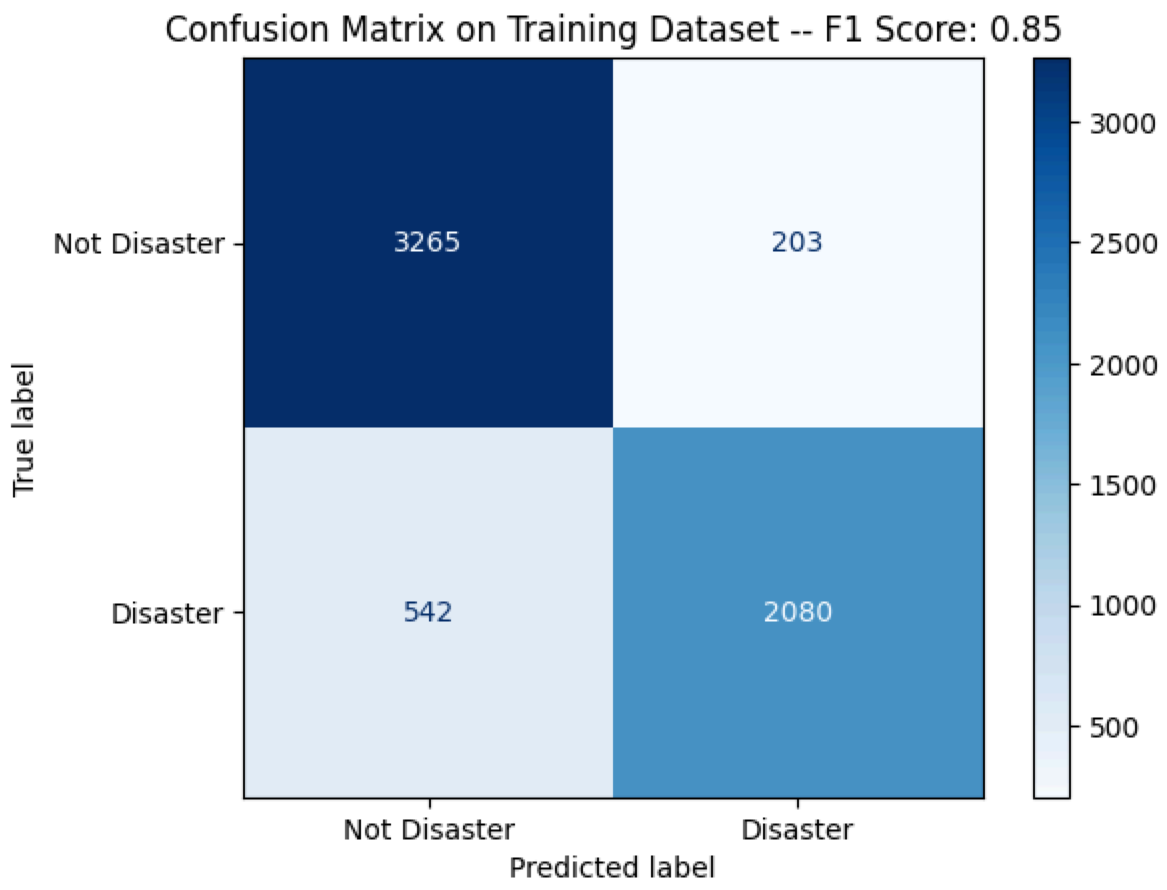
```
loss = history.history['loss']
val_loss = history.history['val_loss']

# Plot the loss
plt.figure(figsize=(8, 6))
plt.plot(epochs, loss, 'bo-', label='Training Loss')
plt.plot(epochs, val_loss, 'ro-', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)
plt.show()
```



```
In [101…  def displayConfusionMatrix(y_true, y_pred, dataset):
              disp = ConfusionMatrixDisplay.from_predictions(
                  y_true,
                  np.argmax(y_pred, axis=1),
                  display_labels=["Not Disaster","Disaster"],
                  cmap=plt.cm.Blues
              )

              tn, fp, fn, tp = confusion_matrix(y_true, np.argmax(y_pred, axis=1)).rav
              f1_score = tp / (tp+((fn+fp)/2))

              disp.ax_.set_title("Confusion Matrix on " + dataset + " Dataset -- F1 Sc
```

```
In [102…  y_pred_train = classifier.predict(X_train)

          displayConfusionMatrix(y_train, y_pred_train, "Training")
```

```
191/191 ━━━━━━━━━━━━━━━━━━━━ 220s 1s/step
```



The **confusion matrix** above is used to evaluate the performance of the classification model, and it helps you understand how well the model is distinguishing between different Disaster and Not Disaster.

Structure of the Confusion Matrix

For a binary classification problem, a confusion matrix looks like this:

	Predicted Positive (1)	**Predicted Negative (0)**
Actual Positive (1)	True Positive (TP)	False Negative (FN)
Actual Negative (0)	False Positive (FP)	True Negative (TN)

Where:

- **True Positive (TP)**: Correctly predicted Disaster Tweets.
- **False Positive (FP)**: Incorrectly predicted Disaster Tweets (weren't actually disaster tweets).
- **False Negative (FN)**: Incorrectly predicted as Not Disaster (were Disaster Tweets).
- **True Negative (TN)**: Correctly predicted Not Disaster Tweets.

```
In [ ]:
```