From Text to Insight: Disaster Detection with NLP

1. Problem Description

I participated in the Kaggle competition "Natural Language Processing with Disaster Tweets," which required me to build a machine learning model to classify tweets as either related to a real disaster or not:

Objective:

`0`: Non-disaster tweet.

`1`: Disaster tweet.

Challenges:

Tweets are short and may contain unstructured data, including typos, symbols, and emojis.

This task required natural language processing (NLP) techniques for feature extraction and classification.

---

2. Data Processing and Feature Extraction

Data Description

Training dataset (`train.csv`):

Contained 7613 records with tweet text (`text`), keywords (`keyword`), location (`location`), and labels (`target`).

Test dataset (`test.csv`):

Contained tweet text (`text`), keywords (`keyword`), and location (`location`), but without labels.

Data Cleaning

I filled missing values in the `keyword` and `location` columns with `'missing'` and `'unknown'`, respectively.

I cleaned the text by removing URLs, punctuation, and extra spaces, and converted all text to lowercase.

Feature Extraction

I used the TF-IDF method to convert text into numerical features, selecting the top 5000 most frequent words for embedding.

---

3. Model Selection and Training

Model Selection

I chose Logistic Regression as the baseline model because it is simple, efficient, and interpretable.

Performance Results:

Accuracy: 80.4%.

The confusion matrix indicated that the model had lower recall when classifying disaster-related tweets (class `1`).

Results Analysis

Classification Report:

```
vbnet

Copy code

Precision: 0.82 (Disaster Class)

Recall: 0.69 (Disaster Class)

F1-Score: 0.75 (Disaster Class)
```

Confusion Matrix:

The model misclassified some disaster-related tweets as non-disaster, leading to lower recall for class `1`.

---

4. Submission and Testing Results

Test Data Prediction

I applied the same cleaning and feature extraction process to the test dataset.

Using the trained logistic regression model, I predicted the labels for the test data.

Submission File

I generated the `submission.csv` file in the required format:

```
bash

Copy code

id,target

0,1

1,0

2,1
```

After submitting to Kaggle, I reviewed my ranking on the leaderboard.

---

5. Learnings and Future Improvements

Key Takeaways

I gained experience in building a complete NLP pipeline, including data cleaning, feature extraction, model training, and evaluation.

This project helped me understand how to handle real-world text classification problems.

Future Improvements

I plan to experiment with deep learning models, such as LSTM, to capture sequential patterns in text.

I aim to explore more advanced embedding techniques like Word2Vec or GloVe to improve feature representation.

Fine-tuning hyperparameters could help improve recall and overall performance.

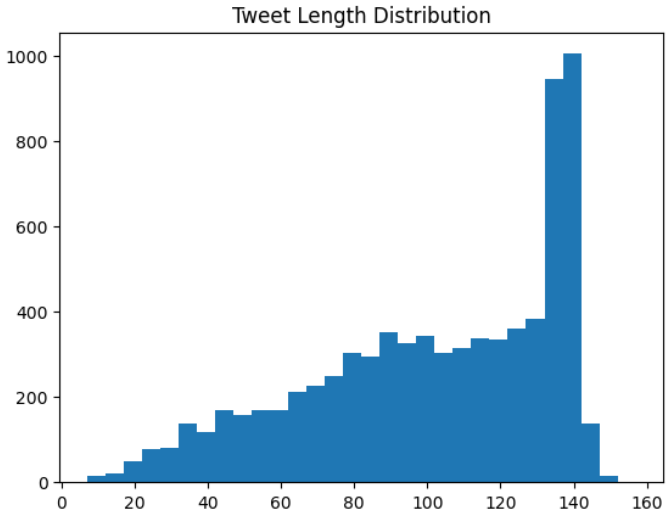
[19]:
```
import pandas as pd
data = pd.read_csv("train.csv")
print(data.shape)
data.head()
```

```
(7613, 5)
```

[19]:

	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1

[20]:
```
import matplotlib.pyplot as plt
data['text_length'] = data['text'].apply(len)
plt.hist(data['text_length'], bins=30)
plt.title("Tweet Length Distribution")
plt.show()
```



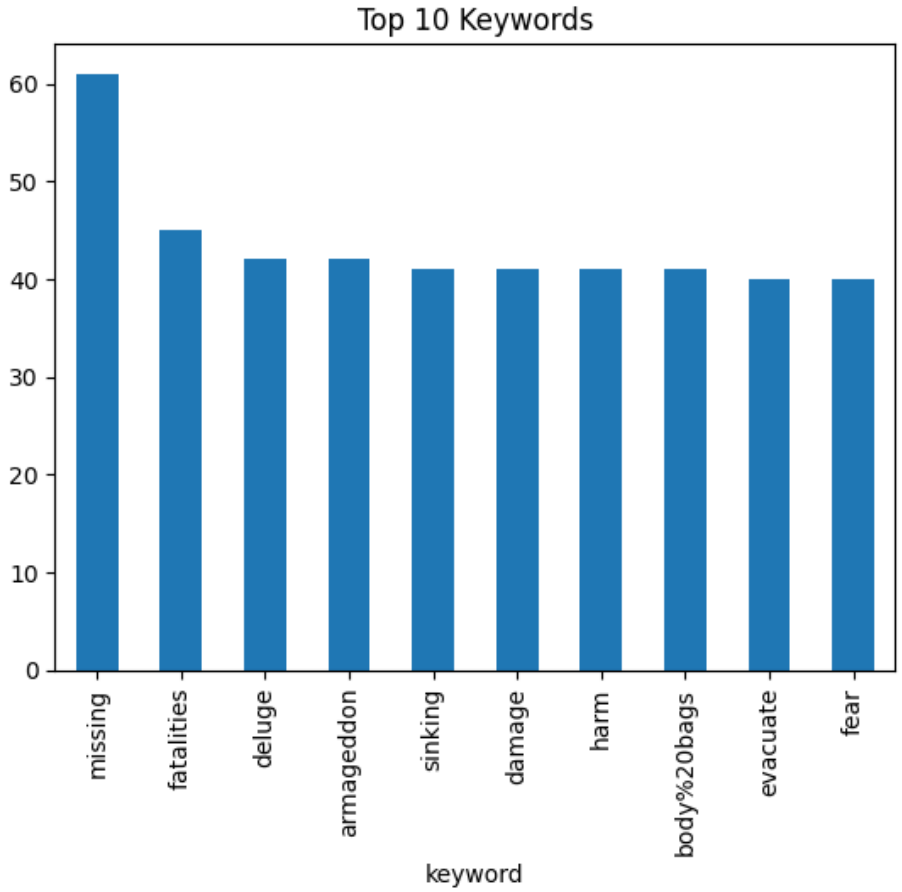
[21]:
```
print(data.isnull().sum())
```

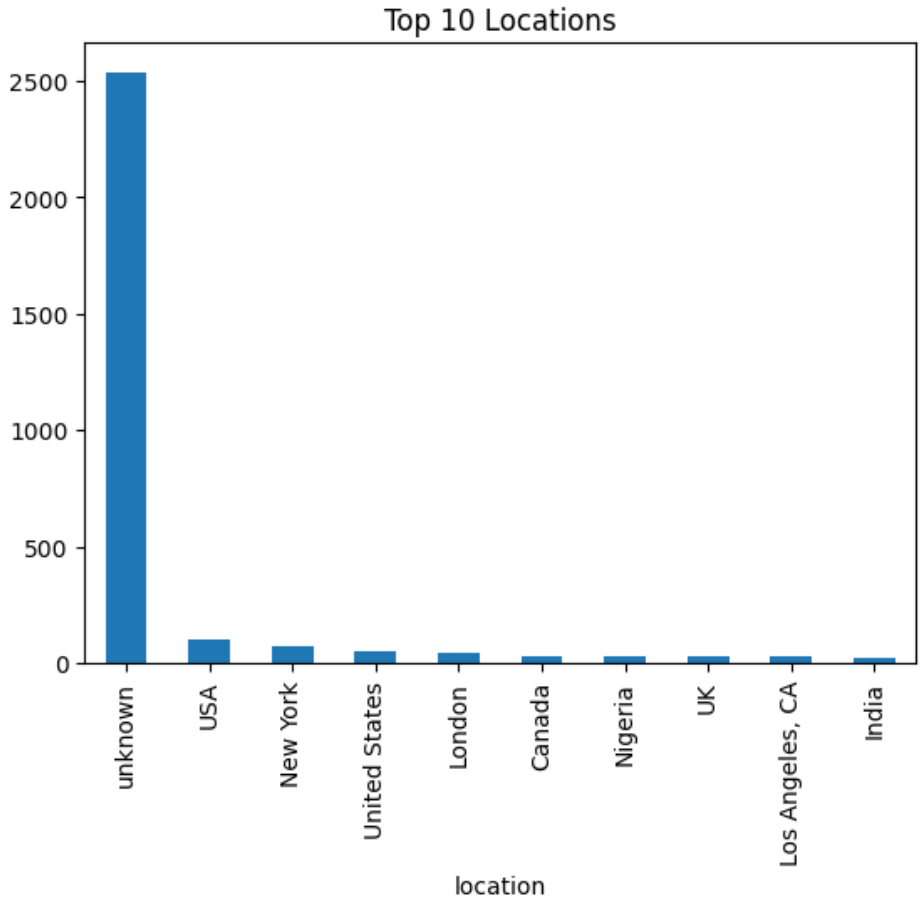
```
id                0
keyword          61
location       2533
text              0
target            0
text_length       0
dtype: int64
```

[22]:
```
data['keyword'] = data['keyword'].fillna('missing')
data['location'] = data['location'].fillna('unknown')


print(data.isnull().sum())


keyword_counts = data['keyword'].value_counts()
keyword_counts[:10].plot(kind='bar', title='Top 10 Keywords')
plt.show()


location_counts = data['location'].value_counts()
location_counts[:10].plot(kind='bar', title='Top 10 Locations')
plt.show()


import re
def clean_text(text):
    text = text.lower()
    text = re.sub(r'http\S+', '', text)
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    text = re.sub(r'\s+', ' ', text).strip()
    return text

data['clean_text'] = data['text'].apply(clean_text)


from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(max_features=5000)
X = tfidf.fit_transform(data['clean_text']).toarray()
y = data['target']


from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
print("Training data shape:", X_train.shape)
print("Test data shape:", X_test.shape)
```

```
id             0
keyword        0
location       0
text           0
target         0
text_length    0
dtype: int64
```



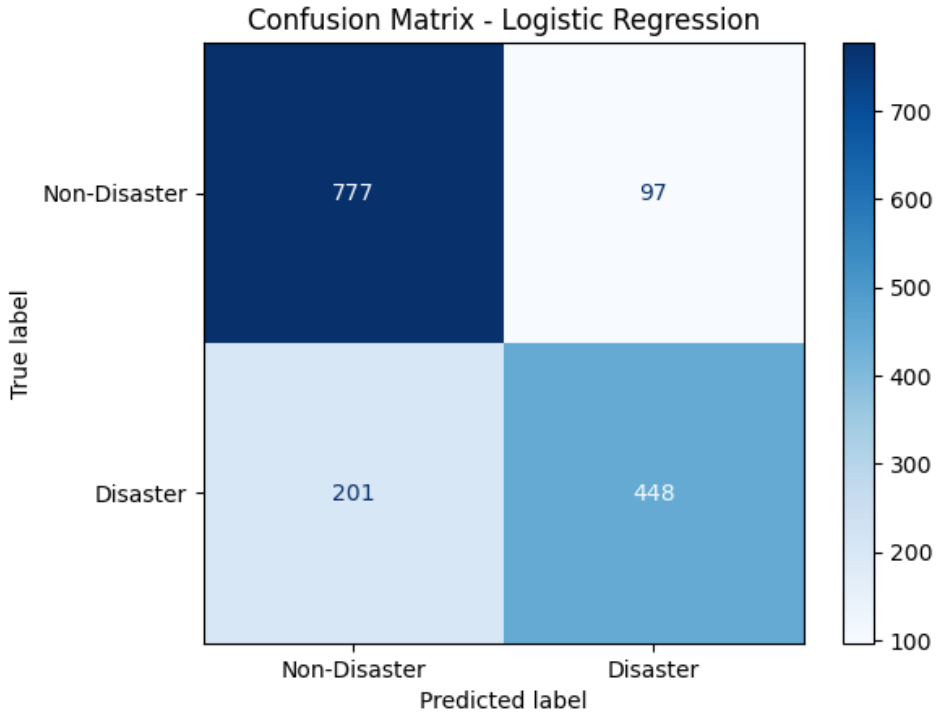



```
Training data shape: (6090, 5000)
Test data shape: (1523, 5000)
```

[29]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt


logistic_model = LogisticRegression(max_iter=1000, random_state=42)
logistic_model.fit(X_train, y_train)


y_pred = logistic_model.predict(X_test)


print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))


ConfusionMatrixDisplay.from_predictions(y_test, y_pred, display_labels=['Non-Disaster', 'Disaster'], cmap='Blues')
plt.title("Confusion Matrix - Logistic Regression")
plt.show()
```

```
Accuracy: 0.804333552199606
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.89      0.84       874
           1       0.82      0.69      0.75       649

    accuracy                           0.80      1523
   macro avg       0.81      0.79      0.79      1523
weighted avg       0.81      0.80      0.80      1523
```



[31]:

```
test_data = pd.read_csv("test.csv")


test_data['keyword'] = test_data['keyword'].fillna('missing')
test_data['location'] = test_data['location'].fillna('unknown')
test_data['clean_text'] = test_data['text'].apply(clean_text)


X_test_final = tfidf.transform(test_data['clean_text']).toarray()


test_predictions = logistic_model.predict(X_test_final)

submission = pd.DataFrame({'id': test_data['id'], 'target': test_predictions})
submission.to_csv("submission.csv", index=False)

print("Submission file 'submission.csv' created successfully!")
```

```
Submission file 'submission.csv' created successfully!
```