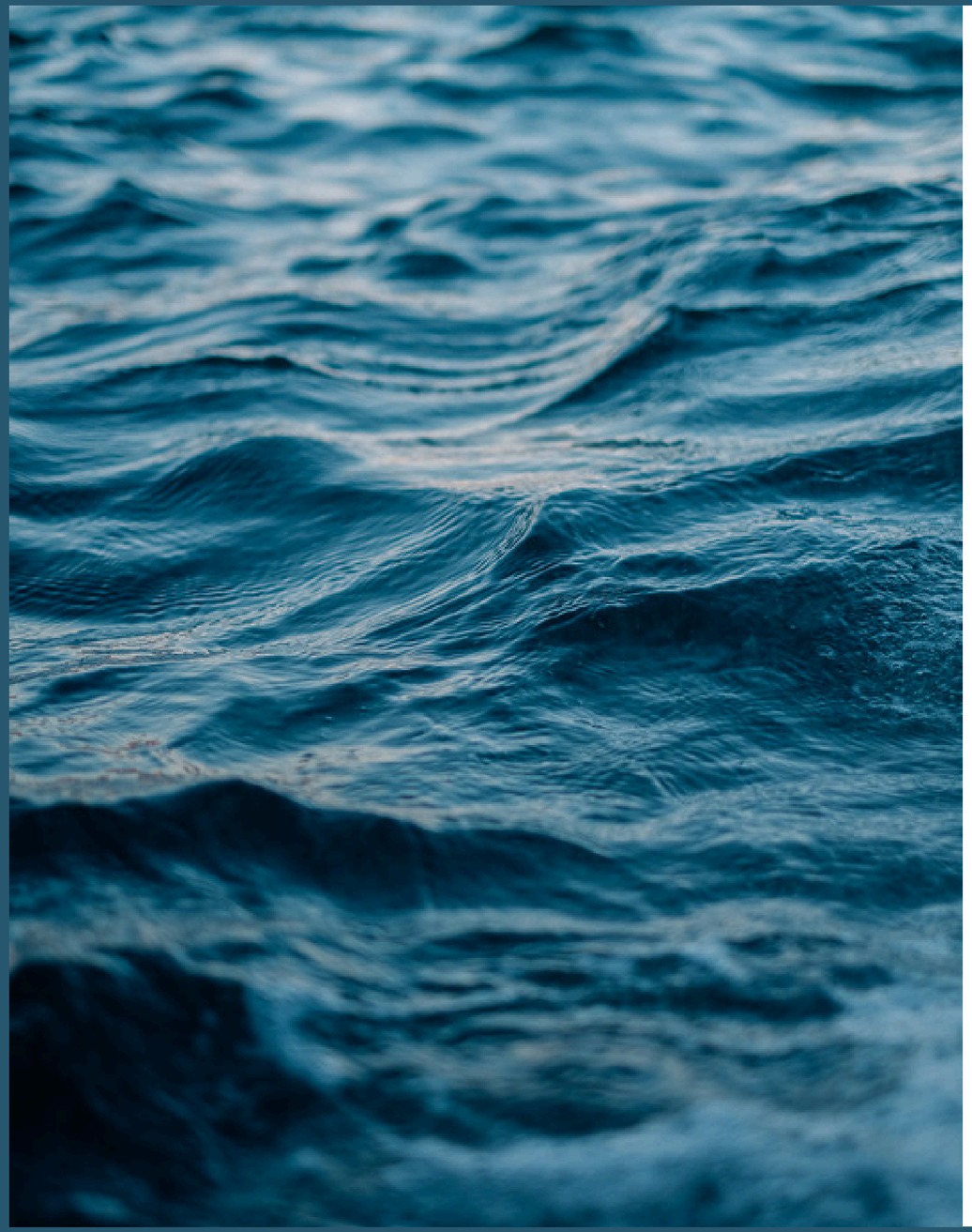
人工三甲 第四組
B0928009 吳冠諭
B0928023 盧于璇
B0928026 洪詩晴

DATA MINING PROJECT

DISASTER TWEETS OR NOT ?

CONTENT



01

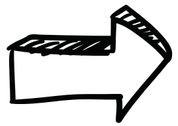
INTRODUCTION

WHY WE CHOICE THIS PROBLEM

社交平台已成為人們獲取新聞或訊息的主要渠道之一

災難事件和**緊急事件**的重要訊息來源之一

災難發生時  社交平台上發布相關訊息及推文

 包含了大量的無關訊息、虛假訊息

HOW DO WE DO THAT

通過**自然語言處理技術**對災難事件推文進行分類、判斷虛假

提高災難事件的響應速度、救援效率

題目連結：https://www.kaggle.com/competitions/nlp-getting-started

INTRODUCTION

1. What kind of the problem the project is ?

   使用推文文本來偵測及預測它們是否與真實災難有關

2. the problems/goal it addressed ?

   協助政府、新聞機構等組織更快且準確地識別災難事件並及時做出反應

INTRODUCTION

3 Why is it difficult and a challenge?

文本通常包含大量雜訊和歧義

使用人工方式處理大量推文數據十分困難

4 What are the performances of previous methods on the problem (or similar problems) if any.

以前的方法通常基於統計分析和規則設置

無法處理複雜的自然語言和多樣的表達方式(口語化)

DATA(CSV)

TRAINING DATA:

train

id	keyword	location	text	target

ID　推文關鍵字　推文地點　　推文內容　　標籤

train

id	keyword	location	text	target
1			Our Deeds are the Reason of this #earthquake May ALLAH Forgive us all	1
4			Forest fire near La Ronge Sask. Canada	1
5			All residents asked to 'shelter in place' are being notified by officers. No other evacuation or shelter in place orders are expected	1
6			13,000 people receive #wildfires evacuation orders in California	1
7			Just got sent this photo from Ruby #Alaska as smoke from #wildfires pours into a school	1
8			#RockyFire Update => California Hwy. 20 closed in both directions due to Lake County fire - #CAfire #wildfires	1
10			#flood #disaster Heavy rain causes flash flooding of streets in Manitou, Colorado Springs areas	1

DATA(CSV)

TESTING DATA:

id	keyword	location	text
ID	推文關鍵字	推文地點	推文內容

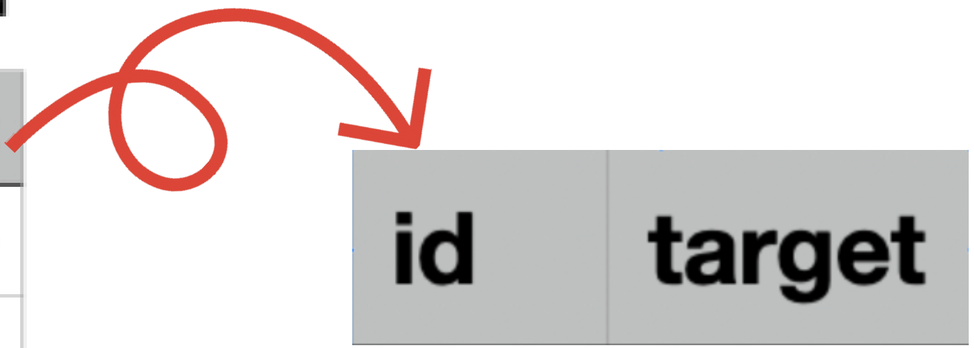
test

id	keyword	location	text
0			Just happened a terrible car crash
2			Heard about #earthquake is different cities, stay safe everyone.
3			there is a forest fire at spot pond, geese are fleeing across the street, I cannot save them all
9			Apocalypse lighting. #Spokane #wildfires
11			Typhoon Soudelor kills 28 in China and Taiwan
12			We're shaking...It's an earthquake
21			They'd probably still show more life than Arsenal did yesterday, eh? EH?

DATA

sample_su

id	target
0	0
2	0
3	0
9	0
11	0
12	0
21	0



id	target

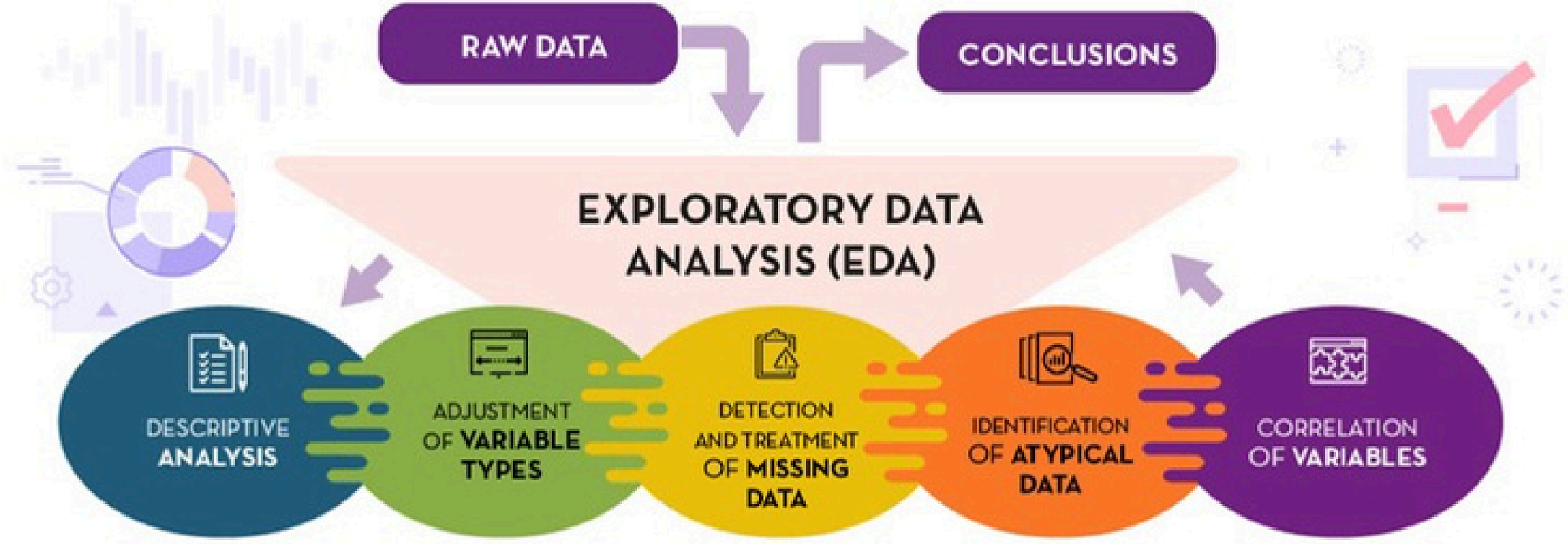
ID 標籤

1 :涉及真實災難事件

0: 不涉及真實災難事件

02
ANALYSIS DATA

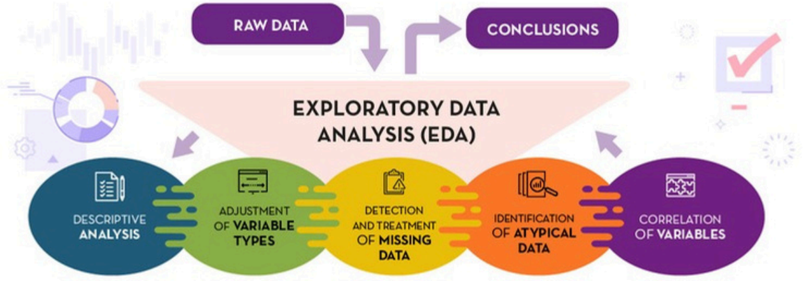
EDA : EXPLORATORY DATA ANALYSIS



探索式資料分析

EDA：EXPLORATORY DATA ANALYSIS

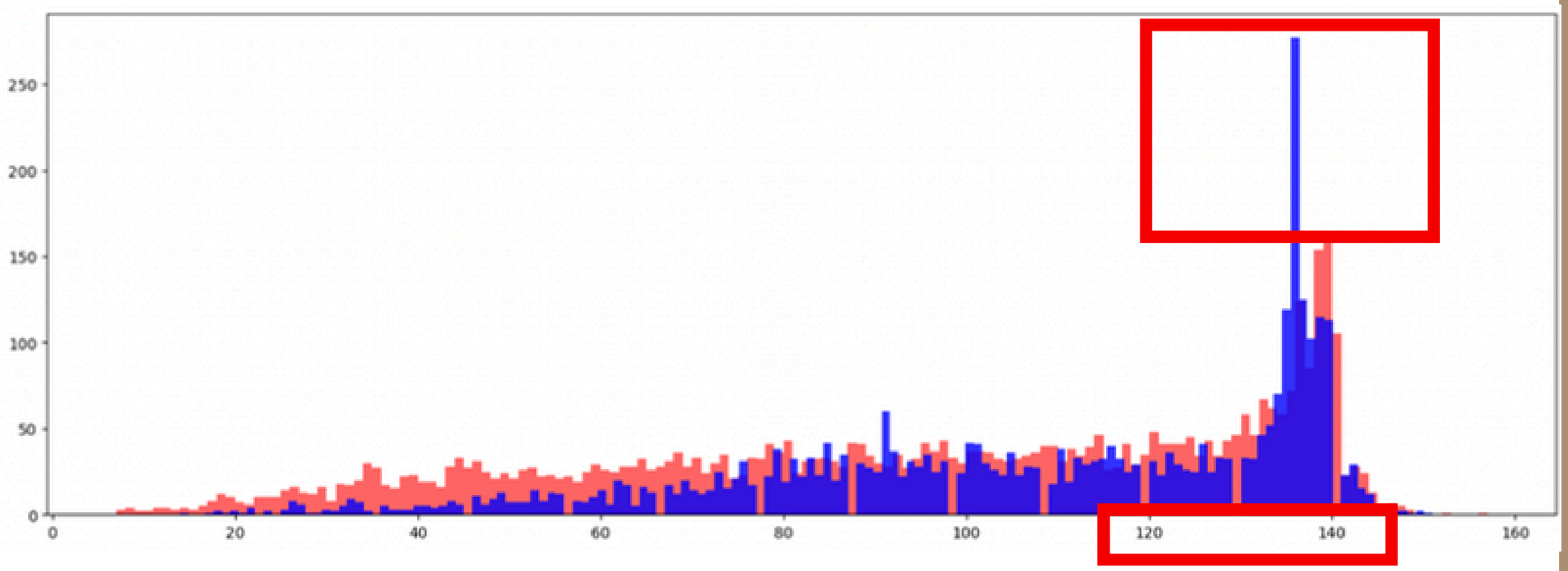
1. 獲取資料的資訊、結構和特點
2. 數據可視化
3. 檢查有無離群值或異常值
4. 分析各變數間的關聯性並找出重要的變數



1. 推文長度與推文真實性的關係

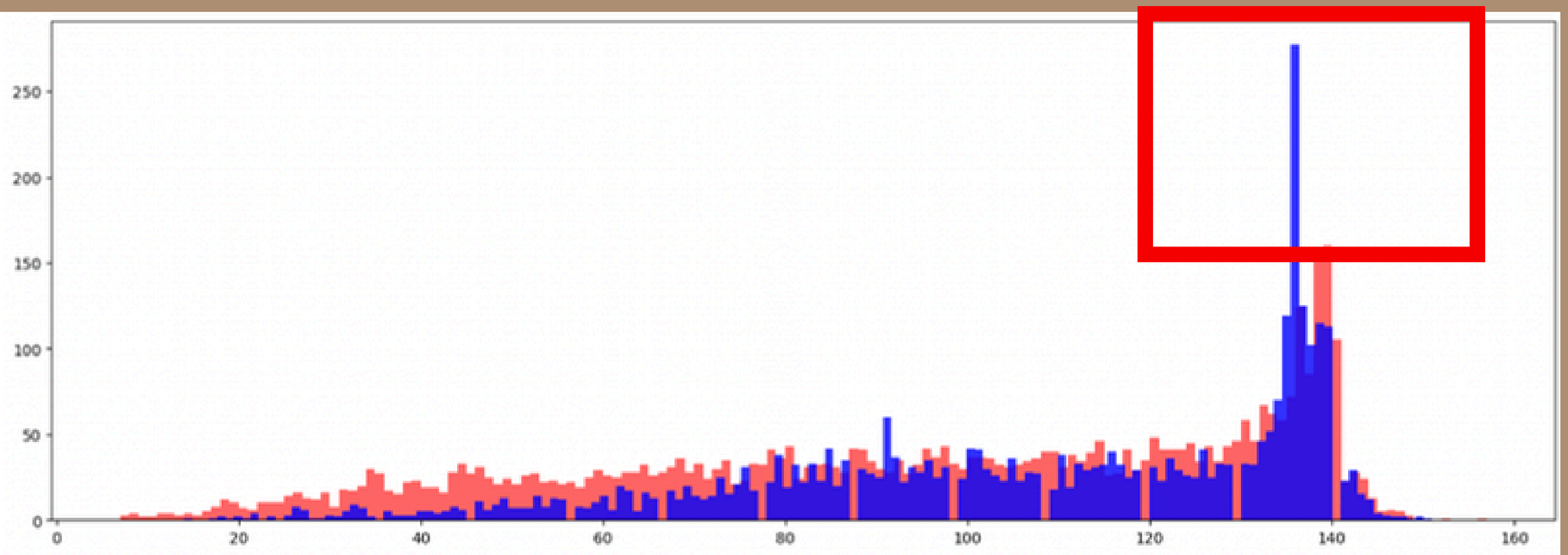
紅色：非真實災難推文
藍色：真實災難推文

y軸：各長度範圍推文數量



x軸：推文長度（以字符數作為計算）

DEMO



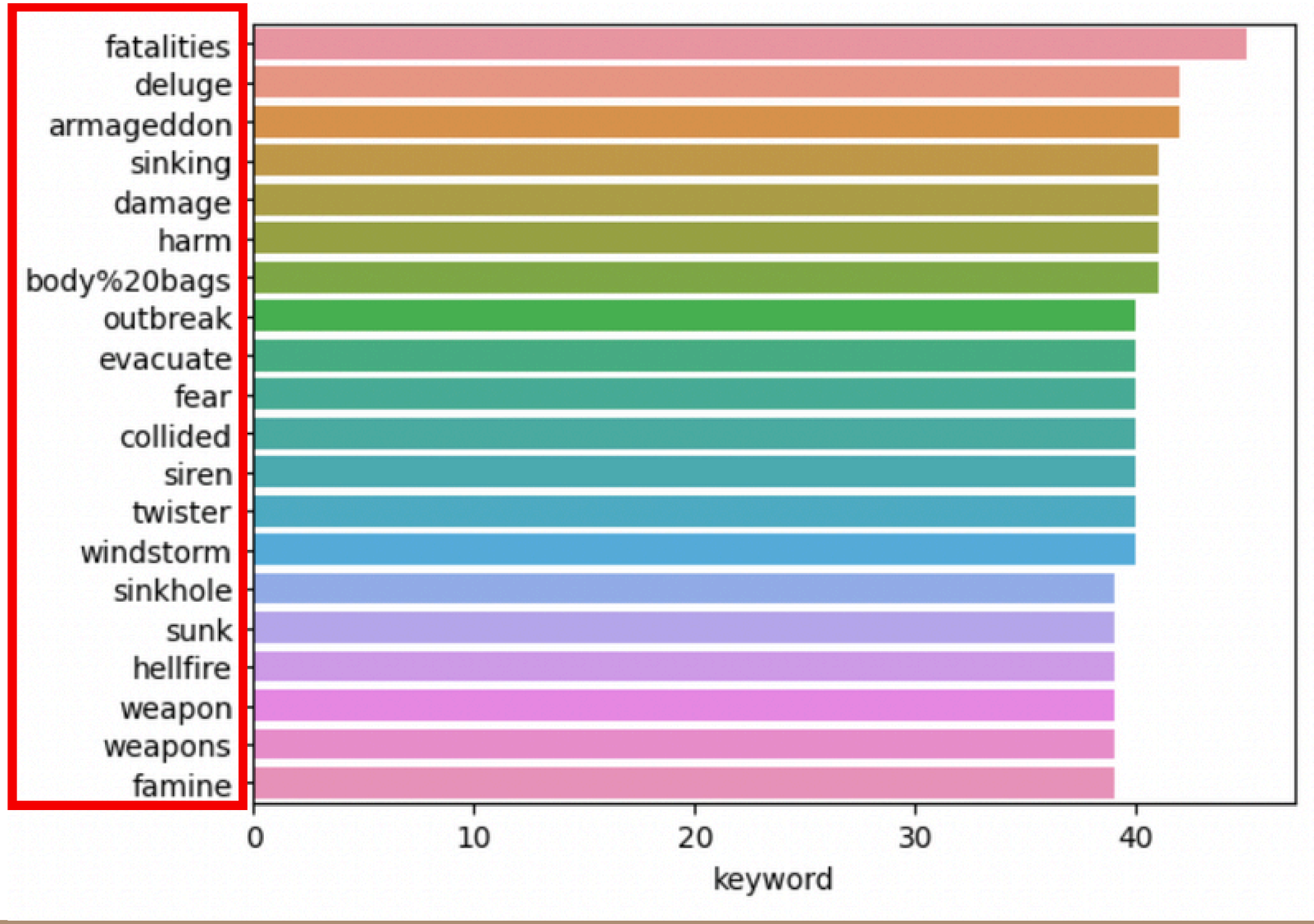

```
# plot -> value -> length
plt.rcParams['figure.figsize'] = (18.0, 6.0)
bins = 150
plt.hist(tweet[tweet['target'] == 0]['length'], alpha=0.6, bins=bins, label='Not', color='red')
plt.hist(tweet[tweet['target'] == 1]['length'], alpha=0.8, bins=bins, label='Real', color='blue')
plt.show()
```

target:0 非真實災難推文
target:1 真實災難推文

文本字符長度

alpha值：直方圖透明度

2. 訓練集頻率前20高關鍵字



DEMO

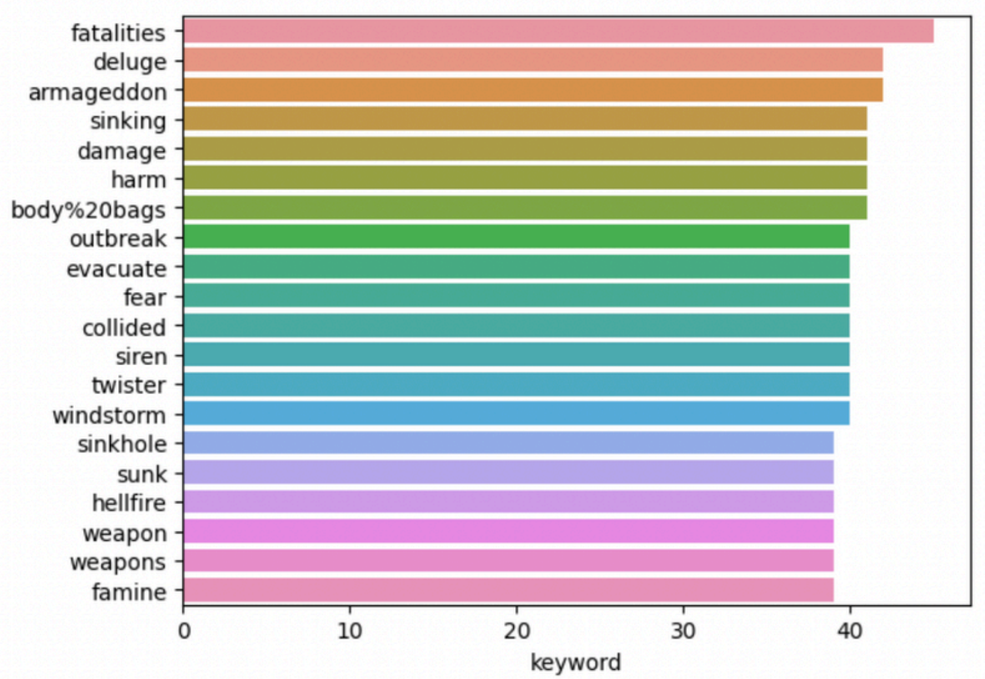
```
sns.barplot(y=train['keyword'].value_counts()[:20].index,x=train['keyword'].value_counts()[:20],orient='h')
```

```
<AxesSubplot:xlabel='keyword'>
```

計算訓練集裡每個關鍵字出現的次數，並選取前20個出現頻率最高的關鍵字

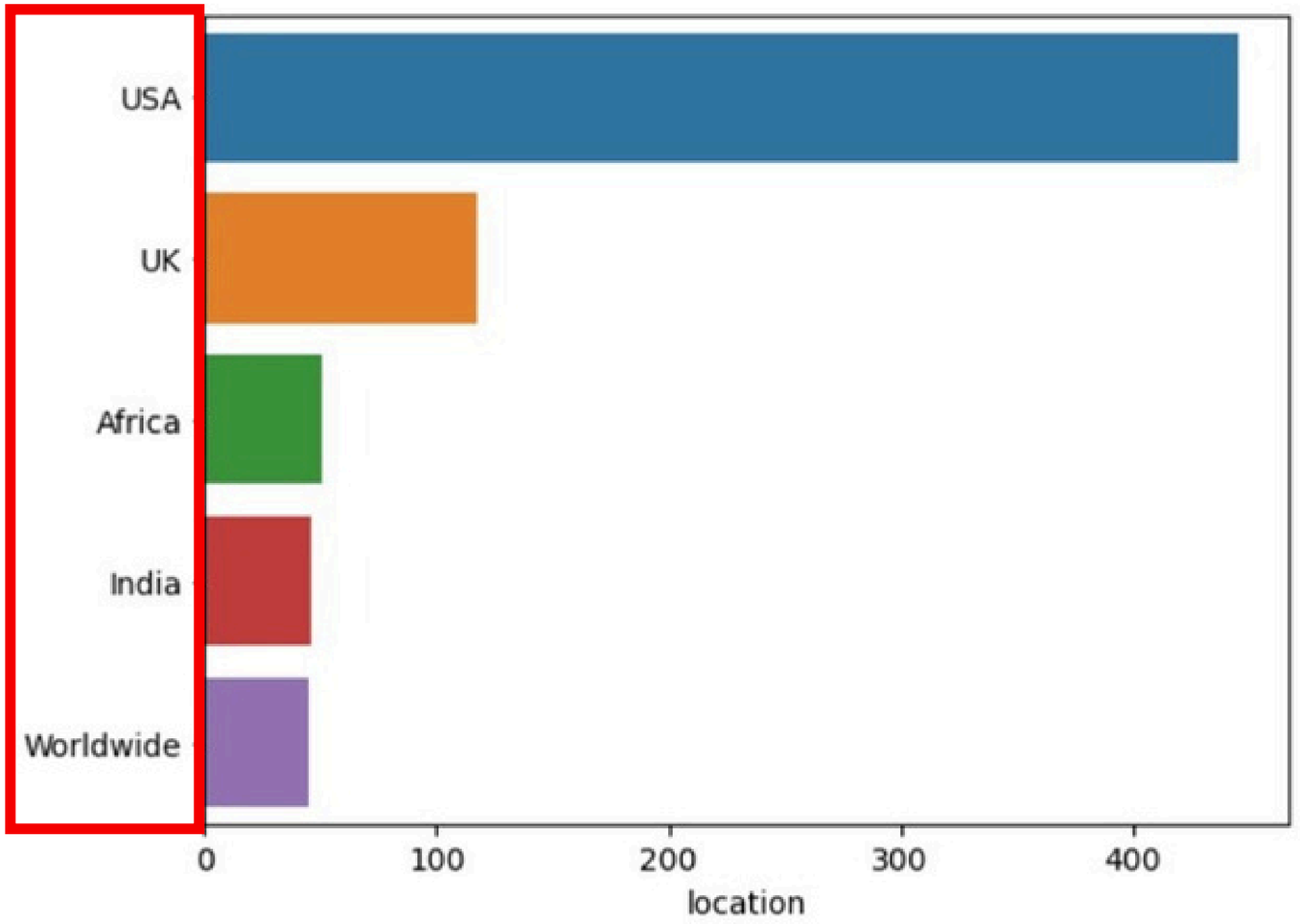
x參數為關鍵字出現的次數

y參數為關鍵字的索引



3. 災難相關推文最常出現的前五個國家

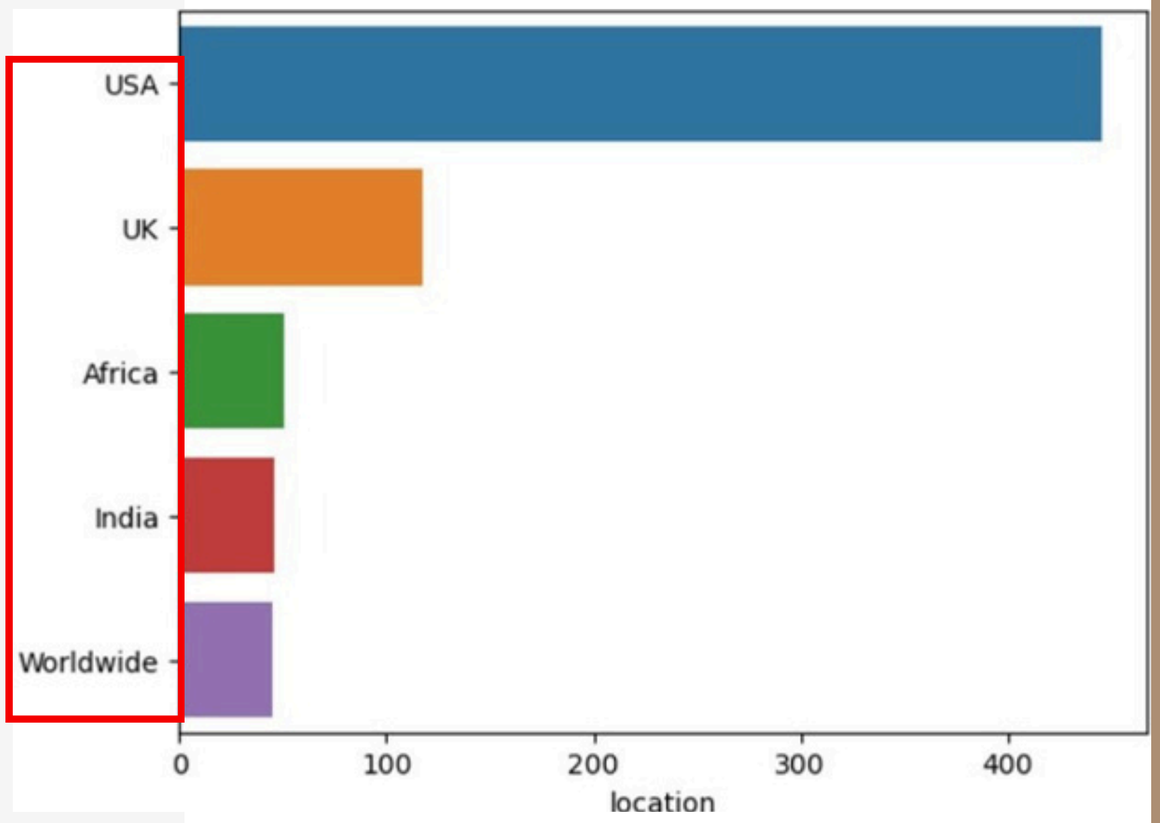
```
Out[17]: <AxesSubplot:xlabel='location'>
```



DEMO

```
# Replacing the ambigious locations name with Standard names
train['location'].replace({'United States':'USA',
                           'New York':'USA',
                           "London":'UK',
                           "Los Angeles, CA":'USA',
                           "Washington, D.C.":'USA',
                           "California":'USA',
                            "Chicago, IL":'USA',
                            "Chicago":'USA',
                           "New York, NY":'USA',
                           "California, USA":'USA',
                           "FLorida":'USA',
                           "Nigeria":'Africa',
                           "Kenya":'Africa',
                           "Everywhere":'Worldwide',
                           "San Francisco":'USA',
                           "Florida":'USA',
                           "United Kingdom":'UK',
                           "Los Angeles":'USA',
                           "Toronto":'Canada',
                           "San Francisco, CA":'USA',
                           "NYC":'USA',
                           "Seattle":'USA',
                           "Earth":'Worldwide',
                           "Ireland":'UK',
                           "London, England":'UK',
                           "New York City":'USA',
                           "Texas":'USA',
                           "London, UK":'UK',
                           "Atlanta, GA":'USA',
                           "Mumbai":"India"},inplace=True)
```

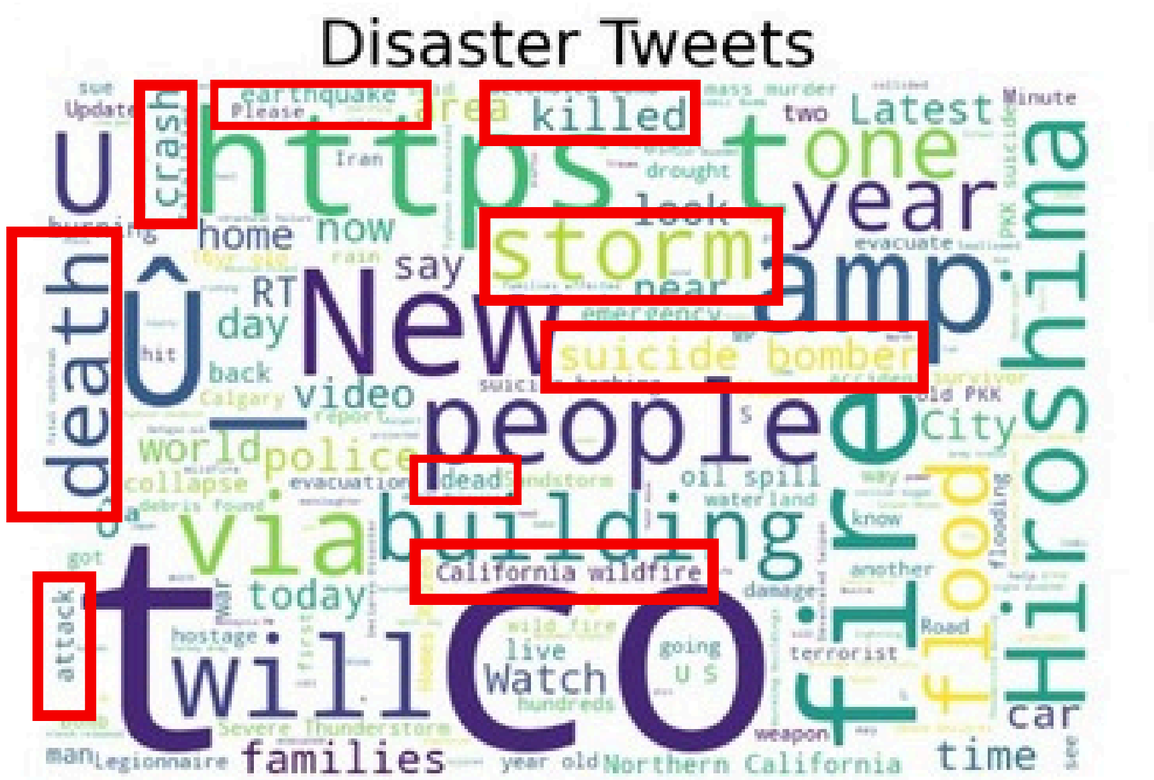
```
sns.barplot(y=train['location'].value_counts()[:5].index,x=train['location'].value_counts()[:5],
            orient='h')
```

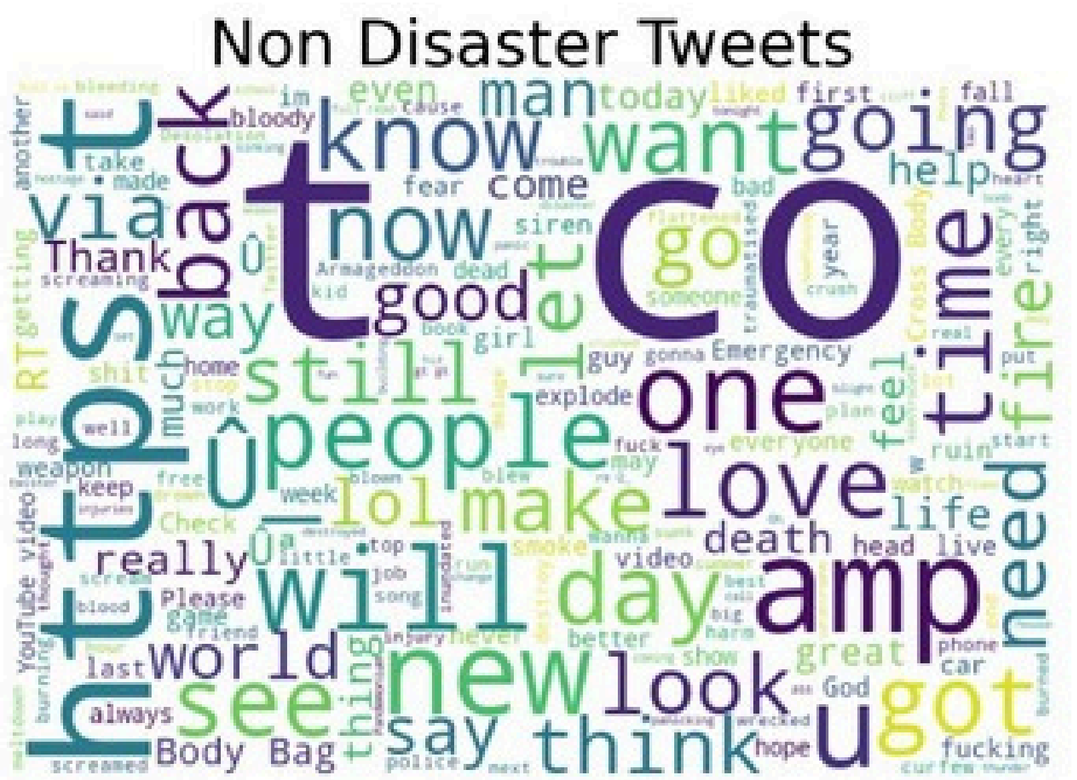


將模糊的地點名稱替換成標準國家名稱

計算前五個出現次數最多的國家

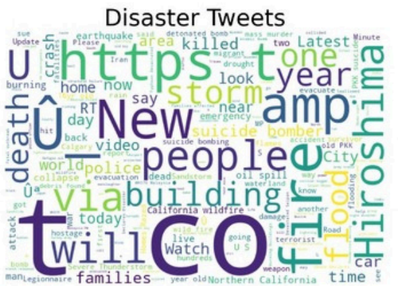
真實災難VS非真實災難推文主題比較

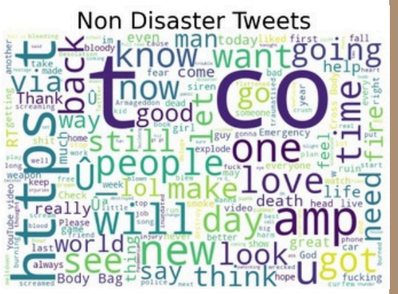




Word Cloud 文字雲圖

DEMO






```
from wordcloud import WordCloud

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=[26, 8])
wordcloud1=WordCloud(background_color='white',width=600,height=400).generate(" ".join(disaster_tweets))

ax1.imshow(wordcloud1)
ax1.axis('off')
ax1.set_title('Disaster Tweets',fontsize=40);

wordcloud2 = WordCloud( background_color='white',
                        width=600,
                        height=400).generate(" ".join(non_disaster_tweets))
ax2.imshow(wordcloud2)
ax2.axis('off')
ax2.set_title('Non Disaster Tweets',fontsize=40);
```

單詞的出現頻率越高 -> 單詞在文字雲圖中就會越大

03

IMPLEMENT

01

DATA PRE-PROCESSING

缺失値情況檢測

Missing values in Train dataset

```
print("Missin values in Train dataset : ")
null_data = train_df.isnull().sum().to_frame().rename(columns={0:'Miss values'})
null_data["Miss values percentage %"] = train_df.isnull().sum() * 100 / len(train_df)
null_data.style
```

```
Missin values in Train dataset :
```

	Miss values	Miss values percentage %
id	0	0.000000
keyword	56	0.744582
location	2490	33.107300
text	0	0.000000
target	0	0.000000

isnull()檢測缺失值

缺失值情況檢測及處理

檢測缺失值數量

刪除兩個欄位

```
In [5]: train.isnull().sum()

Out[5]: id              0
        keyword        61
        location     2533
        text            0
        target          0
        dtype: int64
```

```
In [6]: train = train.drop(columns = ['keyword','location'])
        train.dtypes

Out[6]: id          int64
        text       object
        target      int64
        dtype: object
```

DATA PRE-PROCESSING

```
def preprocessing(text):
    text = text.lower()
    pattern = re.compile('[^a-z]')
    words = nltk.word_tokenize(text)
    stop_words = set(nltk.corpus.stopwords.words('english'))
    words = [PorterStemmer().stem(word) for word in words if word.lower() not in stop_words]
    preprocessed_text = ' '.join(words)
    return preprocessed_text
```

1. 轉換為小寫
2. 正規化
3. 英文分詞
4. 去除停用字
5. 詞幹提取

Cleaning text 文本清理

Cleaning text

```
def text_cleaner(text):

    text=text.lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text

train_df['text']=train_df['text'].apply(lambda x :text_cleaner(x))
test_df['text']=test_df['text'].apply(lambda x :text_cleaner(x))
```

```
train_df['text'].sample(10)
```

```
7142     well im also gay but girls like some too so  ...
2375    temecafreeman gm i pray any attack of the enem...
4587    health fact  of muscle mass is made up of flui...
6721    severe thunderstorm warnings have been cancell...
5051    ral½l sends a message of condolence to vietnam...
6335    the result of failure in correcting structural...
2198    interesting  aircraft debris found on la reuni...
3003    dust devil maintenance fee  buy up la rotary s...
1839    honestly nightmarish god driving to new places...
2521    free kindle book  aug   thriller  desolation r...
```

1.text.lower():將文本轉換為小寫字母

2.'\[.*?\]'：方框內的內容

3.'https?://\S+|www\.\S+':URL連結

4.'<.*?>+': HTML標籤

5.'[%s]' ：移除標點符號

6.'\n':換行符

7.'\w*\d\w*':移除包含數字的單詞

Tokenization process 分詞

Tokenization process

```
Tokenizer=RegexpTokenizer(r'\w+')
train_df['text']=train_df['text'].apply(lambda x : Tokenizer.tokenize(x))
test_df['text']=test_df['text'].apply(lambda x : Tokenizer.tokenize(x))
```

```
train_df.sample(10)
```

	id	keyword	location	text	target
2198	3150	debris	Bristol, UK	[interesting, aircraft, debris, found, on, la,...	1
4476	6366	hostages	china	[hot, specially, modified, to, land, in, a, st...	0
4202	5965	hazard	USA	[biggest, lead, hazard, in, new, england, hist...	0
195	274	ambulance	‖ c h i c a g o ‖	[when, you, dont, know, which, way, an, ambula...	1
2170	3112	debris	NaN	[aircraft, debris, found, on, la, reunion, is,...	1
4349	6175	hijack	NaN	[rickybonessxm, fuck, specially, there, new, s...	0
2108	3027	death	NaN	[going, to, starve, to, death]	0
4920	7008	mayhem	GLOBAL/WORLDWIDE	[raynbowaffair, editor, in, chief, diamondkesa...	0
2102	3019	death	NaN	[rss, judge, orders, texas, to, recognize, spo...	0
6421	9181	suicide%20bomber	NaN	[bbclaurak, why, is, no, one, talking, about, ...	1

正規表達式分詞器（RegexpTokenizer)

r'\w+':將文本的內容切成一個個單詞
以進行分詞處理

02

MODEL BUILDING

Text Feature Extraction 文本特徵提取

TF-IDF

- 考慮詞的**頻率**以及詞在整個語料庫中的**重要性**
- 能夠捕捉詞的重要性和區分能力
  以助區分不同文本之間的特徵
- 通過將詞頻乘以逆文檔頻率因子
  得到一個更具區分能力的特徵表示
- 適用於需要考慮**詞的重要性和權重**的文本分析任務

Counter Vectorization

- 將文本轉為**詞頻矩陣**
  以統計每個詞在文本中出現的次數
- 不考慮詞的重要性及權重
- 生成的特徵向量稀疏
- 適用於簡單的文本分類任務

TF-IDF

Using TfidfVectorizer on our text

```
tfidf=TfidfVectorizer(min_df=2,max_df=0.5,ngram_range=(1,2))

train_tfidf=tfidf.fit_transform(train_df['text'])
test_tfidf = tfidf.transform(test_df["text"])
```

Spliting data to train and test (TF_IDF)

```
X_test_tf , X_train_tf , y_test_tf , y_train_tf = train_test_split(train_tfidf , train_df['target'] ,
                                                                   test_size=0.2 , random_state=1)
```

最小文檔頻率（min_df）：表示詞至少在幾個文檔中出現才會被考慮為特徵

最大文檔頻率（max_df）：它表示詞在文檔中的出現頻率的上限

過濾掉在太多文檔中出現的常見詞，以保留那些更具特殊性和區分能力的詞

N-gram範圍（ngram_range）:指定詞的連續子序列的範圍

可以捕捉到更多詞之間的關聯性和上下文訊息

Counter Vectorization

Using CounterVectorizer on our text

```
count_vectorizer=CountVectorizer()
train_vectors=count_vectorizer.fit_transform(train_df['text'])

test_vectors = count_vectorizer.transform(test_df["text"])
```

對訓練集的文本數據進行向量化

對測試集的文本數據進行向量化

```
print(train_vectors[0].todense())
```

Spliting data to train and test (CounterVectorizer)

```
X_train , X_test , y_train , y_test  = train_test_split(train_vectors , train_df['target'] , test_size=0.2 ,
random_state=1)
```

CountVectorizer():此向量化器將文本轉換為計數矩陣

統計每個詞在文本中出現的次數

Models we use

SVM

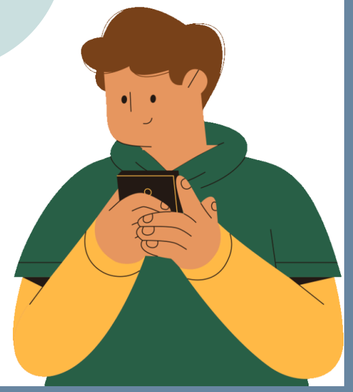
Sequential

Naive Bayes

Logistic Regression

Random ForestClassifier

XGBoost



Why we use these models

- **SVM**

1.**SVM是一種強大的分類算法** ⇨ 適用於**二分類**和**多分類任務**

2.**解決高維問題**：自然語言處理中的特徵維度通常很高,SVM具有**處理高維數據**的優勢

⇨ 它可以有效**處理大量特徵**並找到有效的**分類邊界**

3.**對於小數據集有效**：SVM在處理小數據集時通常表現良好

⇨ SVM可以**避免過度擬合**和處理較少的樣本

SVM

CounterVectorizer

```
SVC_CounterVectorizer=SVC()
model_prediction(SVC_CounterVectorizer)
```

準確率80.8%

```
Accuracy_Score of SVC model on Training Data is: 94.68085106382979
Accuracy_Score of SVC model on Testing Data is: 80.81524147097917

--------------------------------------------------------------------------------
Precision Score of SVC model is: 0.8728571428571429
Recall Score of SVC model is: 0.6397905759162303
F1 Score of SVC model is: 0.738368580060423
--------------------------------------------------------------------------------
```

TF_IDF

```
SVC_tf=SVC()
model_prediction_tf(SVC_tf)
```

準確率76%

```
Accuracy_Score of SVC model on Training Data is: 97.7403633141338
Accuracy_Score of SVC model on Testing Data is: 76.709726443769

--------------------------------------------------------------------------------
Precision Score of SVC model is: 0.8606615059817 03
Recall Score of SVC model is: 0.5433140826299423
F1 Score of SVC model is: 0.6661220043572985
--------------------------------------------------------------------------------
```

模型正確**預測**為真實災難事件的樣本

比例為87.3% / 86%

模型正確**檢測**到實際真實災難事件

比例為63.9 % / 54.3%

F1值是精確率和召回率的加權調和平均值

模型的F1值為73.8% / 66.6%

越接近1表示模型的**預測能力**越好

Why we use these models

- **Naive Bayes**

1. **簡單快速**：此模型是一個非常簡單和易於實現的分類算法

➪ 訓練和預測過程都**非常快速**,特別適用於大型數據集

2.**小數據集效果良好**：即使在小數據集上,此模型也能提供合理的性能

➪ 通常比其他更複雜的分類方法**需要更少的訓練數據**

3.**處理高維數據**：此模型在處理高維特徵空間的文本數據時表現出色

➪它能有效地處理**大量的特徵**,例如單詞在**文本中的頻率或詞袋模型**

4. **對於獨立特徵的假設**：在自然語言處理中，某些特徵可能在文本中**相對獨立**(ex:單詞)

➪這使得Naive Bayes成為一個合理的選擇

Naive Bayes

CounterVectorizer

準確率81%

```
Gussian_CounterVectorizer=MultinomialNB()
model_prediction(Gussian_CounterVectorizer)
```

```
Accuracy_Score of MultinomialNB model on Training Data is: 90.99544072948328
Accuracy_Score of MultinomialNB model on Testing Data is: 80.94816127603013

--------------------------------------------------------------------------------
Precision Score of MultinomialNB model is: 0.7959413754227734
Recall Score of MultinomialNB model is: 0.7392670157068063
F1 Score of MultinomialNB model is: 0.7665580890336591
--------------------------------------------------------------------------------
```

TF_IDF

準確率77%

```
Gussian_tf=MultinomialNB()
model_prediction_tf(Gussian_tf)
```

```
Accuracy_Score of MultinomialNB model on Training Data is: 90.03101462117856
Accuracy_Score of MultinomialNB model on Testing Data is: 77.45060790273556

--------------------------------------------------------------------------------
Precision Score of MultinomialNB model is: 0.8609226594301221
Recall Score of MultinomialNB model is: 0.5637494446912483
F1 Score of MultinomialNB model is: 0.6813422818791947
--------------------------------------------------------------------------------
```

模型正確預測為真實災難事件的樣本

比例為79.5% / 86%

模型正確檢測到實際真實災難事件

比例為74 % / 56.3%

F1值是精確率和召回率的加權調和平均值

模型的F1值為76.6% / 68.1%

越接近1表示模型的預測能力越好

Why we use these models

- **Logistic Regression**

1.**解釋性強**：此模型通過計算每個特徵的**權重**（係數）來**評估對目標的貢獻程度**

➡ 因此可以理解每個特徵對預測結果的影響

2.**線性分類效果好**：當文本數據具有線性可分性時,此模型可以提供**很好的分類效果**

➡ 能夠找到一個最佳的線性邊界來區分不同的文本類別

3.**快速訓練和預測**：此模型的訓練和預測過程非常高效

➡ 它通常適用於**大型數據集**,訓練速度快,預測速度也相對快速

4.**可處理二元和多元分類**：此模型可以用於處理**二元分類和多元分類**的問題

➡ 在NLP中,此模型常用於將文本分類到多個類別中

Logistic Regression

CounterVectorizer

```
LogesticReg_CounetrVectorizer=LogisticRegression()
model_prediction(LogesticReg_CounetrVectorizer)
```

準確率80%

```
Accuracy_Score of LogisticRegression model on Training Data is: 95.26975683890578
Accuracy_Score of LogisticRegression model on Testing Data is: 80.01772264067345

---------------------------------------------------------------------------
Precision Score of LogisticRegression model is: 0.815
Recall Score of LogisticRegression model is: 0.6827225130890052
F1 Score of LogisticRegression model is: 0.743019943019943
```

模型正確**預測**為真實災難事件的樣本

➡ 比例為81.5% / 85.4%

模型正確**檢測**到實際真實災難事件

➡ 比例為68.2 % / 56.7%

TF_IDF

```
LogesticReg_tf=LogisticRegression()
model_prediction_tf(LogesticReg_tf)
```

準確率77%

```
Accuracy_Score of LogisticRegression model on Training Data is: 90.91714665485158
Accuracy_Score of LogisticRegression model on Testing Data is: 77.35562310030394

---------------------------------------------------------------------------
Precision Score of LogisticRegression model is: 0.8537074148296593
Recall Score of LogisticRegression model is: 0.567747667703243
F1 Score of LogisticRegression model is: 0.6819637139807898
```

F1值是精確率和召回率的加權調和平均值

➡ 模型的F1值為74% / 68%

越接近1表示模型的**預測能力**越好

Why we use these models

- **Random ForestClassifier**

1.**高準確性**：此模型能夠結合多個決策樹的預測結果,並以平均的方式進行最終預測

👉 可提高整體的分類準確性,特別是當單個決策樹**容易過擬合時**

2.**處理高維數據**：在NLP中,文本特徵常常具有**高維性**（大量的單詞或詞語）

👉此模型能夠處理高維特徵空間,並找到重要特徵對於分類任務的貢獻

3.**自動特徵選擇**：此模型通常具有**內置的特徵選擇機制**

👉它可以計算**特徵的重要性**,並根據這些重要性選擇最佳的特徵

4.**可解釋性**：此模型可以提供特徵的**重要性排序**,用於解釋模型的決策過程

👉對理解NLP任務中的特徵和影響因素非常有價值

Random Forest

CounterVectorizer

準確率79%

```
RandomForset_CounterVectorizer=RandomForestClassifier(n_estimators=100)
model_prediction(RandomForset_CounterVectorizer)

Accuracy_Score of RandomForestClassifier model on Training Data is: 98.86018237082067
Accuracy_Score of RandomForestClassifier model on Testing Data is: 79.84297740363315

---------------------------------------------------------------------------

Precision Score of RandomForestClassifier model is: 0.8310439560439561
Recall Score of RandomForestClassifier model is: 0.6335078534031413
F1 Score of RandomForestClassifier model is: 0.718954248366013
```

模型正確**預測**為真實災難事件的樣本

比例為83.1% / 81.6%

模型正確**檢測**到實際真實災難事件

比例為63.4 % / 54.3%

TF_IDF

```
RandomForset_tf=RandomForestClassifier(n_estimators=100)
model_prediction_tf(RandomForset_tf)

Accuracy_Score of RandomForestClassifier model on Training Data is: 98.89233495790873
Accuracy_Score of RandomForestClassifier model on Testing Data is: 75.2659574468085

---------------------------------------------------------------------------

Precision Score of RandomForestClassifier model is: 0.8165443629086058
Recall Score of RandomForestClassifier model is: 0.543758329631275
F1 Score of RandomForestClassifier model is: 0.6527999999999999
```

準確率75%

F1值是精確率和召回率的加權調和平均值

模型的F1值為72% / 65%

越接近1表示模型的**預測能力**越好

Why we use these models

- **XGBoost**

1.**高準確性**：此模型是一個強大的集成學習算法

➡ 通過結合多個弱學習器（決策樹）的預測結果,可以**提高整體的分類準確性**

2.**高效處理高維特徵**：NLP中的文本數據往往具有高維特徵

➡此模型具有良好的處理高維數據的能力,可以有效地利用這些特徵來進行預測

3.**特徵重要性評估**：此模型能夠計算特徵的重要性

➡幫助了解哪些特徵對於預測結果的貢獻最大

4.**處理缺失值**：此模型能夠有效處理**包含缺失值的數據**

➡可將缺失值分配到左子樹或右子樹中,從而充分利用有限的訊息

XGBoost

CounterVectorizer

```
XGB_CounterVectorizer=XGBClassifier()
model_prediction(XGB_CounterVectorizer)
```

準確率78%

```
Accuracy_Score of XGBClassifier model on Training Data is: 83.41565349544074
Accuracy_Score of XGBClassifier model on Testing Data is: 78.33407177669473

------------------------------------------------------------------------
Precision Score of XGBClassifier model is: 0.8357348703170029
Recall Score of XGBClassifier model is: 0.6073298429319371
F1 Score of XGBClassifier model is: 0.703456640388114
------------------------------------------------------------------------
```

模型正確**預測**為真實災難事件的樣本

比例為83.5% / 77.1%

模型正確**檢測**到實際真實災難事件

比例為60.7 % / 58.9%

TF_IDF

```
XGB_tf=XGBClassifier()
model_prediction_tf(XGB_tf)
```

準確率74%

```
Accuracy_Score of XGBClassifier model on Training Data is: 89.41072219760744
Accuracy_Score of XGBClassifier model on Testing Data is: 74.94300911854104

------------------------------------------------------------------------
Precision Score of XGBClassifier model is: 0.7712456344586729
Recall Score of XGBClassifier model is: 0.5886272767658818
F1 Score of XGBClassifier model is: 0.6676744771982867
------------------------------------------------------------------------
```

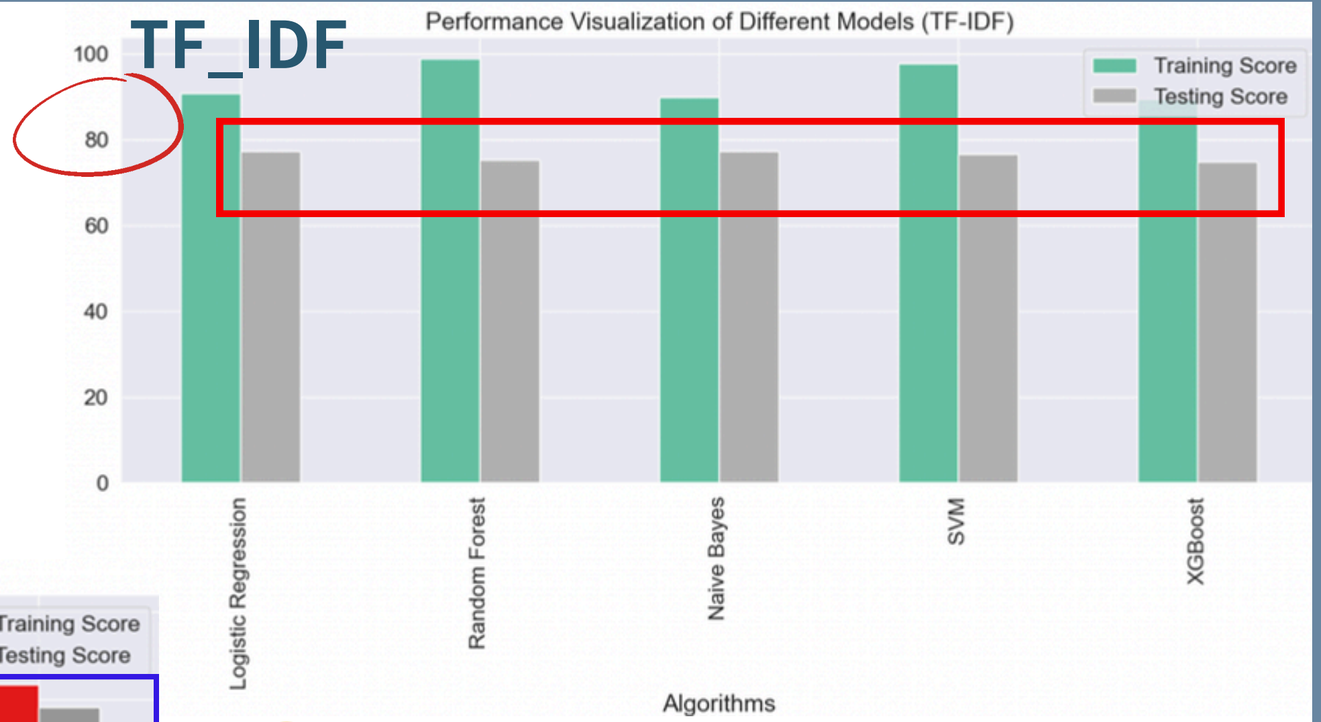
F1值是精確率和召回率的加權調和平均值

模型的F1值為70.3% / 66.8%

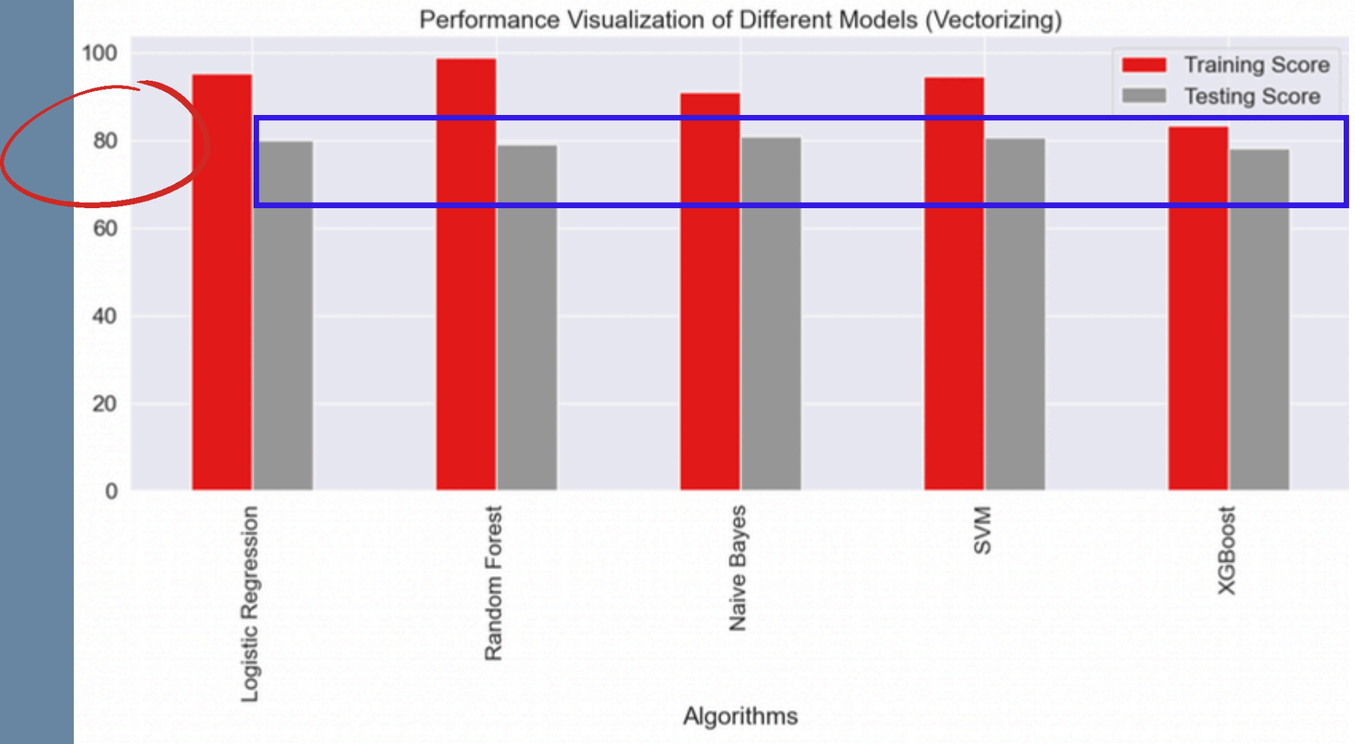
越接近1表示模型的**預測能力**越好

Comparison

TF_IDF



CounterVectorizer



根據圖表可得以下結論：

文本經CounterVectorization 特徵提取後

比經過TF_IDF特徵提取後的模型訓練準確率高

準確率皆達8成

Why we use these models

- **Sequential**

1.**適應不同長度的輸入**：

➡ 此模型可以靈活地處理**不同長度的輸入序列**並保持模型一致性的同時進行有效的預測

2.**適用於序列標記任務**：此模型對於序列標記任務特別有效

➡ 它可以將**每個位置的輸入映射到對應的標記或類別**

3.**學習長期依賴關係**：此模型能夠透過**長短期記憶（LSTM）**單元等進行訓練

➡ 以學習和捕捉句子中彼此的**上下文關係**

4.**詞嵌入的應用**：此模型可以有效地利用**詞嵌入（Word Embeddings）**來表示單詞

➡ 從而更好地捕捉單詞之間的**語義相似性和關聯性**

Sequential

```
# extract valid content
train = tweet_pad[:tweet.shape[0]]
test = tweet_pad[:tweet.shape[0]:]
# then train test split
X_train, X_test, y_train, y_test = train_test_split(train, tweet['target'].values, test_size=0.25)
X_train.shape, X_test.shape
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, 50, 100)           2280800

 spatial_dropout1d_3 (Spatia  (None, 50, 100)          0
 lDropout1D)

 lstm_3 (LSTM)               (None, 100)               80400

 dense_3 (Dense)             (None, 1)                 101

=================================================================
Total params: 2,361,301
Trainable params: 80,501
Non-trainable params: 2,280,800
```

選擇Sequential模型搭配LSTM

1.可充分利用LSTM在序列數據處理方面的優勢

2.更好地處理文本數據的序列特徵和上下文訊息

3.提高分類任務的準確性和性能

```
Epoch 44/50
58/58 - 3s - loss: 0.2171 - accuracy: 0.9129 - val_loss: 0.2332 - val_accuracy: 0.9123 - 3s/epoch - 52ms/step
Epoch 45/50
58/58 - 3s - loss: 0.2056 - accuracy: 0.9171 - val_loss: 0.2401 - val_accuracy: 0.9112 - 3s/epoch - 54ms/step
Epoch 46/50
58/58 - 3s - loss: 0.2105 - accuracy: 0.9168 - val_loss: 0.2565 - val_accuracy: 0.9044 - 3s/epoch - 47ms/step
Epoch 47/50
58/58 - 3s - loss: 0.2148 - accuracy: 0.9133 - val_loss: 0.2456 - val_accuracy: 0.9060 - 3s/epoch - 48ms/step
Epoch 48/50
58/58 - 3s - loss: 0.2214 - accuracy: 0.9128 - val_loss: 0.2525 - val_accuracy: 0.9081 - 3s/epoch - 50ms/step
Epoch 49/50
58/58 - 3s - loss: 0.2129 - accuracy: 0.9173 - val_loss: 0.2448 - val_accuracy: 0.9107 - 3s/epoch - 48ms/step
Epoch 50/50
58/58 - 3s - loss: 0.1971 - accuracy: 0.9231 - val_loss: 0.2643 - val_accuracy: 0.9049 - 3s/epoch - 47ms/step
```

```
batch_size=100, epochs=50
```

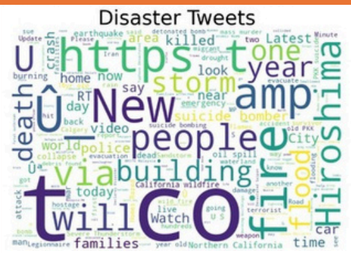
準確率唯一達9成

04
SUMMARY

FINAL RESULT

					TARGET
38	123	accident		robynjillllian wisdomteeth feel like im go accid teesha gonna come	0
39	124	accident	All Motorways, UK	northbound junction current delay min due accid c	1
40	125	accident	Frankfurt, Germany	daveoshri soembi say met accid week would super jelli dave p	0
41	127	accident	Gresham, OR	accid hit run cold block se vista ter gresham	1
42	140	accident		happen accid like	1

Target 0:推文文本有提及有關災難的關鍵字但並非與真實災難相關
Target 1:推文文本有提及有關災難性的關鍵字且與真實災難有關



SUMMARY

轉換為小寫

正規化

英文分詞

去除停用詞

詞幹提取

- 對數據可視化分析
- 進行多層數據前處理
- 使用6種模型及2種文本特徵提取方法
- 比較不同結果差異

SVM

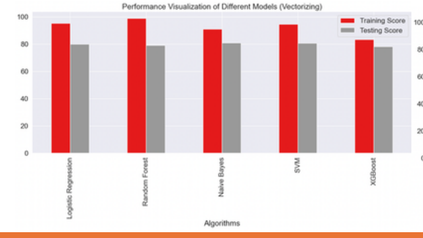
Sequential

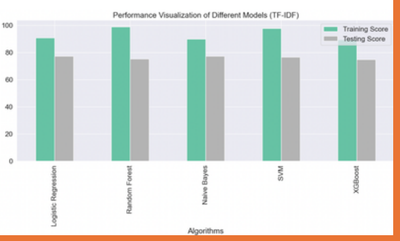
Naive Bayes

Logistic Regression

Random ForestClassifier

XGBoost





THANKS FOR LISTENING