Natural Language Processing with Disaster Tweets

108321015 許雯茹

Goal: predict which Tweets are about real disasters(1) and which ones are not(0).

Dataset:

使用 Kaggle 提供的 train.csv , test.csv 作為資料集

資料集	size	key(features)
Train	7613	id, text , location , keyword , target
Test	3263	id, text, location, keyword

columns	
Id	每個 tweet 的編號
Text	Tweets 內容
Location	Tweet 發送的地點
Keyword	災難分類
Target	只有 train.csv 有，為每則 tweet 的 label , 1 或 0

最後 test.csv 的測試結果存至 sample_submission.csv，再下載下來提交給 kaggle。

DataPreprocess:

1. 將 train set 的資料 shuffle 後再訓練，以避免資料間的相依性。
2. 針對 text 使用 CountVectorizer 建立字典，把所有的字收入字典，且根據每則推文出現的字產生 word vector (length=dictionary size)。

Ex; train_vector[0]=[0,0,0,…….,1,0,1]　　length(train_vector[0])=dic.length()

Model and dimensionality reduction:

我們把上面產生的 word vectors，作為 train set 丟入 classifier 訓練

測試的方式均使用 cross validation 分三堆（cv =3 accuracy = f1)。

我們使用 f1 score 的原因是資料集性質為 binary targets。

First test:

Model : RidgeClassifier with cross validation(cv=3) scoring by f1

```
array([0.59453669, 0.56498283, 0.64082434])
```

Second test:

把每個 text 句首/句尾加上其 keyword 之後，效果反而降低 6%

```
array([0.56524153, 0.5468095 , 0.58577822])
```

Model : LogisticRegression，f1 score 比 RidgeClassifier 提升 4%

```
[0.6387547  0.61347869 0.68350669]
```

=>需大量的 regularization(因為 text 維度大，資料量大) :

Model : RidgeClassifier(alpha=6.5 , default = 1.0)，f1 score 提升 10%

```
[0.72970843 0.67888101 0.74103272]
```

Model : SGDClassifier(alpha=0.1 , default = 0.0001)，f1 score 提升 7%

```
[0.64814815 0.65799842 0.71107607]
```

Third test :

將資料 shuffle 過後

Model :　RidgeClassifier(alpha =6.5)，f1 score 提升 6%

```
[0.80141844 0.78959811 0.79936933]
```

Model： SGDClassifier(alpha =0.1) ，f1 score 提升 0.5%

```
[0.71631206 0.69109535 0.67126527]
```

PCA TEST :

利用 **pca** 降低 text 的維度，配合 LogisticRegression(c=0.25)

pca(n_components = 0.99) scoring=accuracy

```
[0.79314421 0.79038613 0.803311  ]
```

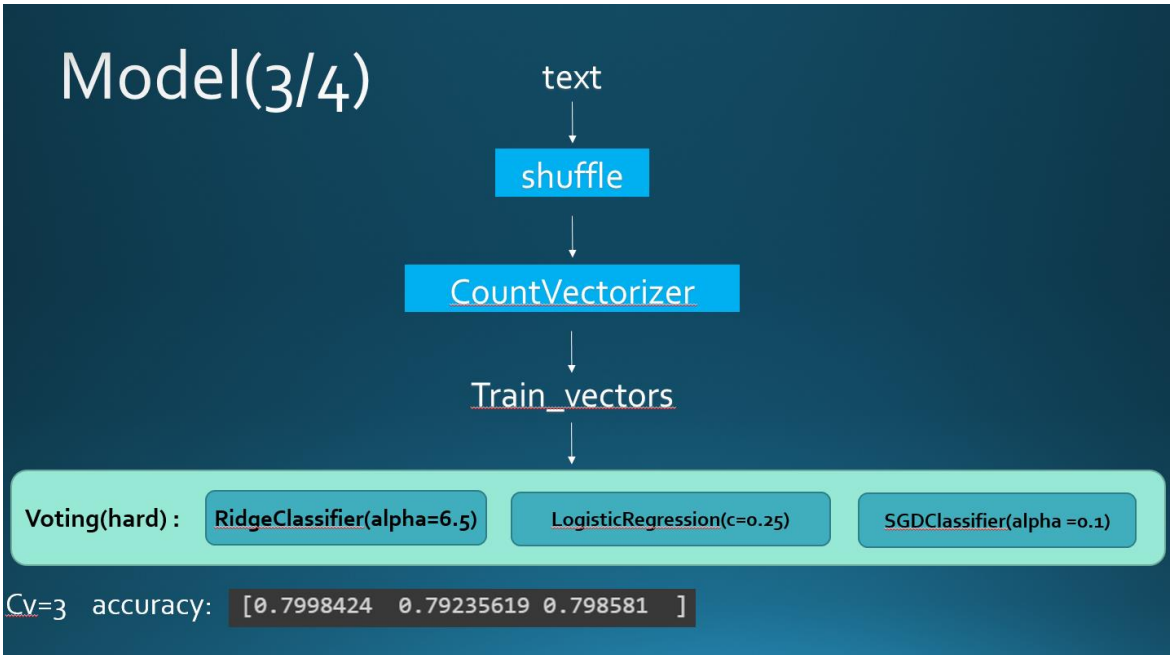
Kpca(n_components = 2, kernel = rbf, gamma=0.05) accuracy 約為 63%

降到二維，保留的資訊應該已經剩很少，但是他依然有 63%正確率。

效果不理想原因: 降維過多，且最後沒有 preimage 回原來的維度就丟入 scoring

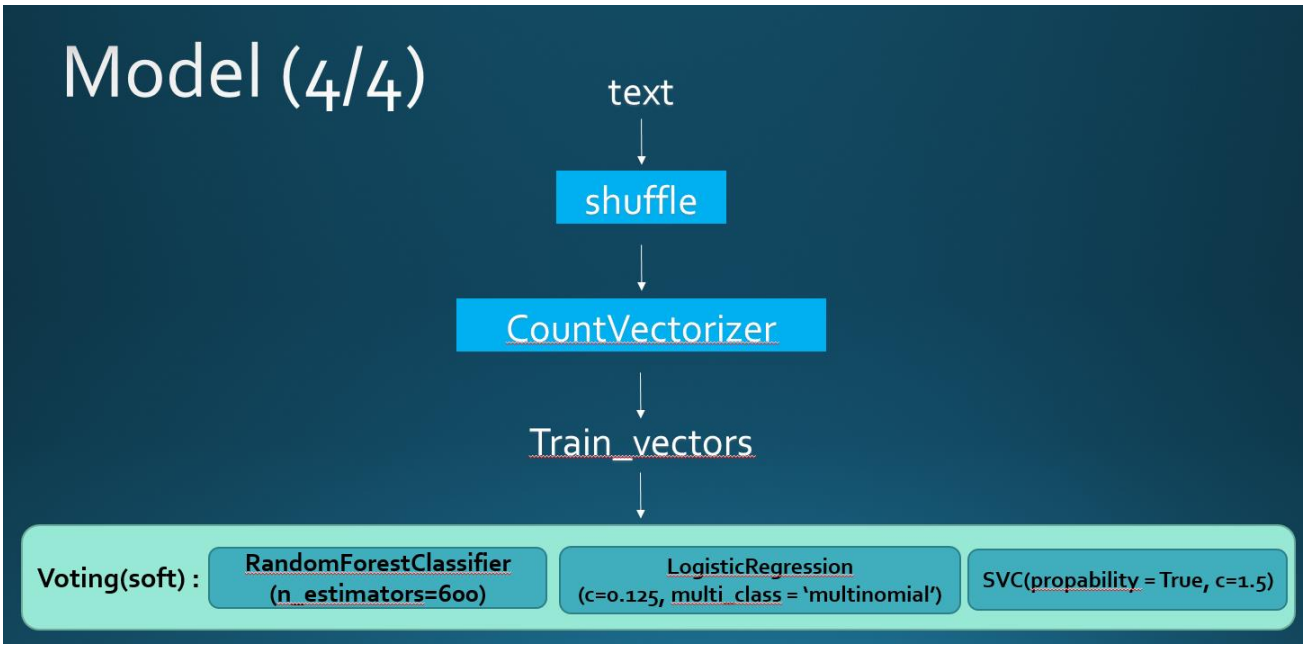
```
array([0.63356974, 0.63356974, 0.61647615])
```

Final tuning :



Model :
Voting (hard) => ridgeclassifier (alpha=6.5)
LogisticRegression(c=0.25)
SGDClassifier(alpha =0.1)
Accuracy: 0.7998424

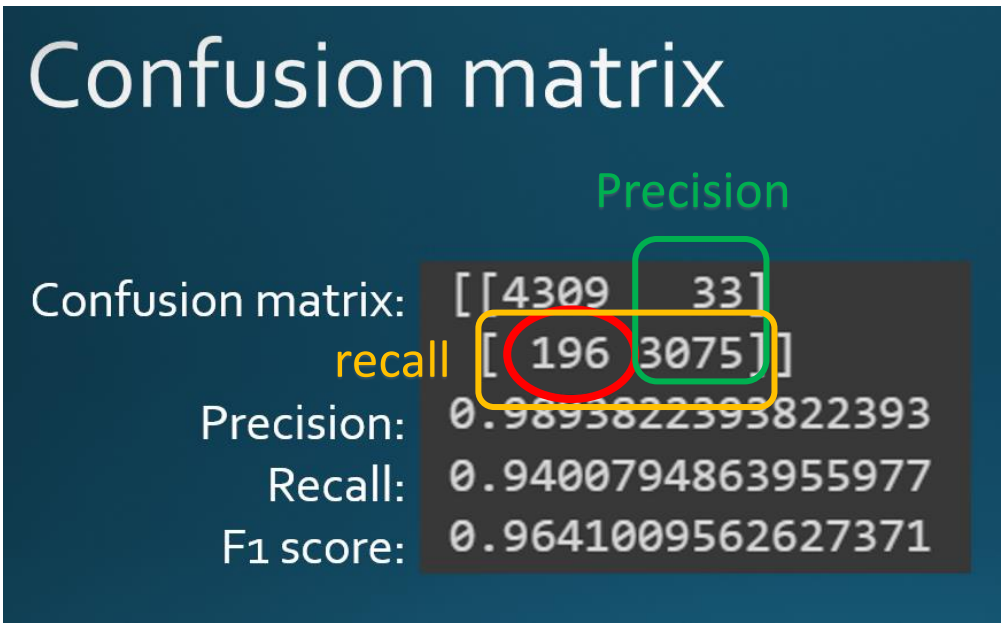
Hard voting 效果沒有比表現最好的 model 的效果好，我們認為是因為 voting 的 model 太少，沒辦法真正凸顯出 hard voting 的優點。因此決定改用 soft voting。



Model :
Voting (soft) => RandomForestClassifier(n_estimators=600)
LogisticRegression(c=0.125, multi_class = 'multinomial')
SVC(propability = True, c=1.5)
Accuracy: 0.80685849

雖說效果只比 hard voting 好一點，但 soft voting 確實是所有 classifier 中表現最好的，我們認為，只要再提升每個 classifier 的效能，少數的 model 也能用 soft voting 發揮最大的優勢。

```
lr = LogisticRegression(C=0.125,multi_class='multinomial',random_state=1)
svm = SVC(probability=True,C=1.5)
rnd_clf = RandomForestClassifier(n_estimators=600,random_state=1)
voting_clf=VotingClassifier(estimators=[('lr',lr),('rnd',rnd_clf('svm',svm)],voting='soft')
```



整體上只看個別數值都表現不錯，日後如要再改進 model，可以朝降低紅圈中的數值進步，因為此 task 為災難預測，FN 的數值偏高，代表有較多災難沒有被預測出來，這點是有待加強的。

提交上 kaggle 後的 accuracy 是使用 soft Voting Classifier： **0.80570** 排名 ：

277/867

277	pangru		0.80570