Stock Sentiment Analysis

Author : 李攀 21180330

朱泽邕 21180429

李晨 65180205

Abstract ：

In this research ,we mainly focus on the stock sentiment analysis based on news ,involved techniques include :

Data cleaning , remove stop words, and 6 models of NLP :

1. Random Forest
2. Logic regression
3. Naive Bayes
4. Gredient Boosting
5. SGD
6. KNN

We use cleaned data to train those models and get their corresponding accuracy

Keywords :

NLP , Data Cleaning,Deep learning

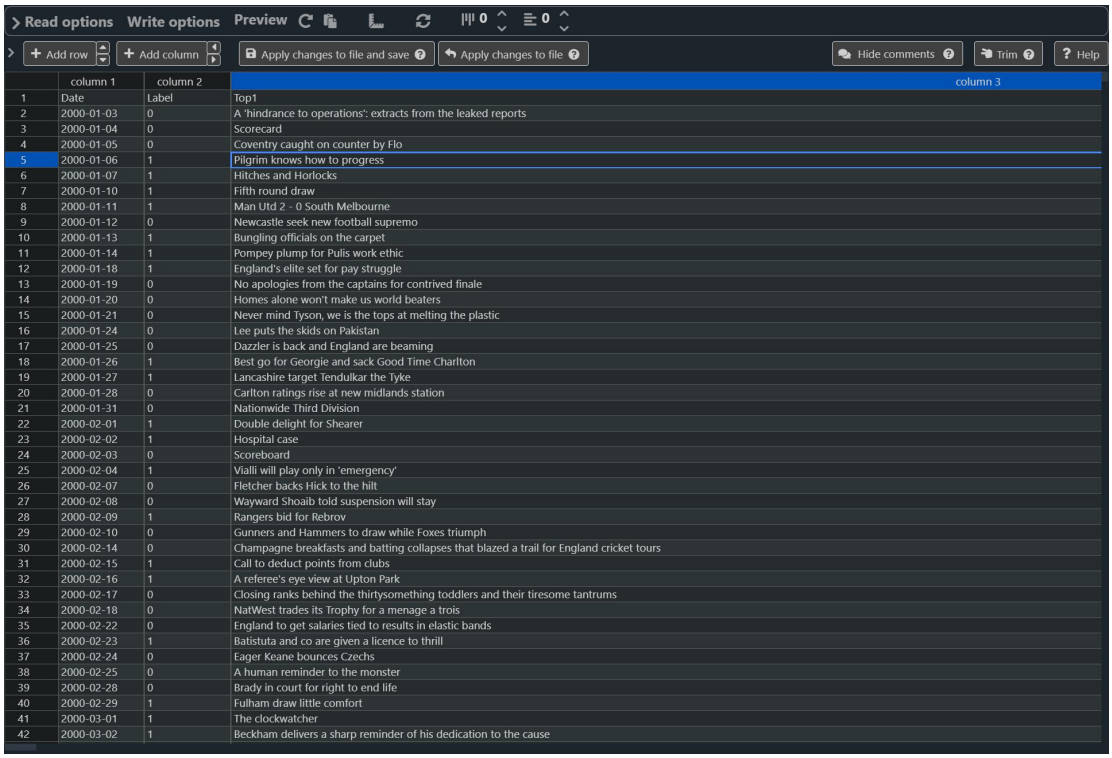
Data:

Our data are mainly gathered from Kaggle , and divide data into train set and test set based on date:

Train set : all news prior to 2015010 are categorized into train set

Test set: all news after 20141231 are categorized into train set

Table schema:



Read options Write options Preview

Add row Add column Apply changes to file and save Apply changes to file Hide comments Trim Help

	column 1	column 2	column 3
1	Date	Label	Top1
2	2000-01-03	0	A 'hindrance to operations': extracts from the leaked reports
3	2000-01-04	0	Scorecard
4	2000-01-05	0	Coventry caught on counter by Flo
5	2000-01-06	1	Pilgrim knows how to progress
6	2000-01-07	1	Hitches and Horlocks
7	2000-01-10	1	Fifth round draw
8	2000-01-11	1	Man Utd 2 - 0 South Melbourne
9	2000-01-12	0	Newcastle seek new football supremo
10	2000-01-13	1	Bungling officials on the carpet
11	2000-01-14	1	Pompey plump for Pulis work ethic
12	2000-01-18	1	England's elite set for pay struggle
13	2000-01-19	0	No apologies from the captains for contrived finale
14	2000-01-20	0	Homes alone won't make us world beaters
15	2000-01-21	0	Never mind Tyson, we is the tops at melting the plastic
16	2000-01-24	0	Lee puts the skids on Pakistan
17	2000-01-25	0	Dazzler is back and England are beaming
18	2000-01-26	1	Best go for Georgie and sack Good Time Charlton
19	2000-01-27	1	Lancashire target Tendulkar the Tyke
20	2000-01-28	0	Carlton ratings rise at new midlands station
21	2000-01-31	0	Nationwide Third Division
22	2000-02-01	1	Double delight for Shearer
23	2000-02-02	1	Hospital case
24	2000-02-03	0	Scoreboard
25	2000-02-04	1	Vialli will play only in 'emergency'
26	2000-02-07	0	Fletcher backs Hick to the hilt
27	2000-02-08	0	Wayward Shoaib told suspension will stay
28	2000-02-09	1	Rangers bid for Rebrov
29	2000-02-10	0	Gunners and Hammers to draw while Foxes triumph
30	2000-02-14	0	Champagne breakfasts and batting collapses that blazed a trail for England cricket tours
31	2000-02-15	1	Call to deduct points from clubs
32	2000-02-16	1	A referee's eye view at Upton Park
33	2000-02-17	0	Closing ranks behind the thirtysomething toddlers and their tiresome tantrums
34	2000-02-18	0	NatWest trades its Trophy for a menage a trois
35	2000-02-22	0	England to get salaries tied to results in elastic bands
36	2000-02-23	1	Batistuta and co are given a licence to thrill
37	2000-02-24	0	Eager Keane bounces Czechs
38	2000-02-25	0	A human reminder to the monster
39	2000-02-28	0	Brady in court for right to end life
40	2000-02-29	1	Fulham draw little comfort
41	2000-03-01	1	The clockwatcher
42	2000-03-02	1	Beckham delivers a sharp reminder of his dedication to the cause

layout :

Date : Label (1 for positive , 0 for negative) : Top1 - Top25 headline of that day

Process :

1. Read in csv file and find missing values

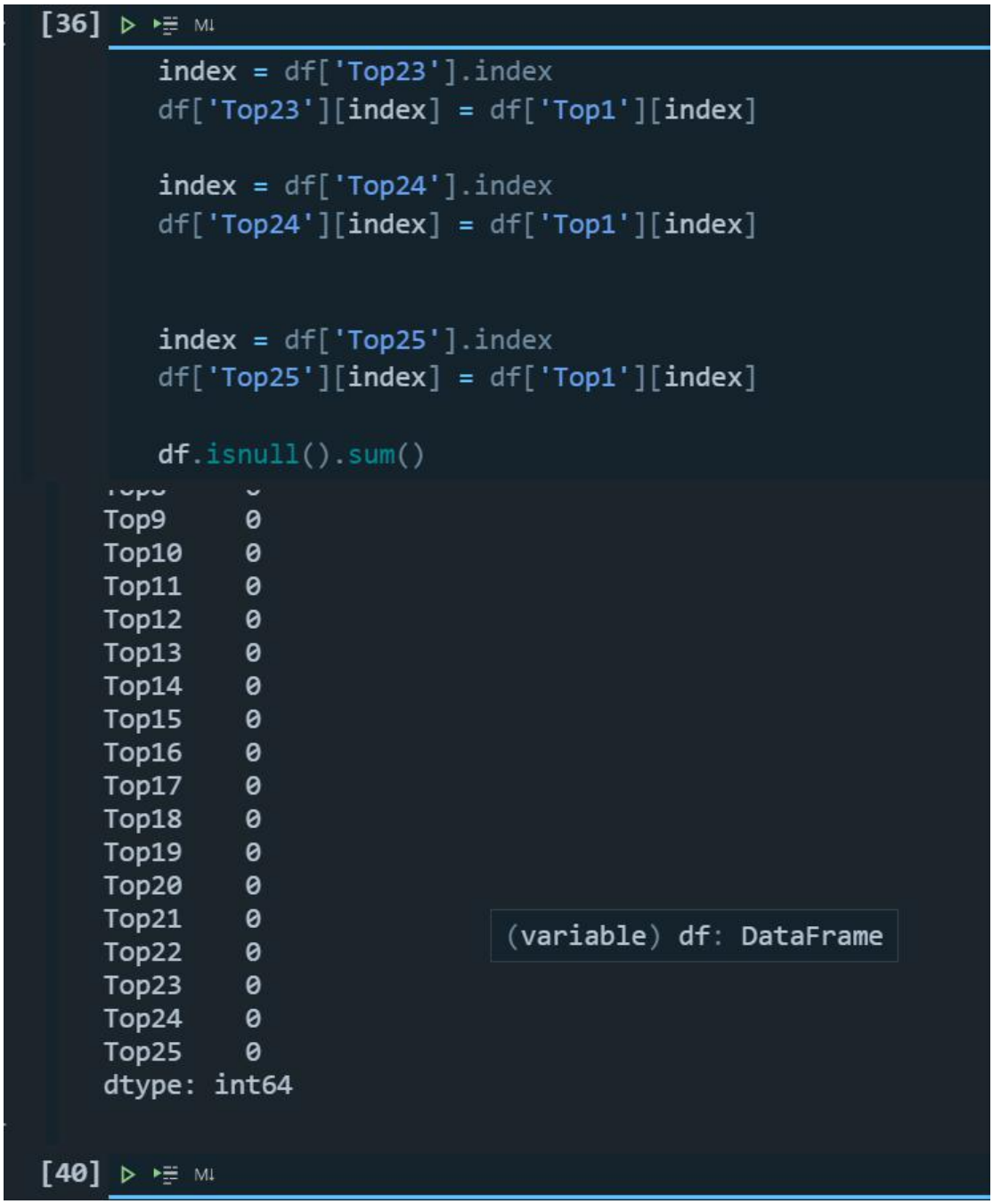
```
df=pd.read_csv('Data.csv', encoding = "ISO-8859-1")
```

```
stop_words = set({})
with open('stopwords','r') as f:
    for line in f:
        stop_words.add(line.strip())
df.isnull().sum()
```

```
Date     0
Label    0
Top1     0
Top2     0
Top3     0
Top4     0
Top5     0
Top6     0
Top7     0
Top8     0
Top9     0
Top10    0
Top11    0
Top12    0
Top13    0
Top14    0
Top15    0
Top16    0
Top17    0
Top18    0
Top19    0
Top20    0
Top21    0
Top22    0
Top23    1
Top24    3
Top25    3
dtype: int64
```

From the result,we inspect that top 23,top 24,top25 has missing values

2. Replace missing values



All missing values are replaced with the content of Top1

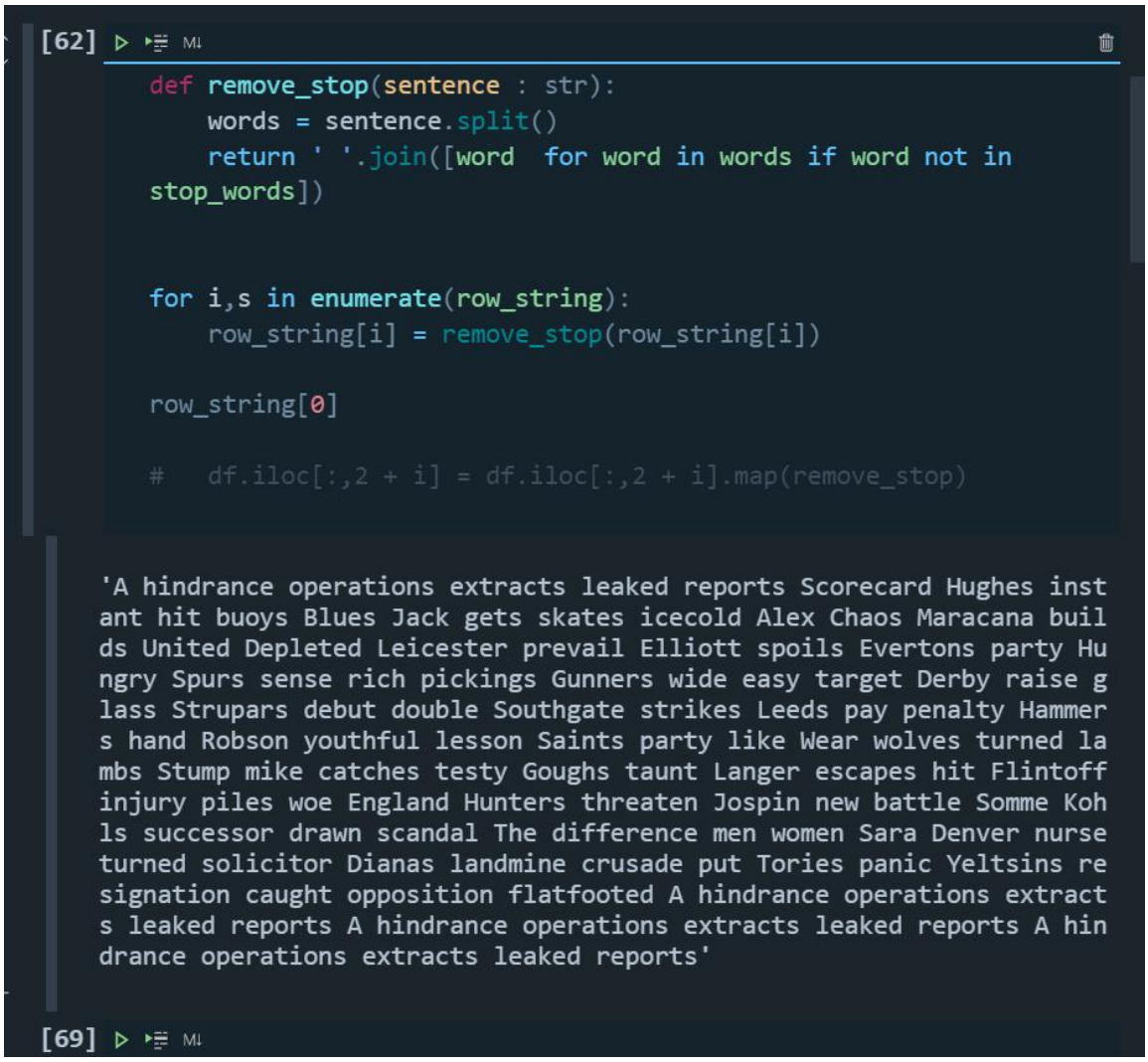
3. Remove punctuation with regex and convert all words into lower case

4. [67]
```
import re
re_obj = re.compile(r'[^a-zA-Z ]')
row_string = []


for i in range(len(df.index)):
    row_string.append(' '.join(top for top in df.iloc[i,2:]))
for i,_ in enumerate(row_string):
    # row_string[i].replace(r'[^a-zA-Z]','',regex = True,
inplace=True)
    row_string[i] =  re_obj.sub('' lower: Any i])
    row_string[i] = row_string[i].lower()
row_string[0]
```

```
'a hindrance operations extracts leaked reports scorecard hughes inst
ant hit buoys blues jack gets skates icecold alex chaos maracana buil
ds united depleted leicester prevail elliott spoils evertons party hu
ngry spurs sense rich pickings gunners wide easy target derby raise g
lass strupars debut double southgate strikes leeds pay penalty hammer
s hand robson youthful lesson saints party like wear wolves turned la
mbs stump mike catches testy goughs taunt langer escapes hit flintoff
injury piles woe england hunters threaten jospin new battle somme koh
ls successor drawn scandal the difference men women sara denver nurse
turned solicitor dianas landmine crusade put tories panic yeltsins re
signation caught opposition flatfooted a hindrance operations extract
s leaked reports a hindrance operations extracts leaked reports a hin
drance operations extracts leaked reports'
```

[62]

5. Remove stop words from a pre collected set of words.



```
[62]
def remove_stop(sentence : str):
    words = sentence.split()
    return ' '.join([word  for word in words if word not in
stop_words])


for i,s in enumerate(row_string):
    row_string[i] = remove_stop(row_string[i])

row_string[0]

#   df.iloc[:,2 + i] = df.iloc[:,2 + i].map(remove_stop)
```

```
'A hindrance operations extracts leaked reports Scorecard Hughes inst
ant hit buoys Blues Jack gets skates icecold Alex Chaos Maracana buil
ds United Depleted Leicester prevail Elliott spoils Evertons party Hu
ngry Spurs sense rich pickings Gunners wide easy target Derby raise g
lass Strupars debut double Southgate strikes Leeds pay penalty Hammer
s hand Robson youthful lesson Saints party like Wear wolves turned la
mbs Stump mike catches testy Goughs taunt Langer escapes hit Flintoff
injury piles woe England Hunters threaten Jospin new battle Somme Koh
ls successor drawn scandal The difference men women Sara Denver nurse
turned solicitor Dianas landmine crusade put Tories panic Yeltsins re
signation caught opposition flatfooted A hindrance operations extract
s leaked reports A hindrance operations extracts leaked reports A hin
drance operations extracts leaked reports'
```

```
[69]
```

6. Replace data frame content with a single Series of processed string.

[69]

```
df['Top1'] = pd.Series(row_string)
df = df[['Date','Label','Top1']]
df.columns = ['Date','Label','Content']
df.head()
```

	Date	Label	Content
0	2000-01-03	0	a hindrance operations extracts leaked reports...
1	2000-01-04	0	scorecard the best lake scene leader german sl...
2	2000-01-05	0	coventry caught counter flo uniteds rivals roa...
3	2000-01-06	1	pilgrim knows progress thatcher facing ban mci...
4	2000-01-07	1	hitches horlocks beckham united survive breast...

[70]

For now,all the preparation work has been done,next feed these data into models and train them,

8: convert string into bag of words

[76]

```
## implement BAG OF WORDS
countvector=CountVectorizer(ngram_range=(2,2))
traindataset=countvector.fit_transform(headlines)
```

9. Random Forest model:

```
# implement RandomForest Classifier
randomclassifier=RandomForestClassifier(n_estimators=200,
criterion='entropy')
randomclassifier.fit(traindataset,train['Label'])
```

```
RandomForestClassifier(criterion='entropy', n_estimators=200)
```

[78]

```
## Predict for the Test Dataset
test_transform= test['Content']
test_dataset = countvector.transform(test_transform)
predictions = randomclassifier.predict(test_dataset)
```

[81]

```
## Import library to check accuracy
from sklearn.metrics import classification_report,
confusion_matrix,accuracy_score
```

[83]

```
matrix=confusion_matrix(test['Label'],predictions)
print(matrix)
score=accuracy_score(test['Label'],predictions)
print(score)
report=classification_report(test['Label'],predictions)
print(report)
```

```
[[131  55]
 [  2 190]]
0.8492063492063492
              precision    recall  f1-score   support

           0       0.98      0.70      0.82       186
           1       0.78      0.99      0.87       192

    accuracy                           0.85       378
   macro avg       0.88      0.85      0.85       378
weighted avg       0.88      0.85      0.85       378
```

Accuracy : 0.849

10 : logic regression :

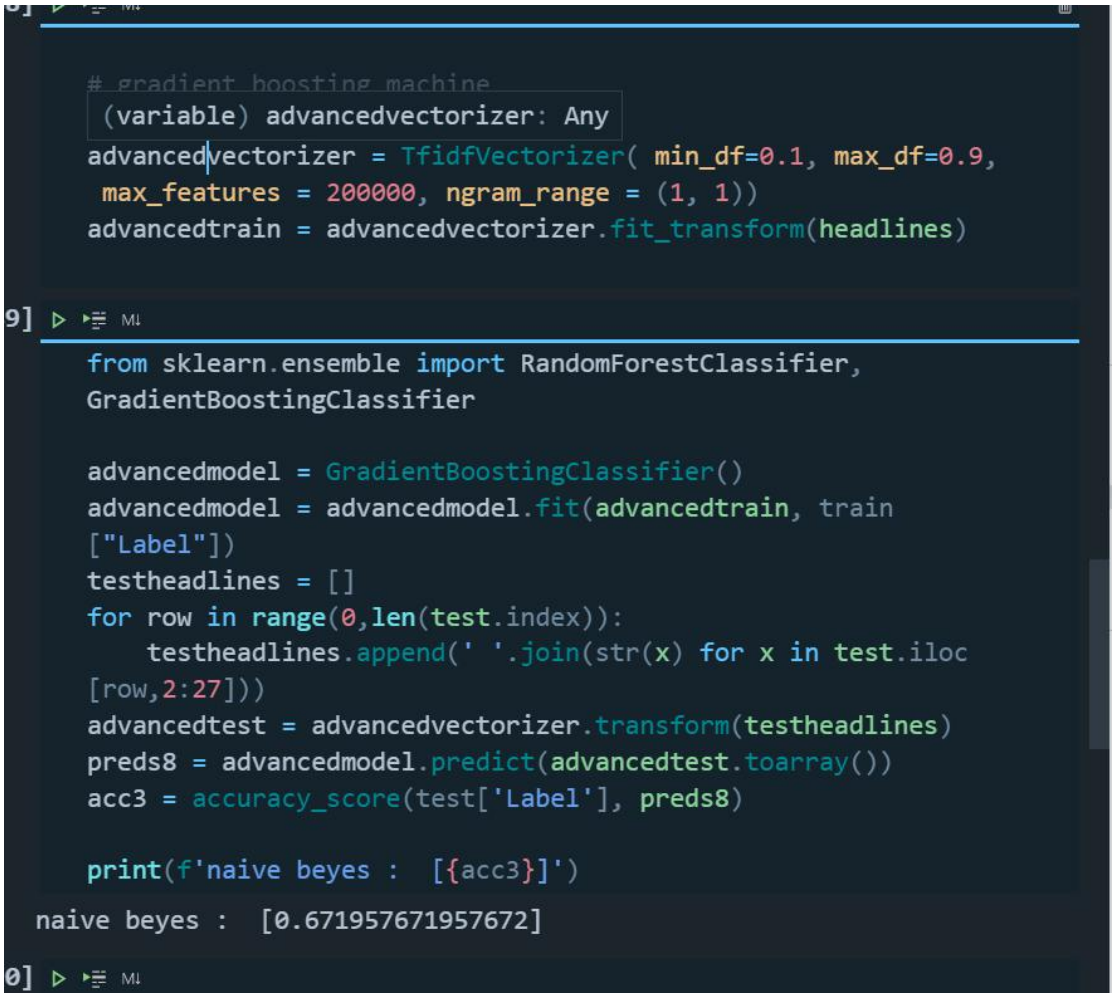
```
[84]
from sklearn.linear_model import SGDClassifier, SGDRegressor,
LogisticRegression

basicmodel = LogisticRegression()
basicmodel = basicmodel.fit(traindataset, train["Label"])

[85]
preds1 = basicmodel.predict(test_dataset)

acc1=accuracy_score(test['Label'], preds1)
print(f'logic regression [{acc1}]')

logic regression [0.8333333333333334]

[86]
```

Accuracy : 0.8333

11 : Naive Bayes

```
# * naive beyes


from sklearn.feature_extraction.text import TfidfVectorizer,
CountVectorizer
advancedvectorizer = TfidfVectorizer( min_df=0.1, max_df=0.7,
 max_features = 200000, ngram_range = (1, 1))
advancedtrain = advancedvectorizer.fit_transform(headlines)


[87]
from sklearn.naive_bayes import MultinomialNB

advancedmodel = MultinomialNB(alpha=0.01)
advancedmodel = advancedmodel.fit(advancedtrain, train
["Label"])
testheadlines = []
for row in range(0,len(test.index)):
    testheadlines.append(' '.join(str(x) for x in test.iloc
[row,2:27]))
advancedtest = advancedvectorizer.transform(testheadlines)
preds4 = advancedmodel.predict(advancedtest)
acc2=accuracy_score(test['Label'], preds4)
print(f'naive beyes :  [{acc2}]')

naive beyes :  [0.5185185185185185]

[88]
```

12 : Gradient Boosting Machine



```
# gradient boosting machine
(variable) advancedvectorizer: Any
advancedvectorizer = TfidfVectorizer( min_df=0.1, max_df=0.9,
 max_features = 200000, ngram_range = (1, 1))
advancedtrain = advancedvectorizer.fit_transform(headlines)
```

```
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier

advancedmodel = GradientBoostingClassifier()
advancedmodel = advancedmodel.fit(advancedtrain, train
["Label"])
testheadlines = []
for row in range(0,len(test.index)):
    testheadlines.append(' '.join(str(x) for x in test.iloc
[row,2:27]))
advancedtest = advancedvectorizer.transform(testheadlines)
preds8 = advancedmodel.predict(advancedtest.toarray())
acc3 = accuracy_score(test['Label'], preds8)

print(f'naive beyes :  [{acc3}]')
```

```
naive beyes :  [0.671957671957672]
```

13: Stochastic Gradient descent

```
# stochastic gradient descent


from sklearn.linear_model import SGDClassifier,
SGDRegressor,LogisticRegression

advancedmodel = SGDClassifier(loss='modified_huber',
random_state=0, shuffle=True)
advancedmodel = advancedmodel.fit(advancedtrain, train
["Label"])
testheadlines = []
for row in range(0,len(test.index)):
    testheadlines.append(' '.join(str(x) for x in test.iloc
[row,2:27]))
advancedtest = advancedvectorizer.transform(testheadlines)
preds10 = advancedmodel.predict(advancedtest.toarray())
acc4 = accuracy_score(test['Label'], preds10)


print(f'SGD :  :  [{acc4}]')
```

```
SGD :  :  [0.5476190476190477]
```

14 : KNN

In summary :

Model	Accuracy
Random Forest	0.849
logic regression	0.833
Naive Bayes	0.518
Gradient Boosting Machine	0.672
Stochastic Gradient descent	0.547
KNN	0.629

Conclusion : From the above results ,we could infer that Random Forest achieves the best accuracy,thought it is a simple algorithm,and Naive Bayes gain the worst result .

This is not the case that the more complicated an algorithm is ,the better it is , the results really depends on the data set and read-world situations

Bibliography :

1. Neuro–Linguistic Programming (goodtherapy.org)
2. Natural language processing - Wikipedia
3. Stock Price Movement Based On News Headline - Analytics Vidhya
4. Stock-Prediction from News — A Naive Approach | by Andreas Stöckl | Medium
5. Top 5 Pre-Trained NLP Language Models (daffodilsw.com)
6. Learn Python - Free Interactive Python Tutorial