Sentiment_Analysis

September 22, 2023

[1]:
```
import numpy as np
import pandas as pd

from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
from sklearn import metrics

import string
import spacy

from sklearn.model_selection import train_test_split

np.random.seed(42)
```

[2]:
```
finance = pd.read_csv("./drive/MyDrive/Dataset/RT/data.csv")
finance.head()
```

[2]:
```
                                            Sentence Sentiment
0  The GeoSolutions technology will leverage Bene…  positive
1  $ESI on lows, down $1.50 to $2.50 BK a real po…  negative
2  For the last quarter of 2010 , Componenta 's n…  positive
3  According to the Finnish-Russian Chamber of Co…   neutral
4  The Swedish buyout firm has sold its remaining…   neutral
```

[3]:
```
labels = finance["Sentiment"].unique()
labels
```

[3]:
```
array(['positive', 'negative', 'neutral'], dtype=object)
```

[4]:
```
# This function is used to convert to labels so the computer can understand them
def convert_labels_nums(label):
  if label == "positive":
    return 1
  elif label == "negative":
    return -1
  else:
    return 0
```

[5]:
```
finance["Sentiment"] = finance["Sentiment"].apply(convert_labels_nums)
```

[6]:
```
finance.head()
```

[6]:
```
                                            Sentence  Sentiment
0  The GeoSolutions technology will leverage Bene…          1
1  $ESI on lows, down $1.50 to $2.50 BK a real po…         -1
2  For the last quarter of 2010 , Componenta 's n…          1
3  According to the Finnish-Russian Chamber of Co…          0
4  The Swedish buyout firm has sold its remaining…          0
```

[7]:
```
# Text PreProcessing
nlp = spacy.load("en_core_web_sm")
stop_words = nlp.Defaults.stop_words
print(stop_words)
```

```
{'had', 'nowhere', 'nevertheless', 'themselves', 'yourselves', 'with', 'latter',
'they', 'itself', 'else', 'but', 'an', 'do', 'within', 'almost', 'get', 'its',
'may', '‘ve', 'one', 'several', 'our', "'s", 'might', 'already', 'forty',
'bottom', 'between', 'fifteen', 'noone', 'sometime', '‘m', 'n‘t', 'never',
'everyone', 'becomes', 'which', 'beside', 'wherein', 'thereby', 'if', "'m",
'by', 'my', 'alone', "'re", 'after', 'therefore', 'up', 'go', '’m', 'down',
'therein', 'against', 'why', 'then', 'amongst', 'using', '’ll', 'perhaps',
'beyond', 'done', 'much', 'unless', 'of', 'cannot', 'ca', 'side', 'as', 'n’t',
'made', 'did', 'due', 'six', 'too', 'although', 'very', 'or', 'full', 'could',
'often', 'mostly', 'when', 'ours', 'together', 'nine', 'neither', 'upon',
'amount', 'latterly', 'nobody', 'am', 'rather', 'top', 'until', 'ourselves',
'meanwhile', 'two', 'everywhere', 'same', 'been', 'somewhere', 'doing', 'him',
'on', 'also', 'under', 'a', 'will', 'hereby', 'put', 'so', 'toward', 'quite',
'few', 'there', 'only', 'next', 'i', 'how', 'both', 'further', 'me', 'anything',
'before', 'because', 'part', 'less', 'twelve', 'whence', 'be', 'see', 'in',
'‘d', 'into', 'seem', 'should', 'some', 'now', 'elsewhere', 'does', 'whereupon',
'own', 'more', 'herself', 'none', 'twenty', 'nor', 'nothing', 'we', 'really',
'behind', 'whatever', 'who', 'us', 'other', 'others', 'least', 'name', 'across',
"'ve", 'always', 'another', 'here', 'somehow', 'hundred', 'them', 're',
'either', 'back', 'out', '‘re', 'just', 'seems', 'indeed', 'his', 'yet',
'hereupon', 'moreover', 'call', 'thereafter', 'these', 'fifty', 'thence',
'eleven', 'no', 'throughout', 'those', 'sometimes', 'someone', 'yours',
'thereupon', 'seemed', 'without', 'whither', 'something', 'each', 'anyone',
'ever', 'still', 'your', 'while', 'whole', 'it', 'namely', '’d', 'anyhow',
'through', 'onto', 'such', 'since', 'well', 'show', 'you', 'myself', 'become',
'third', 'whose', 'per', "n't", 'is', 'three', 'hers', 'she', 'were', 'five',
'from', 'the', 'every', 'keep', 'was', 'he', 'first', 'sixty', 'thru', 'anyway',
'hence', 'everything', 'whenever', 'himself', 'at', 'whereafter', 'used', 'and',
'regarding', '‘s', 'can', 'last', 'give', 'during', 'to', 'wherever', 'whereby',
'many', 'front', 'however', 'towards', 'where', 'various', 'thus', "'d",
'serious', 'are', 'being', '’s', 'about', 'most', 'what', 'enough', 'via',
'though', 'have', '‘ll', 'whereas', 'not', 'again', 'among', 'along', '’re',
'for', 'say', 'even', 'make', 'once', 'herein', 'except', 'please', 'off',
'take', 'any', 'seeming', 'below', 'must', 'her', 'formerly', 'above', 'that',
'eight', 'otherwise', 'ten', 'has', 'mine', 'over', 'beforehand', 'yourself',
'afterwards', '’ve', 'whoever', 'four', 'empty', 'around', 'their', 'than',
'becoming', 'whom', 'became', 'besides', 'move', 'hereafter', 'former', "'ll",
'anywhere', 'all', 'whether', 'this', 'would'}
```

[8]:
```
punctuations = string.punctuation
print(punctuations)
```

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

[9]:
```
# Creating a Tokenizer Function

def spacy_tokenizer(text):
  # Creating a doc
  doc = nlp(text)

  # Lemmatizing each token
  mytokens = [word.lemma_.lower().strip() for word in doc ]

  # Remove stop words and punctuations
  mytokens = [ word for word in mytokens if word not in stop_words and word not␣
 ↪in punctuations ]

  return mytokens
```

1 Count Vectorizer

[10]:
```
count_vector = CountVectorizer(tokenizer = spacy_tokenizer)
```

[11]:
```
# Splitting the data
X = finance["Sentence"] # features
y = finance["Sentiment"] # labels
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪stratify = y)
```

[12]:
```
X_train_vectors = count_vector.fit_transform(X_train)
X_test_vectors = count_vector.transform(X_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528:
UserWarning: The parameter 'token_pattern' will not be used since 'tokenizer' is
not None'
  warnings.warn(
```

[13]:
```
X_train_vectors.shape, X_test_vectors.shape
```

[13]:
```
((4673, 9404), (1169, 9404))
```

2 TF-IDF Vectorizer

[28]:
```
tfidf_vector = TfidfVectorizer(tokenizer = spacy_tokenizer)
```

[29]:
```
X_train_vectors2 = tfidf_vector.fit_transform(X_train)
X_test_vectors2 = tfidf_vector.transform(X_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528:
UserWarning: The parameter 'token_pattern' will not be used since 'tokenizer' is
not None'
  warnings.warn(
```

[30]:
```
X_train_vectors2.shape, X_test_vectors2.shape
```

[30]:
```
((4673, 9404), (1169, 9404))
```

3 Model Building

Different multiclass classification methods were used, such as: - Linear SVM - Naive Bayes Classifier

[16]:
```
# Naive Bayes Classifier
from sklearn.naive_bayes import MultinomialNB

clf = MultinomialNB()

clf.fit(X_train_vectors,y_train)
```

[16]:
```
MultinomialNB()
```

[26]:
```
def clf_metrics_score(y_true,y_pred,param="weighted"):
  # 'micro', 'macro', 'weighted'
  print("Accuracy:",metrics.accuracy_score(y_true,y_pred))
  print("Precision:",metrics.precision_score(y_true,y_pred,average=param))
  print("Recall:",metrics.recall_score(y_true,y_pred,average=param))
```

[27]:
```
pred = clf.predict(X_test_vectors)
clf_metrics_score(y_test,pred)
```

```
Accuracy: 0.6757912745936698
Precision: 0.6650905636654316
Recall: 0.6757912745936698
```

[31]:
```
clf2 = MultinomialNB()
clf2.fit(X_train_vectors2,y_train)
```

[31]:
```
MultinomialNB()
```

[32]:
```
pred2 = clf2.predict(X_test_vectors2)
clf_metrics_score(y_test,pred2)
```

```
Accuracy: 0.6449957228400343
Precision: 0.6466019082279065
Recall: 0.6449957228400343
```

[39]:
```
from sklearn import svm
lin_clf = svm.LinearSVC(loss="hinge",dual=True,multi_class="crammer_singer")
lin_clf.fit(X_train_vectors,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
```

[39]:
```
LinearSVC(loss='hinge', multi_class='crammer_singer')
```

[40]:
```
pred3 = lin_clf.predict(X_test_vectors)
clf_metrics_score(y_test,pred3)
```

```
Accuracy: 0.6313088109495295
Precision: 0.6186847144993874
Recall: 0.6313088109495295
```

[41]:
```
lin_clf.fit(X_train_vectors2,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
```

[41]:
```
LinearSVC(loss='hinge', multi_class='crammer_singer')
```

[42]:
```
pred4 = lin_clf.predict(X_test_vectors2)
clf_metrics_score(y_test,pred4)
```

```
Accuracy: 0.6458511548331908
Precision: 0.6291786746944827
Recall: 0.6458511548331908
```

[ ]:
```

```