21bai1217-sentiment-analysis

June 17, 2023

[ ]:
```
import re
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
import math
import nltk
from sklearn.feature_extraction.text import CountVectorizer
from collections import defaultdict
nltk.download('wordnet')
data = pd.read_csv('IMDB Dataset.csv')
data
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
```

[ ]:

	review	sentiment
0	One of the other reviewers has mentioned that …	positive
1	A wonderful little production. The…	positive
2	I thought this was a wonderful way to spend ti…	positive
3	Basically there's a family where a little boy …	negative
4	Petter Mattei's "Love in the Time of Money" is…	positive
…	…	…
49995	I thought this movie did a down right good job…	positive
49996	Bad plot, bad dialogue, bad acting, idiotic di…	negative
49997	I am a Catholic taught in parochial elementary…	negative
49998	I'm going to have to disagree with the previou…	negative
49999	No one expects the Star Trek movies to be high…	negative

[50000 rows x 2 columns]

[ ]:
```
data
```

[ ]:

	review	sentiment
0	One of the other reviewers has mentioned that …	positive
1	A wonderful little production. The…	positive

```
2      I thought this was a wonderful way to spend ti…  positive
3      Basically there's a family where a little boy …  negative
4      Petter Mattei's "Love in the Time of Money" is…  positive
…                                                    …         …
49995  I thought this movie did a down right good job…  positive
49996  Bad plot, bad dialogue, bad acting, idiotic di…  negative
49997  I am a Catholic taught in parochial elementary…  negative
49998  I'm going to have to disagree with the previou…  negative
49999  No one expects the Star Trek movies to be high…  negative

[50000 rows x 2 columns]
```

[ ]:
```
import re
import nltk
from nltk.corpus import stopwords

def remove_tags(string):
    removelist = ""
    result = re.sub('<.*?>', '', string)  # remove HTML tags
    result = re.sub('https://.*', '', result)  # remove URLs
    result = re.sub(r'[^a-zA-Z0-9' + removelist + ']', ' ', result)  # remove␣
 ↪non-alphanumeric characters
    result = result.lower()
    return result

data['review'] = data['review'].apply(lambda cw: remove_tags(cw))

nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
data['review'] = data['review'].apply(lambda x: ' '.join([word for word in x.
 ↪split() if word not in stop_words]))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[ ]:
```
data
```

[ ]:
```
                                                  review sentiment
0      one reviewers mentioned watching 1 oz episode …  positive
1      wonderful little production filming technique …  positive
2      thought wonderful way spend time hot summer we…  positive
3      basically family little boy jake thinks zombie…  negative
4      petter mattei love time money visually stunnin…  positive
…                                                    …         …
49995  thought movie right good job creative original…  positive
49996  bad plot bad dialogue bad acting idiotic direc…  negative
49997  catholic taught parochial elementary schools n…  negative
```

```
49998  going disagree previous comment side maltin on…  negative
49999  one expects star trek movies high art fans exp…  negative

[50000 rows x 2 columns]
```

[ ]:
```
import nltk
from nltk.stem import WordNetLemmatizer
w_tokenizer = nltk.tokenize.WhitespaceTokenizer()
nltk.download('wordnet')

def lemmatize_text(text):
    lemmatizer = WordNetLemmatizer()
    tokens = nltk.word_tokenize(text)
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens]
    lemmatized_text = ' '.join(lemmatized_tokens)
    data['review'] = data.review.apply(lemmatized_text)
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
```

[ ]:
```
data
```

[ ]:
```
                                                  review sentiment
0      one reviewers mentioned watching 1 oz episode …  positive
1      wonderful little production filming technique …  positive
2      thought wonderful way spend time hot summer we…  positive
3      basically family little boy jake thinks zombie…  negative
4      petter mattei love time money visually stunnin…  positive
…                                                    …         …
49995  thought movie right good job creative original…  positive
49996  bad plot bad dialogue bad acting idiotic direc…  negative
49997  catholic taught parochial elementary schools n…  negative
49998  going disagree previous comment side maltin on…  negative
49999  one expects star trek movies high art fans exp…  negative

[50000 rows x 2 columns]
```

[ ]:
```
reviews = data['review'].values
labels = data['sentiment'].values
encoder = LabelEncoder()
encoded_labels = encoder.fit_transform(labels)
```

[ ]:
```
train_sentences, test_sentences, train_labels, test_labels =␣
 ↪train_test_split(reviews, encoded_labels, stratify = encoded_labels)
```

[ ]:
```
vec = CountVectorizer(max_features = 3000)
X = vec.fit_transform(train_sentences)
```

```
vocab = vec.get_feature_names_out()
X = X.toarray()
word_counts = {}
for l in range(2):
    word_counts[l] = defaultdict(lambda: 0)
for i in range(X.shape[0]):
    l = train_labels[i]
    for j in range(len(vocab)):
        word_counts[l][vocab[j]] += X[i][j]
```

[ ]:
```
def laplace_smoothing(n_label_items, vocab, word_counts, word, text_label):
    a = word_counts[text_label][word] + 1
    b = n_label_items[text_label] + len(vocab)
    return math.log(a/b)
```

[ ]:
```
def group_by_label(x, y, labels):
    data = {}
    for l in labels:
        data[l] = x[np.where(y == l)]
    return data
```

[ ]:
```
def fit(x, y, labels):
    n_label_items = {}
    log_label_priors = {}
    n = len(x)
    grouped_data = group_by_label(x, y, labels)
    for l, data in grouped_data.items():
        n_label_items[l] = len(data)
        log_label_priors[l] = math.log(n_label_items[l] / n)
    return n_label_items, log_label_priors
```

[ ]:
```
def predict(n_label_items, vocab, word_counts, log_label_priors, labels, x):
    result = []
    for text in x:
        label_scores = {l: log_label_priors[l] for l in labels}
        words = set(w_tokenizer.tokenize(text))
        for word in words:
            if word not in vocab: continue
            for l in labels:
                log_w_given_l = laplace_smoothing(n_label_items, vocab,␣
 ↪word_counts, word, l)
                label_scores[l] += log_w_given_l
        result.append(max(label_scores, key=label_scores.get))
    return result
```

[ ]:
```
labels = [0,1]
n_label_items, log_label_priors = fit(train_sentences,train_labels,labels)
```

```
pred = predict(n_label_items, vocab, word_counts, log_label_priors, labels,␣
  ↪test_sentences)
print("Accuracy of prediction on test set : ", accuracy_score(test_labels,pred))
```

```
Accuracy of prediction on test set :  0.854
```