amazon-reviews

March 18, 2021

1 Class Data

[1]:
```
import random

class Sentiment:
    NEGATIVE = "NEGATIVE"
    NEUTRAL = "NEUTRAL"
    POSITIVE = "POSITIVE"

class Review:
    def __init__(self, text, score):
        self.text = text
        self.score = score
        self.sentiment = self.get_sentiment()

    def get_sentiment(self):
        if self.score <= 2:
            return Sentiment.NEGATIVE
        elif self.score == 3:
            return Sentiment.NEUTRAL
        else: #Score 4 or 5
            return Sentiment.POSITIVE

class ReviewContainer:
    def __init__(self, reviews):
        self.reviews = reviews

    def get_text(self):
        return [x.text for x in self.reviews]

    def get_sentiment(self):
        return [x.sentiment for x in self.reviews]

    def evenly_distribute(self):
        negative = list(filter(lambda x: x.sentiment == Sentiment.NEGATIVE,␣
 ↪self.reviews))
```

```
        positive = list(filter(lambda x: x.sentiment == Sentiment.POSITIVE,␣
↪self.reviews))
        positive_shrunk = positive[:len(negative)]
        self.reviews = negative + positive_shrunk
        random.shuffle(self.reviews)
```

2 Load Data

[2]:
```
import json

file_name = 'Books_small_10000.json'
reviews = []
with open(file_name) as f:
    for line in f:
        review = json.loads(line)
        reviews.append(Review(review['reviewText'], review['overall']))

reviews[5].text
```

[2]:
```
'I hoped for Mia to have some peace in this book, but her story is so real and
raw.  Broken World was so touching and emotional because you go from Mia\'s
trauma to her trying to cope.  I love the way the story displays how there is no
"just bouncing back" from being sexually assaulted.  Mia showed us how those
demons come for you every day and how sometimes they best you. I was so in the
moment with Broken World and hurt with Mia because she was surrounded by people
but so alone and I understood her feelings.  I found myself wishing I could give
her some of my courage and strength or even just to be there for her.  Thank you
Lizzy for putting a great character\'s voice on a strong subject and making it
so that other peoples story may be heard through Mia\'s.'
```

3 Prep Data

[3]:
```
from sklearn.model_selection import train_test_split

training, test = train_test_split(reviews, test_size=0.33, random_state=42)

train_container = ReviewContainer(training)
train_container.evenly_distribute()
len(train_container.reviews)

test_container = ReviewContainer(test)
test_container.evenly_distribute()
```

[4]:
```
train_x = train_container.get_text()
train_y = train_container.get_sentiment()
```

```
test_x = test_container.get_text()
test_y = test_container.get_sentiment()
```

4 Bag of words vectorization

[5]:
```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
vectorizer = TfidfVectorizer()
train_x_vectors = vectorizer.fit_transform(train_x)

#print(train_x[0])
#print(train_x_vectors[0])

#train_x_vectors
#train_y
```

5 Classification

Linear SVM

[6]:
```
from sklearn.svm import SVC

clf_svm = SVC(kernel='linear')
clf_svm.fit(train_x_vectors, train_y)

test_x_vectors = vectorizer.transform(test_x)
clf_svm.predict(test_x_vectors[0])
```

[6]:
```
array(['NEGATIVE'], dtype='<U8')
```

Decision Tree

[7]:
```
from sklearn.tree import DecisionTreeClassifier

clf_dec = DecisionTreeClassifier()
clf_dec.fit(train_x_vectors, train_y)

clf_dec.predict(test_x_vectors[0])
```

[7]:
```
array(['NEGATIVE'], dtype='<U8')
```

Naive Bayes

[8]:
```
from sklearn.naive_bayes import GaussianNB

clf_gnb = DecisionTreeClassifier()
clf_gnb.fit(train_x_vectors, train_y)
```

```
clf_gnb.predict(test_x_vectors[0])
```

[8]:
```
array(['NEGATIVE'], dtype='<U8')
```

Logistic Regression

[9]:
```
from sklearn.linear_model import LogisticRegression

clf_log = DecisionTreeClassifier()
clf_log.fit(train_x_vectors, train_y)

clf_log.predict(test_x_vectors[0])
```

[9]:
```
array(['NEGATIVE'], dtype='<U8')
```

6 Evaluation

[10]:
```
clf_svm.score(test_x_vectors, test_y)
```

[10]:
```
0.8076923076923077
```

[11]:
```
clf_dec.score(test_x_vectors, test_y)
```

[11]:
```
0.6298076923076923
```

[12]:
```
clf_gnb.score(test_x_vectors, test_y)
```

[12]:
```
0.6538461538461539
```

[13]:
```
clf_log.score(test_x_vectors, test_y)
```

[13]:
```
0.6490384615384616
```

F1 Scores

[14]:
```
# F1 Scores
from sklearn.metrics import f1_score

f1_score(test_y, clf_svm.predict(test_x_vectors), average=None,␣
 ↪labels=[Sentiment.POSITIVE, Sentiment.NEUTRAL, Sentiment.NEGATIVE])
```

```
/opt/anaconda3/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1464: UndefinedMetricWarning:
F-score is ill-defined and being set to 0.0 in labels with no true nor predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(
```

[14]:
```
array([0.80582524, 0.        , 0.80952381])
```

[15]:
```
f1_score(test_y, clf_dec.predict(test_x_vectors), average=None,␣
↪labels=[Sentiment.POSITIVE, Sentiment.NEUTRAL, Sentiment.NEGATIVE])
```

[15]:
```
array([0.61691542, 0.        , 0.64186047])
```

[16]:
```
f1_score(test_y, clf_gnb.predict(test_x_vectors), average=None,␣
↪labels=[Sentiment.POSITIVE, Sentiment.NEUTRAL, Sentiment.NEGATIVE])
```

[16]:
```
array([0.65714286, 0.        , 0.65048544])
```

[17]:
```
f1_score(test_y, clf_log.predict(test_x_vectors), average=None,␣
↪labels=[Sentiment.POSITIVE, Sentiment.NEUTRAL, Sentiment.NEGATIVE])
```

[17]:
```
array([0.64903846, 0.        , 0.64903846])
```

[18]:
```
train_y.count(Sentiment.POSITIVE)
```

[18]:
```
436
```

[19]:
```
test_y.count(Sentiment.POSITIVE)
```

[19]:
```
208
```

[20]:
```
test_set = ['I thoroughly enjoed this, 5 stars', 'bad book do not buy',␣
↪'horrible waste of time']
new_test = vectorizer.transform(test_set)

clf_svm.predict(new_test)
```

[20]:
```
array(['POSITIVE', 'NEGATIVE', 'NEGATIVE'], dtype='<U8')
```

7 Tuning our model (with grid Search)

[21]:
```
from sklearn.model_selection import GridSearchCV

parameters = {'kernel': ('linear', 'rbf'), 'C': (1,4,8,16,32)}
svc = SVC()
clf = GridSearchCV(svc, parameters, cv=5)
clf.fit(train_x_vectors, train_y)
```

[21]:
```
GridSearchCV(cv=5, estimator=SVC(),
             param_grid={'C': (1, 4, 8, 16, 32), 'kernel': ('linear', 'rbf')})
```

8 Saving Model

[22]:
```
import pickle

with open('./sentiment_classifier.pkl', 'wb') as f:
    pickle.dump(clf, f)
```

8.1 Load model

[23]:
```
with open('./sentiment_classifier.pkl', 'rb') as f:
    loaded_clf = pickle.load(f)
```

[25]:
```
test_x[0]

loaded_clf.predict(test_x_vectors[0])
```

[25]:
```
array(['NEGATIVE'], dtype='<U8')
```

[ ]:
```

```