Week 10 Quiz

Perrin Anto - paj2117

Due Sat. Nov. 23, 11:59pm

In this quiz, we're going to load documents from 2 topics (space, cars) in the 20newsgroups dataset.

The goal is to train a classifier that classifies documents into these 2 topics based on a term frequency representation of the documents.

We will then calculate mean cross-validaion accuracy of a RandomForestClassifier using this transformation.

Setup Environment

In [24]:
```
import numpy as np
import pandas as pd
```

Load the Dataset

In [9]:
```
# Import fetch_20newsgroups from sklearn.datasets
from sklearn.datasets import fetch_20newsgroups

# Load the dataset into newsgroups using fetch_20newsgroups.
#   Only fetch the training subset using subset='train'.
#   Only fetch the two topics using categories=['sci.space','rec.autos']
# Store in the result into newsgroups
newsgroups = fetch_20newsgroups(subset='train',categories=['sci.space',
'rec.autos'])

# Store the newsgroups.data as docs, newsgroups.target as y and newsgrou
ps.target_names as y_names
docs = newsgroups.data
y = newsgroups.target
y_names = newsgroups.target_names

# Print the number of observations by printing the length of docs
len(docs)
```

Out[9]:
```
1187
```

In [10]:
```
# Print the text of the first document in docs.
docs[0]
```

Out[10]:
```
"From: prb@access.digex.com (Pat)\nSubject: Re: Proton/Centaur?\nOrgani
zation: Express Access Online Communications USA\nLines: 15\nNNTP-Posti
ng-Host: access.digex.net\n\n\nWell thank you dennis for your as usual
highly detailed and informative \nposting.   \n\nThe question i have ab
out the proton, is  could it be  handled at\none of KSC's spare pads, w
ithout major  malfunction,  or could it be\nhandled at kourou  or Vande
nberg?   \n\nNow if it uses storables,  then  how long would it take fo
r the russians\nto equip something at cape york?\n\nIf  Proton were lau
nched from a western site,  how would it compare to the\nT4/centaur?
As i see it, it should lift  very close to the T4.\n\npat\n"
```

In [17]:
```
# Print the target value of the first document in y.
y[0]
```

Out[17]:
```
1
```

In [18]:
```
# Print the target_name of the first document using y_names and y.
y_names[y[0]]
```

Out[18]:
```
'sci.space'
```

Use CountVectorizer to Convert To TF

In [19]:
```
# Import CountVectorizer from sklearn.feature_extraction
from sklearn.feature_extraction.text import CountVectorizer

# Initialize a CountVectorizer object. It should
#   lowercase all text,
#   include both unigrams and bigrams
#   exclude terms that occur in fewer than 10 documents
#   exclude terms that occur in more that 95% of documents
#   exclude all 'english' stopwords
# Store as cvect
cvect = CountVectorizer(lowercase=True,
                        ngram_range=(1,2),
                        min_df=10,
                        max_df=.95,
                        stop_words='english')

# Fit cvect on docs and transform docs into their term frequency represe
ntation.
# Store as X_tf
X_tf = cvect.fit_transform(docs)

# Print the shape of X_tf.
# The number of rows should match the number of documents
#    and the number of columns should be in the thousands
X_tf.shape
```

Out[19]:
```
(1187, 3628)
```

In [22]:
```
# The stopwords learned by cvect are stored as a set in cvect.stop_words
_
# We'd like to print out a small subset of these terms.
# One way to get a subset of a set is to treat it as a list.
# First, convert the stop_words_ set to a list.
# Store as stop_words_list
stop_words_list = list(cvect.stop_words_)

# Print out the first 5 elements in stop_words_list.
# Note that, since a set is unordered,
#    there is no meaning to the ordering of these terms and they may var
y over runs.
stop_words_list[0:5]
```

Out[22]:
```
['lobby people', 'come familiar', '4101 email', '80 aslv', 'anybody con
cerned']
```

Calculate Mean CV Accuracy Using RandomForestClassifier

In [25]:
```
# Import cross_val_score from sklearn.model_selection
from sklearn.model_selection import cross_val_score

# Import RandomForestClassifier from sklearn.ensemble
from sklearn.ensemble import RandomForestClassifier

# Get a set of 5-fold CV scores using
#    a RandomForestClassifier with 50 trees, X_tf and y
# Store as cv_scores
cv_scores = cross_val_score(RandomForestClassifier(n_estimators=50),X_tf
,y,cv=5)

# Print the mean of these cv_scores. The mean accuracy should be above .
9
print(f'mean cv scores: {np.mean(cv_scores):0.2f}')
```

```
mean cv scores: 0.97
```

Optional: Find Important Features

In [ ]:
```
# CountVectorizer stores the feature names (terms in the vocabulary) in
 two ways:
#  1. as a dictionary of term:colum_index pairs, accessed via cvect.voca
bulary_
#  2. as a list of terms, in column index order, accessed via cvect.get_
feature_names()
#
# We can get the indices of the most important features by retraining a
 RandomForestClassifier on X,y
#  and accessing .feature_importances_
#
# Using some combination of the above data-structures,
#  print out the top 10 terms in the vocabulary
#  ranked by the feature importances learned by a RandomForestClassifier
#
# The terms you find will likely not be surprising given the document ca
tegories.
```