Week 10 Quiz

Bryan Gibson - brg2130

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

Load the Dataset

In [2]:
```
# 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 newsgroups.target_names as y_names
docs = newsgroups.data
y_names = newsgroups.target_names
y = newsgroups.target

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

Out[2]: 1187

In [3]:
```
# Print the text of the first document in docs.
docs[0]
```

Out[3]:
```
"From: prb@access.digex.com (Pat)\nSubject: Re: Proton/Centaur?\nOrga
nization: Express Access Online Communications USA\nLines: 15\nNNTP-P
osting-Host: access.digex.net\n\n\nWell thank you dennis for your as
usual highly detailed and informative \nposting.   \n\nThe question i
have about the proton, is  could it be  handled at\none of KSC's spar
e pads, without major  malfunction,  or could it be\nhandled at kouro
u  or Vandenberg?   \n\nNow if it uses storables,  then  how long wou
ld it take for the russians\nto equip something at cape york?\n\nIf
Proton were launched from a western site,  how would it compare to th
e\nT4/centaur?   As i see it, it should lift  very close to the T4.\n
\npat\n"
```

In [4]:
```
# Print the target value of the first document in y.
y[0]
```

Out[4]:
```
1
```

In [5]:
```
# Print the target_name of the first document using y_names and y.
y_names[y[0]]
```

Out[5]:
```
'sci.space'
```

Use CountVectorizer to Convert To TF

In [6]:
```
# 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 repr
esentation.
# 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[6]:
```
(1187, 3628)
```

In [7]:
```
# The stopwords learned by cvect are stored as a set in cvect.stop_wo
rds_
# 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
 vary over runs.
stop_words_list[:5]
```

Out[7]:
```
['awarded usaf', 'article foo', 'old grandma', 'drivability', 'russ']
```

Calculate Mean CV Accuracy Using RandomForestClassifier

In [8]:
```
# 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 abov
e .9
np.mean(cv_scores)
```

Out[8]:
```
0.9730666649839511
```

Optional: Find Important Features

In [9]:
```
# CountVectorizer stores the feature names (terms in the vocabulary)
 in two ways:
#  1. as a dictionary of term:colum_index pairs, accessed via cvect.v
ocabulary_
#  2. as a list of terms, in column index order, accessed via cvect.g
et_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 RandomForestClassif
ier
#
# The terms you find will likely not be surprising given the document
categories.
```

In [10]:
```
vocab_terms = np.array(cvect.get_feature_names())
vocab_terms[:10]
```

Out[10]:
```
array(['00', '000', '000 miles', '01', '01 14', '02', '03', '04', '04
01',
       '06'], dtype='<U26')
```

In [11]:
```
rf = RandomForestClassifier(n_estimators=50).fit(X_tf,y)
sorted(rf.feature_importances_,reverse=True)[:5]
```

Out[11]:
```
[0.06870630317863295,
 0.04009229799821716,
 0.0381058327881347,
 0.02688542021191862,
 0.01787443576967728]
```

In [12]:
```
sorted_feature_indexes = np.argsort(rf.feature_importances_)[::-1]
vocab_terms[sorted_feature_indexes][:10]
```

Out[12]:
```
array(['car', 'space', 'nasa', 'cars', 'moon', 'sci', 'orbit', 'launc
h',
       'science', 'com pat'], dtype='<U26')
```