Week 10 Quiz

Clarissa Tai - rt2822

Due Tues. Nov 15th 11:59pm

In this quiz, we're going to load documents from 2 categories (space, cars) in the 20newsgroups dataset. The goal is to train a classifier that classifies documents into these 2 categories based on a term frequency representation of the documents. We will then calculate mean cross-validaion accuracy of a RandomForestClassifier using this transformation.

Instructions

Replace the Name and UNI in cell above and the notebook filename

Replace all '__' below using the instructions provided.

When completed,

1. make sure you've replaced Name and UNI in the first cell and filename
2. Kernel -> Restart & Run All to run all cells in order
3. Print Preview -> Print (Landscape Layout) -> Save to pdf
4. post pdf to GradeScope

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 using fetch_20newsgroups().
#   Only fetch the two categories of interest using categories=['sci.space','rec.autos']
# Store in the result into newsgroups
newsgroups = fetch_20newsgroups(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 = newsgroups.target
y_names = newsgroups.target_names

# Print the number of observations by printing the length of docs
#  You should get 1187
len(docs)
```

Out[2]:
```
1187
```

In [3]:
```
# Print the text of the first document in docs.
# Note: try printing both with and without the print() statement
#  with the print statement, linebreaks are parsed,
#  without, linebreaks are printed as excape characters
print(docs[0]) # with print
docs[0] # without print
```

```
From: prb@access.digex.com (Pat)
Subject: Re: Proton/Centaur?
Organization: Express Access Online Communications USA
Lines: 15
NNTP-Posting-Host: access.digex.net


Well thank you dennis for your as usual highly detailed and informative
posting.

The question i have about the proton, is  could it be  handled at
one of KSC's spare pads, without major  malfunction,  or could it be
handled at kourou  or Vandenberg?

Now if it uses storables,  then  how long would it take for the russians
to equip something at cape york?

If  Proton were launched from a western site,  how would it compare to the
T4/centaur?   As i see it, it should lift  very close to the T4.

pat
```

Out[3]:
```
"From: prb@access.digex.com (Pat)\nSubject: Re: Proton/Centaur?\nOrganization: Express Access Online Communications US
A\nLines: 15\nNNTP-Posting-Host: access.digex.net\n\n\nWell thank you dennis for your as usual highly detailed and inf
ormative \nposting.   \n\nThe question i have about the proton, is  could it be  handled at\none of KSC's spare pads,
without major  malfunction,  or could it be\nhandled at kourou  or Vandenberg?   \n\nNow if it uses storables,  then
how long would it take for the russians\nto equip something at cape york?\n\nIf  Proton were launched from a western s
ite,  how would it compare to the\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
```

```
from sklearn.feature_extraction.text import CountVectorizer

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

# Fit cvect on docs and transform docs into their term frequency representation.
# 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 above
#  and the number of columns should be near 6000
X_tf.shape
```

Out[6]:
```
(1187, 5893)
```

In [7]:
```
# Print out the last 5 terms in the learned vocabulary in cvect
#   using .get_feature_names_out() which returns a list of terms corresponding
#   to the order of the columns in X_tf
# They should all be related to zoos or zoology
cvect.get_feature_names_out()[-5:]
```

Out[7]:
```
array(['zoo', 'zoo toronto', 'zoology', 'zoology kipling',
       'zoology lines'], dtype=object)
```

In [8]:
```
# 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 vary over runs.
stop_words_list[:5]
```

Out[8]:
```
['profit corporation', 'buy our', 'car never', 'look travis', 'darken my']
```

Calculate Mean CV Accuracy Using RandomForestClassifier

In [9]:
```
# Import cross_val_score from sklearn
from sklearn.model_selection import cross_val_score

# Import RandomForestClassifier from sklearn
from sklearn.ensemble import RandomForestClassifier

# Get a set of 5-fold CV scores using
#  a RandomForestClassifier
#   with 50 trees
#   and n_jobs=-1 all other settings default
#   and the full dataset X_tf and y
# Store as cv_scores
cv_scores = cross_val_score(RandomForestClassifier(n_estimators = 50, n_jobs=-1), X_tf, y)

# Print the mean of these cv_scores rounded to a precision of 2.
#  The mean accuracy should be above .9
cv_scores.mean().round(2)
```

Out[9]:
```
0.97
```

Optional: Find Important Features

In [10]:
```
# CountVectorizer stores the feature names (terms in the vocabulary) in two ways:
#  1. as a dictionary of term:colum_index pairs, accessed via cvect.vocabulary_
#  2. as a list of terms, in column index order, accessed via cvect.get_feature_names_out()
#
# We can get the indices of the most important features by training a new RandomForestClassifier on X_tf,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 with 50 trees
#
# The terms you find will likely not be surprising given the document categories.
important_feat = RandomForestClassifier(n_estimators=50).fit(X_tf,y).feature_importances_
top10_terms = pd.DataFrame(data = important_feat, index = cvect.get_feature_names_out(), columns = ['importance']).sor
top10_terms.index
```

Out[10]:
```
Index(['car', 'space', 'nasa', 'moon', 'nasa gov', 'the car', 'orbit', 'digex',
       'launch', 'cars'],
      dtype='object')
```

In [ ]: