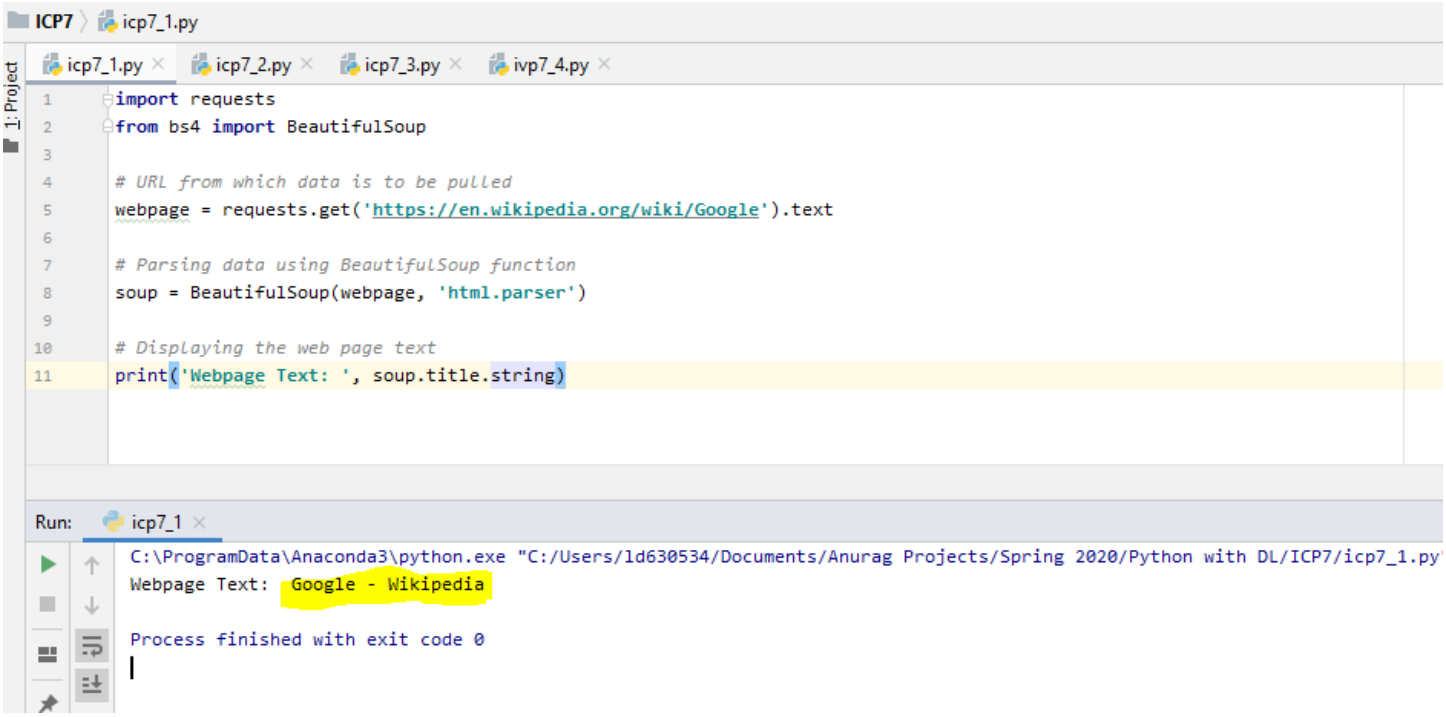
CS5590/490 - Python and Deep Learning Programming
In Class Programming Report - 7
Class ID 24 - Anurag Thantharate
Date Submitted: 03/07/2020

Video Link: https://www.loom.com/share/6df8d95ff1584195b06569aabe5dac7e

---

1. Extract the following web URL text using BeautifulSoup https://en.wikipedia.org/wiki/Google2.
2. Save it in input.txt
3. Apply the following on the text and show output:
    a. Tokenization
    b. POS
    c. Stemming
    d. Lemmatization
    e. Trigram
    f. Named Entity Recognition
4. Change the classifier in the given code to:
    a. KNeighborsClassifier and see how accuracy changes
    b. change the tfidfvectorizer to use bigram and see how the accuracy changes
    TfidfVectorizer(ngram_range=(1,2))
    c. Put argument stop_words='english' and see how accuracy changes



ICP7 〉 icp7_2.py

icp7_1.py | icp7_2.py | icp7_3.py | ivp7_4.py

```
import requests
from bs4 import BeautifulSoup

# URL from which data is to be pulled
webpage = requests.get('https://en.wikipedia.org/wiki/Google').text

# Parsing data using BeautifulSoup function
soup = BeautifulSoup(webpage, 'html.parser')

# Saving the parsed webpage data into the text file titled 'input'
text = soup.get_text()
f= open('input.txt', 'w',encoding='utf-8')
f.write(text)
```

Run: icp7_2

```
C:\ProgramData\Anaconda3\python.exe "C:/Users/1d630534/Documents/Anurag Projects/Spring 2020/Python with DL/ICP7/icp7_2.py"

Process finished with exit code 0
```

Input-txt in Github

ICP7 〉 icp7_3.py

icp7_1.py | icp7_2.py | icp7_3.py | ivp7_4.py

```
import nltk
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer
from nltk.stem import SnowballStemmer
from nltk.stem import WordNetLemmatizer
from nltk import wordpunct_tokenize, pos_tag, ne_chunk
from nltk import ngrams

test_text = open('input.txt', encoding="utf8").read()

# a) Tokenization
token = nltk.word_tokenize(test_text)
print('Tokens identified are', token)


# b) Part Of Speech tagging
pos = nltk.pos_tag(token)
print('Part of Speech associated with the input text', pos)


# c) Stemming - identifying the root or base word of the terms associated
pStemmer = PorterStemmer() #Porter Stemming keeps only prefix for each words and leave non English words like tr
for x in token:
    print('Result of Stemming using PorterStemmer for ', x, 'is ', pStemmer.stem(x))

lStemmer = LancasterStemmer() #Lancaster stemming is a rule-based stemming based on the last letter of the words
for y in token:
    print('Result of Stemming using LancasterStemmer for ', y, 'is ', lStemmer.stem(y))

sStemmer = SnowballStemmer('english')
for z in token:
    print('Result of Stemming using SnowballStemmer for ', z, 'is', sStemmer.stem(z))
```

```
# d) Lemmatization - noramlization of text based on the meaning as part of the speech (converts plurals or adjective to
# their basic, meaningful singular form)
lemmatizer = WordNetLemmatizer()
print('Result of Lemmatization: ', lemmatizer.lemmatize(x))

# e) Trigram
trigram = ngrams(test_text.split(), 3)
for gram in trigram:
    print('Trigram data is ', gram)
print(str(trigram))

# f) Named Entity Recognition
print('Named Entitiy Recognition is ', ne_chunk(pos_tag(wordpunct_tokenize(test_text))))
```

ICP7 〉 ivp7_4.py

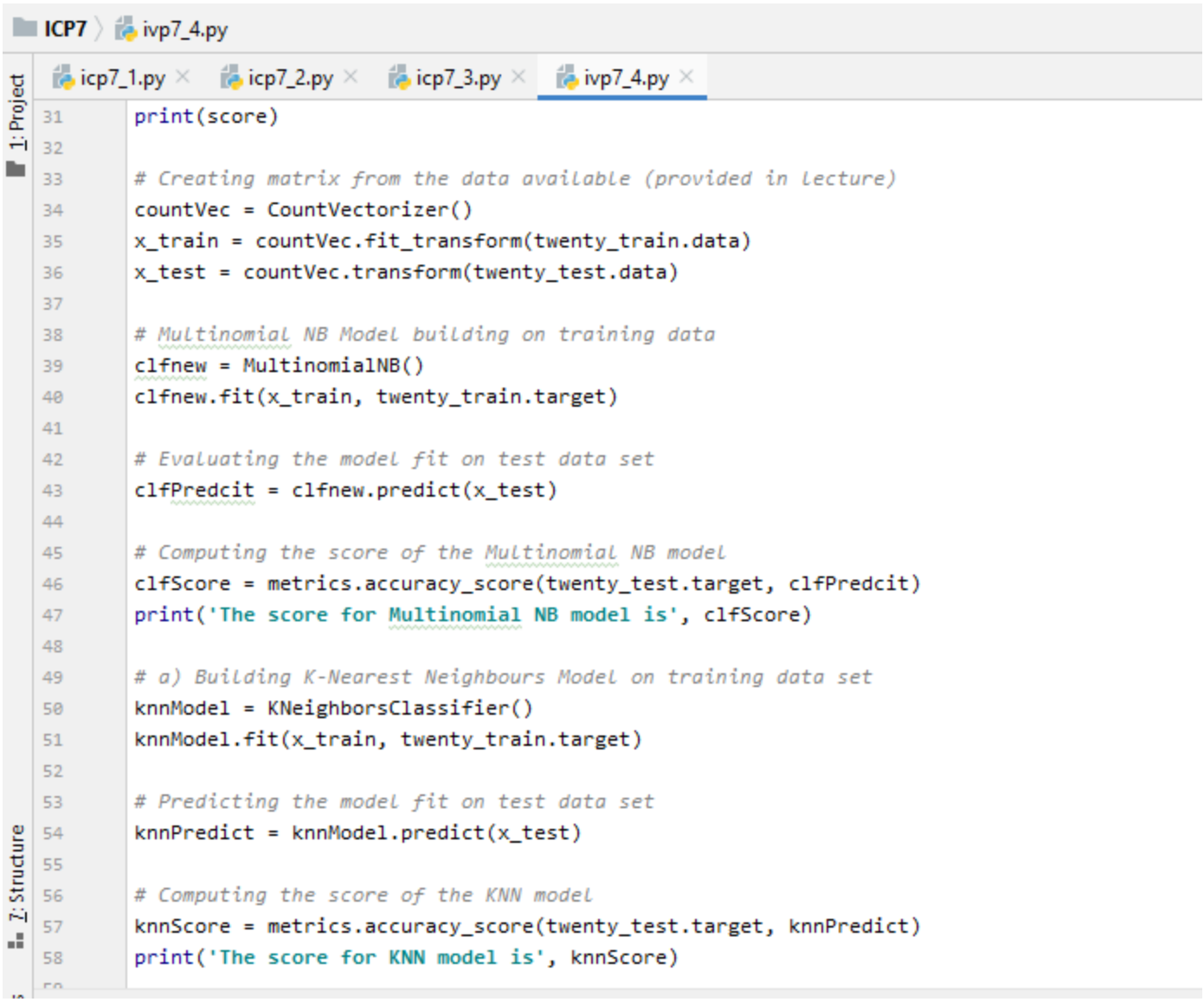
icp7_1.py | icp7_2.py | icp7_3.py | ivp7_4.py

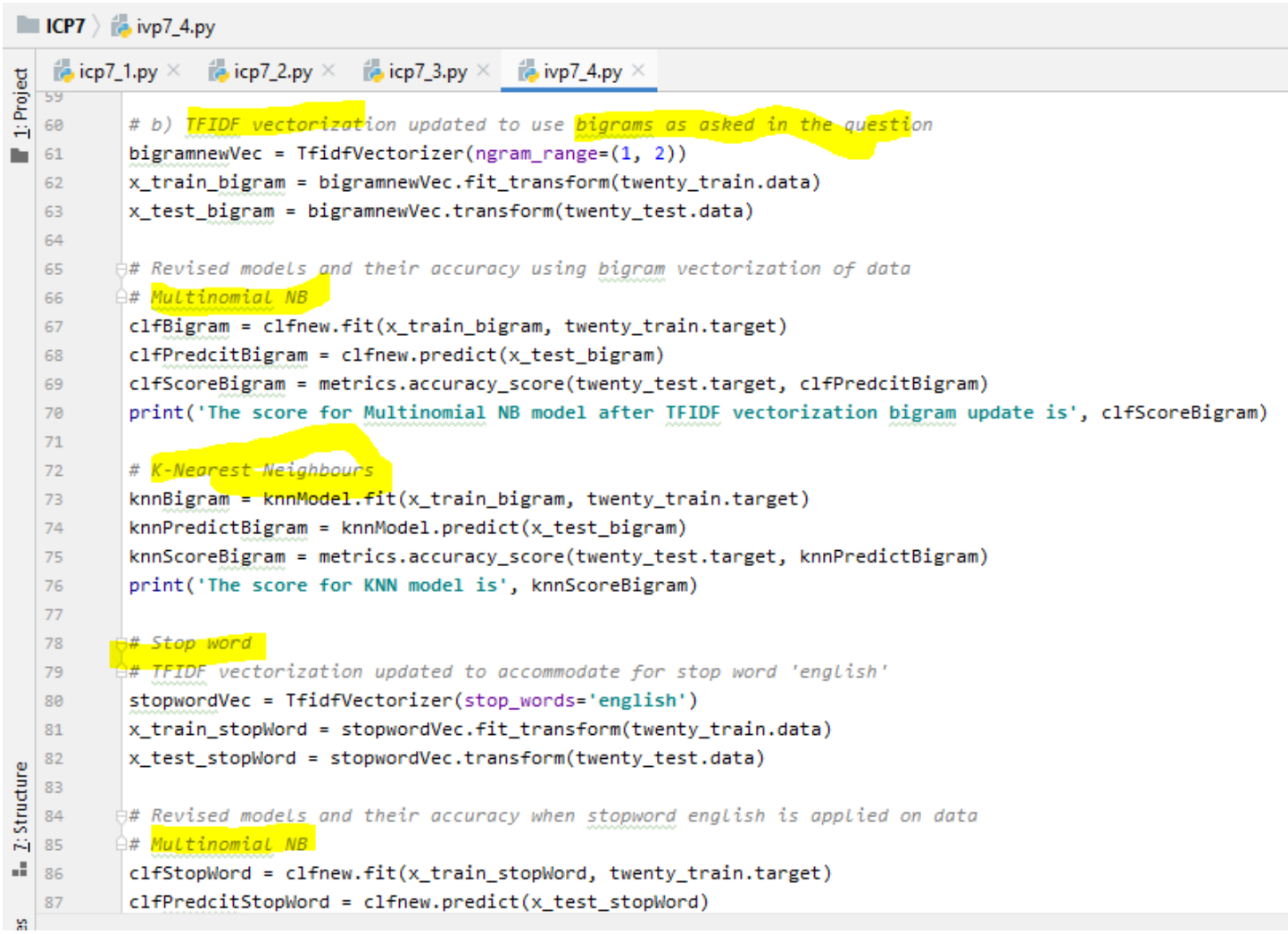
1: Project

```
1   # %%
2   from sklearn.datasets import fetch_20newsgroups
3   from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
4   from sklearn.feature_extraction.text import TfidfTransformer
5   from sklearn import metrics
6   from sklearn.pipeline import Pipeline
7   from sklearn.naive_bayes import MultinomialNB
8   from sklearn.neighbors import KNeighborsClassifier
9   from nltk.corpus import stopwords
10
11  # %%
12  twenty_train = fetch_20newsgroups(subset='train', shuffle=True)
13
14  # %%
15  tfidf_Vect = TfidfVectorizer()
16  X_train_tfidf = tfidf_Vect.fit_transform(twenty_train.data)
17
18  # print(tfidf_Vect.vocabulary_)
19  clf = MultinomialNB()
20  clf.fit(X_train_tfidf, twenty_train.target)
21
22  # %%
23  twenty_test = fetch_20newsgroups(subset='test', shuffle=True)
24  X_test_tfidf = tfidf_Vect.transform(twenty_test.data)
25
26  # %%
27  predicted = clf.predict(X_test_tfidf)
28
```

7: Structure

es


ICP7
ivp7_4.py
icp7_1.py
icp7_2.py
icp7_3.py
ivp7_4.py
1: Project
7: Structure
print(score)
# Creating matrix from the data available (provided in lecture)
countVec = CountVectorizer()
x_train = countVec.fit_transform(twenty_train.data)
x_test = countVec.transform(twenty_test.data)
# Multinomial NB Model building on training data
clfnew = MultinomialNB()
clfnew.fit(x_train, twenty_train.target)
# Evaluating the model fit on test data set
clfPredcit = clfnew.predict(x_test)
# Computing the score of the Multinomial NB model
clfScore = metrics.accuracy_score(twenty_test.target, clfPredcit)
print('The score for Multinomial NB model is', clfScore)
# a) Building K-Nearest Neighbours Model on training data set
knnModel = KNeighborsClassifier()
knnModel.fit(x_train, twenty_train.target)
# Predicting the model fit on test data set
knnPredict = knnModel.predict(x_test)
# Computing the score of the KNN model
knnScore = metrics.accuracy_score(twenty_test.target, knnPredict)
print('The score for KNN model is', knnScore)




```
59
60  # b) TFIDF vectorization updated to use bigrams as asked in the question
61  bigramnewVec = TfidfVectorizer(ngram_range=(1, 2))
62  x_train_bigram = bigramnewVec.fit_transform(twenty_train.data)
63  x_test_bigram = bigramnewVec.transform(twenty_test.data)
64
65  # Revised models and their accuracy using bigram vectorization of data
66  # Multinomial NB
67  clfBigram = clfnew.fit(x_train_bigram, twenty_train.target)
68  clfPredcitBigram = clfnew.predict(x_test_bigram)
69  clfScoreBigram = metrics.accuracy_score(twenty_test.target, clfPredcitBigram)
70  print('The score for Multinomial NB model after TFIDF vectorization bigram update is', clfScoreBigram)
71
72  # K-Nearest Neighbours
73  knnBigram = knnModel.fit(x_train_bigram, twenty_train.target)
74  knnPredictBigram = knnModel.predict(x_test_bigram)
75  knnScoreBigram = metrics.accuracy_score(twenty_test.target, knnPredictBigram)
76  print('The score for KNN model is', knnScoreBigram)
77
78  # Stop word
79  # TFIDF vectorization updated to accommodate for stop word 'english'
80  stopwordVec = TfidfVectorizer(stop_words='english')
81  x_train_stopWord = stopwordVec.fit_transform(twenty_train.data)
82  x_test_stopWord = stopwordVec.transform(twenty_test.data)
83
84  # Revised models and their accuracy when stopword english is applied on data
85  # Multinomial NB
86  clfStopWord = clfnew.fit(x_train_stopWord, twenty_train.target)
87  clfPredcitStopWord = clfnew.predict(x_test_stopWord)
```

```python
# Stop word
# TFIDF vectorization updated to accommodate for stop word 'english'
stopwordVec = TfidfVectorizer(stop_words='english')
x_train_stopWord = stopwordVec.fit_transform(twenty_train.data)
x_test_stopWord = stopwordVec.transform(twenty_test.data)

# Revised models and their accuracy when stopword english is applied on data
# Multinomial NB
clfStopWord = clfnew.fit(x_train_stopWord, twenty_train.target)
clfPredcitStopWord = clfnew.predict(x_test_stopWord)
clfScoreStopWord = metrics.accuracy_score(twenty_test.target, clfPredcitStopWord)
print('The score for Multinomial NB model after stopword english update is', clfScoreStopWord)

# K-Nearest Neighbours
knnStopWord = knnModel.fit(x_train_stopWord, twenty_train.target)
knnPredictStopWord = knnModel.predict(x_test_stopWord)
knnScoreStopWord = metrics.accuracy_score(twenty_test.target, knnPredictStopWord)
print('The score for KNN model is after stop word english is updated is', knnScoreStopWord)
```

ivp7_4

```
C:\ProgramData\Anaconda3\python.exe C:/Users/1d630534/Documents/Anurag Projects/Spring 2020/Python with DL/ICP7/ivp7_4.py
0.7738980350504514
The score for Multinomial NB model is 0.7728359001593202
The score for KNN model is 0.3524960169941583
The score for Multinomial NB model after TFIDF vectorization bigram update is 0.765400955921402
The score for KNN model is 0.6196229421136484
The score for Multinomial NB model after stopword english update is 0.8169144981412639
The score for KNN model is after stop word english is updated is 0.6757833244822092
```
