2 Sentiment Analysis

2.2 Movie Review Data

Let us first start by looking at the data provided with the exercise. We have positive and negative movie reviews labeled by human readers, all positive and negative reviews are in the ‘pos’ and ‘neg’ folders respectively. If you look in- side a sample file, you will see that these review messages have been ‘tokenized’, where all words are separated from punctuations. There are approximately 1000 files in each category with files names starting with cv000, cv001, cv002 and so on. You will split the dataset into training set and testing set.

1. Write some code to load the data from text files.

```
from os import listdir
from os.path import join
from collections import Counter


# function load file text

def load_data (filename):
    file = open(filename , "r")
    text = file.read() # read all
    file.close()
    return text



load_data("review_polarity/txt_sentoken/neg/cv000_29416.txt")
```

```
'plot : two teen couples go to a church party , drink and then drive . \nthey get int
```

```
text_all = []

directory = "review_polarity/txt_sentoken/neg/"
for filename in listdir(directory):
    if not filename.endswith(".txt"):
        continue
    path = directory + "/" + filename
    file = open(path)
    text = file.read()
    file.close()
#     print(text)
    text_all.append(text)

directory = "review_polarity/txt_sentoken/pos/"
for filename in listdir(directory):
    if not filename.endswith(".txt"):
```

```
        continue
    path = directory + "/" + filename
    file = open(path)
    text = file.read()
    file.close()
     #print(text)
    text_all.append(text)


text_all
```

```
['a movie like mortal kombat : annihilation works ( and must be reviewed on ) mult
 "and now the high-flying hong kong style of filmmaking has made its way down to t
 'janeane garofalo in a romantic comedy -- it was a good idea a couple years ago w
 "best remembered for his understated performance as dr . hannibal lecter in micha
 'plot : a young french boy sees his parents killed before his eyes by tim roth ,
 'the happy bastard\'s quick movie review \ndamn that y2k bug . \nit\'s got a head
 "it is movies like these that make a jaded movie viewer thankful for the inventio
 'plot : two teen couples go to a church party , drink and then drive . \nthey get
 ' " quest for camelot " is warner bros . \' first feature-length , fully-animated
 'call it a road trip for the walking wounded . \nstellan skarsg ? rd plays such a
 'that\'s exactly how long the movie felt to me . \nthere weren\'t even nine laugh
 'so ask yourself what " 8mm " ( " eight millimeter " ) is really all about . \nis
 'capsule : in 2176 on the planet mars police taking into custody an accused murde
 'synopsis : a mentally unstable man undergoing psychotherapy saves a boy from a p
 '\'traffic violation\' dr . daniel\'s review of u-turn \ndr . d\'s rating : criti
 "would you believe -- in real life , i mean -- that if you were julia roberts , t
 'one of the most respected names in american independent filmmaking is john sayle
 'of course i knew this going in . \nwhy is it that whenever a tv-star makes a mov
 "there are two things the american film industry should avoid at all costs . \non
 ' " the world on land -- it\'s just too big for me . " \nstarring tim roth , prui
 'i\'m currently accepting all future names for drew barrymore characters . \nin _
 "supposedly based on a true story in which the british drive to build a rail brid
 'one of the responses those that enjoy " detroit rock city " ( probably kiss fans
 'louie is a trumpeter swan with no voice . \nin order to woo his lady love serina
 'a pseudo-intellectual film about the pseudo-intellectual world of art magazines
 "out of sight director steven sorderbergh baffles the hell out of us all in the l
 "lengthy and lousy are two words to describe the boring drama the english patient
 'has it really been two decades since walter matthau coached the bad news bears ?
 "unfortunately it doesn't get much more formulaic than one tough cop . \nthere's
 'funny how your expectations can be defeated , and not in good ways . \nthe ghost
 'talk about a movie that seemed dated before it even hit the theaters ! \nspice w
 'my giant is two movies for the price of one , but neither is worth the cost of a
 "this feature is like a double header , two sets of clich ? s for the price of on
 'one might expect a cathartic viewing experience walking into a new jean-luc goda
 'reindeer games is easily the worst of the three recent films penned by ehren kru
 'a follow-up to disney\'s live-action " 101 dalmatians " that\'s better , more en
 ' " party camp , " is one of the most mindnumbingly brainless comedies i\'ve seen
 'when it comes to the average teenage romantic comedy , i expect negative reviews
 'the classic story & the production which ruined it \nmarking the centennial anni
 "sydney lumet is the director whose work happens to be of varied quality . \nhe i
 'instinct is the kind of movie that inexperienced moviegoers will undoubtedly lab
 'all through its production and into the early days of its initial , aborted pre-
 'capsule : where are you tonight , leni rienfenstal ? \nstarship troopers is an e
 ' " lake placid " marks yet another entry in the series of " predator pics " that
 "among multitude of erotic thrillers , that had been released in the early 1990s
 'hong kong cinema has been going through a bad spell . \nthe last few productions
 'sean connery stars as a harvard law professor who heads back into the courtroom
 'some movies i should just skip . \nmy daughter and i had a really vile time at m
 'i wish i could have been in the pitch meeting for this ridiculous notion of a sp
 'play it to the bone , the newest addition to ron shelton\'s sports - themed repe
```

```
 'if you\'re into watching near on two hours of bored , foul-mouthed florida teens
 'america loves convenience . \nafter all , we\'re the culture that invented the c
 'one-sided " doom and gloom " documentary about the possible annihilation of the
 ' " spawn " features good guys , bad guys , lots of fighting , bloody violence ,
 'the law of crowd pleasing romantic movies states that the two leads must end up
 'i\'m really starting to wonder about alicia silverstone . \nsure , she is one of
 'so what do you get when you mix together plot elements from various successful s
 ' " knock off " is exactly that : a cheap knock off of an action movie . \nit\'s
```

```
!pip install -U nltk
```

```
Requirement already up-to-date: nltk in c:\users\khing\anaconda3\lib\site-packages (3
Requirement already satisfied, skipping upgrade: click in c:\users\khing\anaconda3\li
Requirement already satisfied, skipping upgrade: joblib in c:\users\khing\anaconda3\l
Requirement already satisfied, skipping upgrade: tqdm in c:\users\khing\anaconda3\lib
Requirement already satisfied, skipping upgrade: regex in c:\users\khing\anaconda3\li
```

```
# import nltk
# nltk.download()


# Clean text data
from nltk.corpus import stopwords
import string
def clean_text(data):
    dataframe = data.split()
    table = str.maketrans("","",string.punctuation)
    dataframe = [w.translate(table) for w in dataframe]
    dataframe = [word for word in dataframe if word.isalpha()]

    stop_words = set(stopwords.words('english'))
    dataframe = [w for w in dataframe if not w in stop_words]

    dataframe = [word for word in dataframe if len(word) >1 ]
    return dataframe


def all_text (directory , vocab):
    load_line = []
    for filename in listdir(directory):
        if not filename.endswith(".txt"):
            continue
        path = directory + "/" + filename
        #add_text_to_vocab(path , vocab)
        load = add_text_to_vocab(path , vocab)
        load_line.append(load)
    return load_line


def add_text_to_vocab(filename , vocab):
    text = load_data(filename) # load test
    dataframe = clean_text(text)
    vocab.update(dataframe)
    # filter vocab
```

```
        dataframe = [w for w in dataframe if w in vocab]
        return ''.join(dataframe)


    # define vocab (pos , neg)
    vocab = Counter()
    vocab_pos = all_text("review_polarity/txt_sentoken/pos",vocab)
    vocab_neg = all_text("review_polarity/txt_sentoken/neg",vocab)


    len(vocab)
```

```
     46557
```

```
    vocab.most_common(20)
```

```
     [('film', 8860),
      ('one', 5521),
      ('movie', 5440),
      ('like', 3553),
      ('even', 2555),
      ('good', 2320),
      ('time', 2283),
      ('story', 2118),
      ('films', 2102),
      ('would', 2042),
      ('much', 2024),
      ('also', 1965),
      ('characters', 1947),
      ('get', 1921),
      ('character', 1906),
      ('two', 1825),
      ('first', 1768),
      ('see', 1730),
      ('well', 1694),
      ('way', 1668)]
```

```
    # Save Prepared data
    def save_text (path , filename):
        data = '\n'.join(path)
        file = open(filename,"w")
        file.write(data)
        file.close()


    # หาอันที่มีโอกาสเกิดน้อยที่สุด
    min_occurrence = 5
    tokens = [i for i,j in vocab.items() if j >= min_occurrence]


    print(tokens , len(tokens))
```

```
     ['linda', 'fiorentino', 'disappeared', 'radar', 'deservedly', 'turn', 'cable', 'pic',
```

```
    save_text(tokens,"data_vocab.txt")
```

```
# load vocab
df_vocab = "data_vocab.txt"
df_vocab = load_data(df_vocab)
df_vocab = df_vocab.split()
df_vocab = set(df_vocab)
neg_data = all_text("review_polarity/txt_sentoken/neg/" , vocab)
pos_data = all_text("review_polarity/txt_sentoken/pos/" , vocab)

# save pos_data and neg_data
save_text(neg_data , "data_neg.txt")
save_text(pos_data , "data_pos.txt")
```

2.3 TF-IDF

From a raw text review, you want to create a vector, whose elements indicate the number of each word in each document. The frequency of all words within the documents are the ‘features’ of this machine learning problem.

A popular method for transforming a text to a vector is called tf-idf, short for term frequencyinverse document frequency.

1. Conduct a research about tf-idf and explain how it works.
2. Scikit-learn provides a module for calculating this, this is called TfidfVec- torizer. You can study how this function is used here:

`http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`

Write code to transform your text to tf-idf vector.

Conduct a research about tf-idf and explain how it works.

หลังการทำงานของ Tf-idf คือ ใช้เปรียบเทียบความเหมือนกันของคำสอนคำ โดยวัดจาก tf และ idf การวัด cosine ตรงๆระหว่างคำสองคำไม่ค่อยเหมาะ เนื่องจากข้อมูลจะเบ้หนักมาก แล้วก็แยกแยะกันได้ยาก เช่น คำพวก the , it , they พวกนี้จะมาทำการแยกยากเพราะไม่ค่อยให้ข้อมูลอะไร เราเลยจะใช้ Tf-idf เข้ามาช่วย

Tf : Term frequency คือการนับความถี่ของคำ (t) ในเอกสาร (d) จำนวนคำ tf = count(t,d) หรือเราสามาtake log ฐาน 10 เข้าไปเพื่อปรับสเกลไม่ให้เวอร์เกินไป tf = log10(count(t,d)+1) (+1 เพราะว่าไม่ให้เกิดเคส log0) idf : Inverse Document Frequency df : document Frequency คือจะให้ weight เยอะๆสำหรับคำที่เจอไม่บ่อยในเอาสารมันมาจากแนวคิดว่าถ้าคำไหนเจอบ่อยๆใน เอกสารจะแปลว่ามันไม่สำคัญ df จะแตกต่างจาก collection frquency ตรงที่ถ้ากรอบ collection ของเราคือ document หลายๆชิ้นเวลานับ document frquency เราจะนับคำนั้นๆไปโผล่ใน document กี่ชิ้น ส่วน collection fequency จะนับว่าคำนั้นไปโผล่ใน collection (document ไหนก็ได้) กี่ครั้ง

```
from IPython import display
display.Image("pic_tf-idf/0_nFyAU7l38Wo1dHhK.png")
```

	Collection Frequency	Document Frequency
Romeo	113	1
action	113	31

เราจะใช้ Inverse Document Frequency ดทนเพื่อให้ตีความง่านกว่า Df มีสูตร คือ N/df โดย N คือจำนวน document ใน collection ส่วน df คือจำนวน document ที่มีคำนั้นๆ นอกจากนี้ก็มีการ take log ฐาน10 เขาไปเช่นกัน เพื่อปรับสเกล idf = log10(N/df)

```
display.Image("pic_tf-idf/0_BM0SUOJBB4niQs0f.png")
```

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

ตามรูป N = 37 ทำให้ good กับ sweet ที่มี df = 37 คือ idf ออกมาได้ 0 ยิ่งค่า idf สูงๆแปลว่าคำๆนั้นสำคัญมาก สรุป tf-idf สามารถคำนวณตรงๆ ได้เลยคือ w = tf * idf

```
display.Image("pic_tf-idf/0_xpFqparOAIPNTZw9.png")
```

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Figure 6.8 A tf-idf weighted term-document matrix for four words in four Shakespeare plays, using the counts in Fig. 6.2. For example the 0.049 value for *wit* in *As You Like It* is the product of $\text{tf} = \log_{10}(20+1) = 1.322$ and idf = .037. Note that the idf weighting has eliminated the importance of the ubiquitous word *good* and vastly reduced the impact of the almost-ubiquitous word *fool*.

เราจะใช้ tf-idf เป็นค่ามาตรฐาน (baseline) สำหรับพิจรณา weighting ของ cooccurrence matrix

Refer: https://medium.com/@sirasith.petch/word-embedding-tf-idf-%E0%B9%81%E0%B8%A5%E0%B8%B0-word2vec-%E0%B8%84%E0%B8%B7%E0%B8%AD%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3-

%E0%B9%81%E0%B8%A5%E0%B9%89%E0%B8%A7%E0%B8%A1%E0%B8%B1%E0%B8%99%E0%B8%A1%E0%B8%B5%E0%B8%9B%E0%B8%A3%E0%B8%B0%E0%B9%82%E0%B8%A2%E0%B8%8A%E0%B8%99%E0%B9%8C%E0%B8%A2%E0%B8%B1%E0%B8%87%E0%B9%84%E0%B8%87-9a6c593cf507

Scikit-learn provides a module for calculating this, this is called TfidfVec- torizer. You can study how this function is used here

```
from sklearn.feature_extraction.text import TfidfVectorizer

text_all = []
buff = 0
directory = "review_polarity/txt_sentoken/neg/"
for filename in listdir(directory):
    if not filename.endswith(".txt"):
        continue
    path = directory + "/" + filename
    file = open(path)
    text = file.read()
    file.close()
#     print(text)
    text_all.append(text)

directory = "review_polarity/txt_sentoken/pos/"
for filename in listdir(directory):
    if not filename.endswith(".txt"):
        continue
    path = directory + "/" + filename
    file = open(path)
    text = file.read()
    file.close()
     #print(text)
    text_all.append(text)


print(text_all)
```

```
['a movie like mortal kombat : annihilation works ( and must be reviewed on ) multipl
```

```
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(text_all)


X.shape
```

```
(2000, 39659)
```

```
vectorizer.get_feature_names()
```

```
['00',
 '000',
 '0009f',
 '007',
 '00s',
 '03',
 '04',
 '05',
 '05425',
 '10',
 '100',
 '1000',
 '10000',
 '100m',
 '101',
 '102',
 '103',
 '104',
 '105',
 '106',
 '107',
 '108',
 '109',
 '10b',
 '10s',
 '10th',
 '11',
 '110',
 '111',
 '112',
 '113',
 '1138',
 '114',
 '115',
 '117',
 '118',
 '11th',
 '12',
 '121',
 '122',
 '123',
 '125',
 '126',
 '127',
 '1272',
 '128',
 '129',
 '1298',
 '12th',
 '13',
 '130',
 '1305',
 '131',
 '132',
 '133',
 '135',
 '137',
 '138',
 '139',
```

2.4 Classification

Use 4 different models to classify each movie into positive or negative category.

1. K-Nearestneighbormodel,using module `sklearn.neighbors.KNeighborsClassifier`
2. RandomForest, using module `sklearn.ensemble.RandomForestClassifier`
3. SVM, using module `sklearn.svm.SVC`
4. Neural network, using `sklearn.neural_network.MLPClassifier`

You may pick other models you would like to try. Just present results for at least 4 models. Please provide your code for model fitting and cross validation. Calculate your classification accuracy, precision, and recall.

```
Data_dir = "review_polarity/txt_sentoken/"
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_files
```

```
data = load_files(Data_dir , encoding = "utf-8" , decode_error="replace")
```

```
data
```

```
{'data': ["arnold schwarzenegger has been an icon for action enthusiasts , since t
  "good films are hard to find these days . \ngreat films are beyond rare . \nproo
  "quaid stars as a man who has taken up the proffesion of dragonslayer after he f
  'we could paraphrase michelle pfieffer\'s character in dangerous minds and say t
  "kolya is one of the richest films i've seen in some time . \nzdenek sverak play
  'i don\'t know how many other people have had the idea cross their mind that the
  'capsule : trippy , hyperspeed action machine from hong kong\'s accomplished tsu
  'gothic murder-mystery yarns are not a new conceit . \nin fact , they\'ve been a
  "what i look for in a movie is not necessarily perfection . \nsometimes a movie
  'the law of crowd pleasing romantic movies states that the two leads must end up
  'it\'s ironic that the best films in cinema history are invariably the original
  "getting it right is a far far cry from the teenage sex comedy you might expect
  'synopsis : blond criminal psychologist sarah chooses to copulate with greasy to
  'after enduring mariah carey\'s film debut , glitter , i\'m reminded of a bit fr
  'tim burton has now completed his evolution from the brilliant director of macab
  'bill condon\'s " gods and monsters " is a fascinating look into the last days i
  'a big , busy boxing satire with a surprisingly paltry punch , the great white h
  "the premise is simple , if not bizarre . \na mad scientist ( trace beaulieu as
  'no matter what you suspect , this isn\'t your usual action thriller . \nthe usu
  'when i first heard of contact , the hype was building it up as a sci-fi blockbu
  'hedwig ( john cameron mitchell ) was born a boy named hansel in east berlin . \
  "near the end of frank capra's holiday classic , its a wonderful life , george b
  ' * * * be warned . . . \nthe following review contains some harsh language * *
  'i can already feel the hate letters pouring in on this one , folks . \ni loved
  "accepting his oscar as producer of this year's best picture winner , saul zaent
  ' " if there\'s a beast in men , it meets its match in women , too . " \nstarrin
  'i have a great idea for a movie , one that can\'t miss . \nsee , i\'ve got cool
  'warren beatty returns to the screens in the funniest , craziest and hard hittin
  '14 years ago , national lampoon introduced us to a new family - the griswolds .
```

```
 'like the great musical pieces of mozart himself , amadeus is a true work of art
 'that\'s the question i asked myself when venturing nervously into " the wedding
 'on re-watching italian writer/director dario argento\'s much lauded murder myst
 'the idea at the center of the devil\'s advocate , which is , thus far , one of
 'what\'s shocking about " carlito\'s way " is how good it is . \nhaving gotten a
 'disney\'s 35th animated feature-- a retooling of the olympian legend crossed wi
 ' " mandingo " has traditionally been seen as one of two things : either a much-
 'hilarious , ultra-low budget comedy from film school dropout kevin smith chroni
 'senseless ( r ) marlon wayans is a very talented physical comedian , and it is
 'at one point in this movie there is a staging of an opera that goes completely
 'if anything , " stigmata " should be taken as a warning against releasing simil
 'reindeer games is easily the worst of the three recent films penned by ehren kr
 'most movies seem to release a third movie just so it can be called a trilogy .
 "the release of dolores claiborne into wide release this weekend adds another en
 'so what do you get when you mix together plot elements from various successful
 'did you ever wonder if dennis rodman was actually from this planet ? \nor if sy
 "terrence malick made an excellent 90 minute film adaptation of james jones' wor
 'capsule : gal is a 50s-ish london cockney gangster who has retired to spain . \
 'i\'ve heard it called " jaws with claws " and that\'s a fair summation of the p
 'tempe mills cinema , az--this movie had us in non-stop stitches from beginning
 "as with his other stateside releases , jackie chan's latest chopsocky vehicle ,
 "to me , nicolas cage sounds like an ideal choice for the lead role in a martin
 "review : a dog of flanders is the story of an adorable little boy named nello a
 'susan granger\'s review of " the watcher " ( universal ) \njust what we need :
 'godzilla is a nuclear freak . \nhe is a lizard that has been mutated over the y
 'the main problem with martin lawrence\'s pet project , a thin line between love
 "the plot of big momma's house is martin lawrence in a fat suit and a dress . \n
 'miramax " disinvited " on-line media from press screenings of scream 3 . \nthey
 'david lynch\'s " blue velvet " begins and ends with colorful , bright shots of
```

```
# Calculate count of each category
labels , counts = np.unique(data.target , return_counts = True)
```

```
labels_str = np.array(data.target_names)[labels]
labels_str = dict(zip(labels_str , counts))
```

```
labels_str
```

```
{'neg': 1000, 'pos': 1000}
```

```
data.data
```

```
["arnold schwarzenegger has been an icon for action enthusiasts , since the late 8
 "good films are hard to find these days . \ngreat films are beyond rare . \nproof
 "quaid stars as a man who has taken up the proffesion of dragonslayer after he fe
 'we could paraphrase michelle pfieffer\'s character in dangerous minds and say th
 "kolya is one of the richest films i've seen in some time . \nzdenek sverak plays
 'i don\'t know how many other people have had the idea cross their mind that thei
 'capsule : trippy , hyperspeed action machine from hong kong\'s accomplished tsui
 'gothic murder-mystery yarns are not a new conceit . \nin fact , they\'ve been ar
 "what i look for in a movie is not necessarily perfection . \nsometimes a movie h
 'the law of crowd pleasing romantic movies states that the two leads must end up
 'it\'s ironic that the best films in cinema history are invariably the original d
 "getting it right is a far far cry from the teenage sex comedy you might expect f
 'synopsis : blond criminal psychologist sarah chooses to copulate with greasy ton
 'after enduring mariah carey\'s film debut , glitter , i\'m reminded of a bit fro
```

```
 'tim burton has now completed his evolution from the brilliant director of macabr
 'bill condon\'s " gods and monsters " is a fascinating look into the last days in
 'a big , busy boxing satire with a surprisingly paltry punch , the great white hy
 "the premise is simple , if not bizarre . \na mad scientist ( trace beaulieu as d
 'no matter what you suspect , this isn\'t your usual action thriller . \nthe usua
 'when i first heard of contact , the hype was building it up as a sci-fi blockbus
 'hedwig ( john cameron mitchell ) was born a boy named hansel in east berlin . \n
 "near the end of frank capra's holiday classic , its a wonderful life , george ba
 ' * * * be warned . . . \nthe following review contains some harsh language * * *
 'i can already feel the hate letters pouring in on this one , folks . \ni loved t
 "accepting his oscar as producer of this year's best picture winner , saul zaentz
 ' " if there\'s a beast in men , it meets its match in women , too . " \nstarring
 'i have a great idea for a movie , one that can\'t miss . \nsee , i\'ve got cooli
 'warren beatty returns to the screens in the funniest , craziest and hard hitting
 '14 years ago , national lampoon introduced us to a new family - the griswolds .
 'like the great musical pieces of mozart himself , amadeus is a true work of art
 'that\'s the question i asked myself when venturing nervously into " the wedding
 'on re-watching italian writer/director dario argento\'s much lauded murder myste
 'the idea at the center of the devil\'s advocate , which is , thus far , one of t
 'what\'s shocking about " carlito\'s way " is how good it is . \nhaving gotten a
 'disney\'s 35th animated feature-- a retooling of the olympian legend crossed wit
 ' " mandingo " has traditionally been seen as one of two things : either a much-n
 'hilarious , ultra-low budget comedy from film school dropout kevin smith chronic
 'senseless ( r ) marlon wayans is a very talented physical comedian , and it is t
 'at one point in this movie there is a staging of an opera that goes completely w
 'if anything , " stigmata " should be taken as a warning against releasing simila
 'reindeer games is easily the worst of the three recent films penned by ehren kru
 'most movies seem to release a third movie just so it can be called a trilogy . \
 "the release of dolores claiborne into wide release this weekend adds another ent
 'so what do you get when you mix together plot elements from various successful s
 'did you ever wonder if dennis rodman was actually from this planet ? \nor if syl
 "terrence malick made an excellent 90 minute film adaptation of james jones' worl
 'capsule : gal is a 50s-ish london cockney gangster who has retired to spain . \n
 'i\'ve heard it called " jaws with claws " and that\'s a fair summation of the pl
 'tempe mills cinema , az--this movie had us in non-stop stitches from beginning t
 "as with his other stateside releases , jackie chan's latest chopsocky vehicle ,
 "to me , nicolas cage sounds like an ideal choice for the lead role in a martin s
 "review : a dog of flanders is the story of an adorable little boy named nello an
 'susan granger\'s review of " the watcher " ( universal ) \njust what we need : a
 'godzilla is a nuclear freak . \nhe is a lizard that has been mutated over the ye
 'the main problem with martin lawrence\'s pet project , a thin line between love
 "the plot of big momma's house is martin lawrence in a fat suit and a dress . \nt
 'miramax " disinvited " on-line media from press screenings of scream 3 . \nthey
 'david lynch\'s " blue velvet " begins and ends with colorful , bright shots of f
```

```
data.target
```

```
array([0, 1, 1, ..., 1, 0, 0])
```

```
from sklearn.feature_extraction.text import TfidfVectorizer ,TfidfTransformer
vectorizer = TfidfVectorizer(stop_words="english"  , decode_error= "ignore").fit(data.data
tf_idf = TfidfTransformer()
vector = tf_idf.fit_transform(vectorizer.fit_transform(data.data)) # x
print(vector.shape)
```

```
(2000, 39354)
```

```
# Data preparation
```

```
from sklearn.model_selection import train_test_split
X_train , x_test , y_train , y_test = train_test_split(vector , data.target , test_size =
```

```
X_train.shape
```

```
(1600, 39354)
```

```
y_train.shape
```

```
(1600,)
```

```
x_test.shape
```

```
(400, 39354)
```

Build model

1.K-Nearestneighbor

2.RandomForest

3.SVM

4.Neural network

Just present results for at least 4 models. Please provide your code for model fitting and cross validation.

Calculate your classification accuracy, precision, and recall.

K-Nearestneighbor

```
from sklearn.model_selection import cross_val_predict , cross_val_score
from sklearn.metrics import  accuracy_score , precision_score , recall_score
from sklearn.neighbors import KNeighborsClassifier
np.random.seed(42)

neigh = KNeighborsClassifier(n_neighbors=5)
model = neigh.fit(X_train,y_train)

y_pred = model.predict(x_test)

# cross validation
print(f"model (KNN) accuracy  : {accuracy_score(y_test,y_pred):.2f}")
print(f"model (KNN) precision : {precision_score(y_test, y_pred):.2f}")
print(f"model (KNN) recall    : {recall_score(y_test, y_pred):.2f}")
```

```
model (KNN) accuracy  : 0.66
model (KNN) precision : 0.67
```

```
model (KNN) recall    : 0.68
```

RandomForest

```
from sklearn.ensemble import RandomForestClassifier

np.random.seed(42)

clf = RandomForestClassifier(n_estimators=100, criterion="gini")
model = clf.fit(X_train ,y_train)

y_pred = model.predict(x_test)
print(f"model (RandomForest) accuracy  : {accuracy_score(y_test,y_pred):.2f}")
print(f"model (RandomForest) precision : {precision_score(y_test, y_pred):.2f}")
print(f"model (RandomForest) recall    : {recall_score(y_test, y_pred):.2f}")
```

```
model (RandomForest) accuracy  : 0.80
model (RandomForest) precision : 0.85
model (RandomForest) recall    : 0.76
```

SVM

```
from sklearn.svm import SVC
np.random.seed(42)
svm = SVC(C=1 , kernel="sigmoid" , degree = 3)
model = svm.fit(X_train , y_train)
y_pred = model.predict(x_test)

print(f"model (SVM) accuracy  : {accuracy_score(y_test,y_pred):.2f}")
print(f"model (SVM) precision : {precision_score(y_test , y_pred):.2f}")
print(f"model (SVM) recall    : {recall_score(y_test, y_pred):.2f}")
```

```
model (SVM) accuracy  : 0.80
model (SVM) precision : 0.82
model (SVM) recall    : 0.79
```

Neural network

```
from sklearn.neural_network import MLPClassifier
np.random.seed(42)
Nn = MLPClassifier(hidden_layer_sizes=100 , activation='relu'
                   ,alpha= 0.0001 , learning_rate='constant')
# y_pred_Nn = cross_val_predict(Nn , vector , y , cv = 5)
model = Nn.fit(X_train , y_train)
y_pred = model.predict(x_test)
print(f"model (Neural network) accuracy  : {accuracy_score(y_test,y_pred):.2f}")
print(f"model (Neural network) precision : {precision_score(y_test , y_pred):.2f}")
```

```
print(f"model (Neural network) recall    : {recall_score(y_test ,y_pred):.2f}")
```

```
model (Neural network) accuracy  : 0.81
model (Neural network) precision : 0.82
model (Neural network) recall    : 0.80
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report
y = data.target
KNN = KNeighborsClassifier(n_neighbors=5)
RFC = RandomForestClassifier(n_estimators=100, criterion="gini")
SVM = SVC(C=1 , kernel="sigmoid" , degree = 3)
MLP = MLPClassifier(hidden_layer_sizes=100 , activation='relu'
                    ,alpha= 0.0001 , learning_rate='constant')
names_model = ["KNeighbors" , "RandomForest" , "SCV" , "Neural Network"]
list_model = [KNN,
              RFC,
              SVM,
              MLP]
for i , model in enumerate(list_model):
    y_pred = cross_val_predict(model , vector , y , cv =5)
    print(f"Name model : {names_model[i]}")
    print("Classification report")
    print(classification_report (y , y_pred , target_names = ['Neg' , 'Pos']))
```

```
Name model : KNeighbors
Classification report
              precision    recall  f1-score   support

         Neg       0.66      0.60      0.63      1000
         Pos       0.64      0.69      0.67      1000

    accuracy                           0.65      2000
   macro avg       0.65      0.65      0.65      2000
weighted avg       0.65      0.65      0.65      2000

Name model : RandomForest
Classification report
              precision    recall  f1-score   support

         Neg       0.76      0.84      0.80      1000
         Pos       0.82      0.74      0.78      1000

    accuracy                           0.79      2000
   macro avg       0.79      0.79      0.79      2000
weighted avg       0.79      0.79      0.79      2000

Name model : SCV
Classification report
              precision    recall  f1-score   support

         Neg       0.81      0.80      0.81      1000
         Pos       0.80      0.81      0.81      1000

    accuracy                           0.81      2000
   macro avg       0.81      0.81      0.81      2000
weighted avg       0.81      0.81      0.81      2000
```

```
Name model : Neural Network
Classification report
              precision    recall  f1-score   support

         Neg       0.82      0.80      0.81      1000
         Pos       0.81      0.82      0.81      1000

    accuracy                           0.81      2000
   macro avg       0.81      0.81      0.81      2000
weighted avg       0.81      0.81      0.81      2000
```

2.5 Model Tuning

Can you try to beat the simple model you created above? Here are some things you may try:

- When creating TfidfVectorizer object, you may tweak sublinear_tf parameter which use the tf with logarithmic scale instead of the usual tf.
- You may also exclude words that are too frequent or too rare, by adjusting max_df and min_df.
- Adjusting parameters available in the model, like neural network structure or number of trees in the forest.

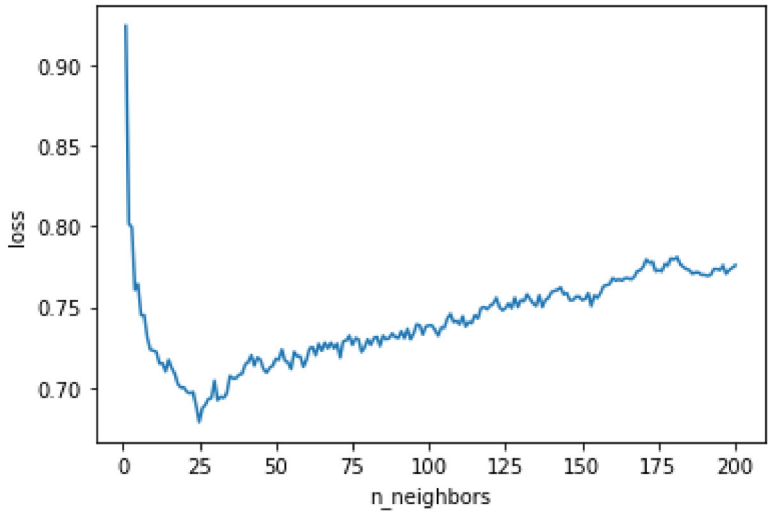
Design at least 3 experiments using these techniques. Show your experimental results.

```
from sklearn.feature_extraction.text import TfidfVectorizer ,TfidfTransformer

vectorizer = TfidfVectorizer(sublinear_tf=True,stop_words="english" ,max_df = 1 , min_df =
tf_idf = TfidfTransformer()
vector = tf_idf.fit_transform(vectorizer.fit_transform(data.data)) # x
print(vector.shape)
```

```
(2000, 15570)
```

```
X_var, x_buff , y_var , y_buff = train_test_split(x_test , y_test , test_size = 0.3 )


# Tune KNN
log_data = {'n_neighbors' : [],
            'loss' : []}
for i in range (1,201):
    KNN = KNeighborsClassifier(n_neighbors=i)
    model = KNN.fit(X_train , y_train)
    y_pred = model.predict(X_var)
    accu = accuracy_score(y_var , y_pred)
    log_data['n_neighbors'].append(i)
    log_data['loss'].append(accu)
plt.plot(log_data['n_neighbors'] ,log_data['loss'])
plt.xlabel('n_neighbors')
plt.ylabel('loss')
```

```
Text(0, 0.5, 'loss')
```

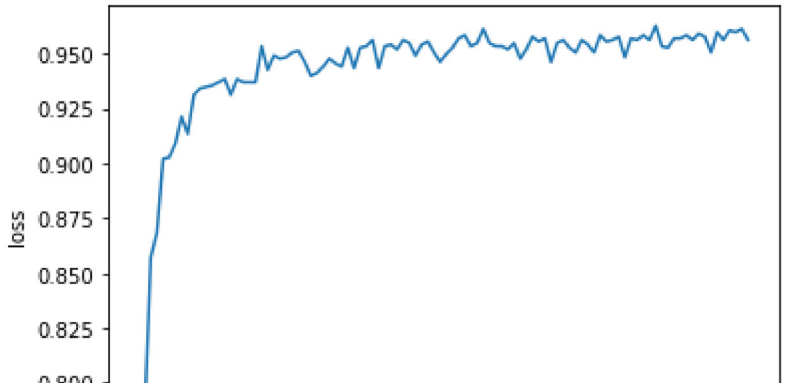



```
KNN = KNeighborsClassifier(n_neighbors= 175)
model = KNN.fit(X_train , y_train)
y_pred = model.predict(x_test)
print(f"Confusion matrix : \
      \n{confusion_matrix(y_test , y_pred)}")
print(f"Classification Report : \
      \n{classification_report (y_test , y_pred , target_names = ['Neg' , 'Pos']) }")
```

```
Confusion matrix :
[[155  44]
 [ 45 156]]
Classification Report :
              precision    recall  f1-score   support

         Neg       0.78      0.78      0.78       199
         Pos       0.78      0.78      0.78       201

    accuracy                           0.78       400
   macro avg       0.78      0.78      0.78       400
weighted avg       0.78      0.78      0.78       400
```

```
# Tune Random Forest
log_data = {'n_estimator' : [],
            'loss' : []}

for i in range(1,101):
    model = RandomForestClassifier(n_estimators=i).fit(X_train , y_train)
    y_pred = model.predict(X_var)
    accu = accuracy_score(y_var , y_pred)
    log_data['n_estimator'].append(i)
    log_data['loss'].append(accu)
plt.plot(log_data['n_estimator'] ,log_data['loss'])
plt.xlabel('n_estimator')
plt.ylabel('loss')
```

```
Text(0, 0.5, 'loss')
```




```
from sklearn.metrics import confusion_matrix
RFC = RandomForestClassifier(n_estimators=85, criterion="gini")
model = RFC.fit(X_train , y_train)
y_pred = model.predict(x_test)
print(f"Confusion matrix : \
      \n{confusion_matrix(y_test , y_pred)}")
print(f"Classification Report : \
      \n{classification_report (y_test , y_pred , target_names = ['Neg' , 'Pos']) }")
```

```
Confusion matrix :
[[168  31]
 [ 56 145]]
Classification Report :
              precision    recall  f1-score   support

         Neg       0.75      0.84      0.79       199
         Pos       0.82      0.72      0.77       201

    accuracy                           0.78       400
   macro avg       0.79      0.78      0.78       400
weighted avg       0.79      0.78      0.78       400
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.neural_network import MLPClassifier
Nn_tuning = MLPClassifier(max_iter=1000 , early_stopping=True , n_iter_no_change=20 )
hidden_layer_sizes = [(70,55,70),(100)]
activation = ["identity" , "logistic" , "tanh" ,"relu"]
solver = ["lbfgs" , "sgd" ,"adam"]
alpha = [0.001 , 0.005 ]
learning_rate = ['constant' , 'invscaling' ,'adaptive']

param_grid = dict(hidden_layer_sizes= hidden_layer_sizes , activation=activation,
                  solver=solver , alpha=alpha , learning_rate = learning_rate)
grid = GridSearchCV(Nn_tuning , param_grid ,n_jobs= -1 , cv=10 )
print(param_grid)
grid.fit(X_train , y_train)
```

```
{'hidden_layer_sizes': [(70, 55, 70), 100], 'activation': ['identity', 'logistic', 't
GridSearchCV(cv=10,
             estimator=MLPClassifier(early_stopping=True, max_iter=1000,
                                     n_iter_no_change=20),
             n_jobs=-1,
             param_grid={'activation': ['identity', 'logistic', 'tanh', 'relu'],
```

```
                         'alpha': [0.001, 0.005],
                         'hidden_layer_sizes': [(70, 55, 70), 100],
                         'learning_rate': ['constant', 'invscaling',
                                           'adaptive'],
                         'solver': ['lbfgs', 'sgd', 'adam']})
```

```
print(f"Good parameters : {grid.best_params_}")
```

```
Good parameters : {'activation': 'relu', 'alpha': 0.005, 'hidden_layer_sizes': (70, 5
```

3 Text Clustering

We have heard about Google News clustering. In this exercise, we are going to implement it with Python.

3.1 Data Preprocessing

Let's switch up and use another dataset called 20newsgroup data, which is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. The data is collected from a university's mailing list, where students exchange opinions in everything from motorcycles to middle east politics.

1. Import data using sklearn.datasets.fetch_20newsgroups
2. Transform data to vector with TfidfVectorizer

```
from sklearn.datasets import fetch_20newsgroups

newsgroups_train = fetch_20newsgroups(subset='train')
```

```
data = newsgroups_train.data
```

```
labels = newsgroups_train.target
```

```
labels
```

```
array([7, 4, 4, ..., 3, 1, 8])
```

```
import numpy as np
true_k = np.unique(labels).shape[0]
true_k
```

```
20
```

```
newsgroups_train.filenames.shape , newsgroups_train.target.shape
```

```
((11314,), (11314,))
```

```
from pprint import pprint
pprint(list(newsgroups_train.target_names))
```

```
['alt.atheism',
 'comp.graphics',
 'comp.os.ms-windows.misc',
 'comp.sys.ibm.pc.hardware',
 'comp.sys.mac.hardware',
 'comp.windows.x',
 'misc.forsale',
 'rec.autos',
 'rec.motorcycles',
 'rec.sport.baseball',
 'rec.sport.hockey',
 'sci.crypt',
 'sci.electronics',
 'sci.med',
 'sci.space',
 'soc.religion.christian',
 'talk.politics.guns',
 'talk.politics.mideast',
 'talk.politics.misc',
 'talk.religion.misc']
```

```
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer,TfidfTransform
from sklearn.pipeline import Pipeline
categories = list(newsgroups_train.target_names)
newsgroups_train = fetch_20newsgroups(subset='train' , categories=categories)

vectorizer = TfidfVectorizer( stop_words='english' , analyzer = 'word')
X = vectorizer.fit_transform(newsgroups_train.data)
```

```
idf = vectorizer.idf_
print(dict(zip(vectorizer.get_feature_names(),idf)))
```

```
{'00': 4.2253700454994085, '000': 4.305969205248958, '0000': 8.359202379228627, '0006
```

```
X.shape
```

```
(18846, 173451)
```

3.2 Clustering

We are going to use the simplest clustering model, k-means clustering, to do this task. Our hope is that this simple algorithm will result in meaningful news categories, without using labels.

1. Fit K-Means clustering model to the text vector. What is the value of K you should pick? Why?

2. Use Silhouette score to evaluate your clusters. Try to evaluate the model for different values of k to see which k fits best for the dataset.

```
from sklearn.cluster import KMeans
from sklearn import metrics
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(42)
score = []
for i in range(2 , 21, 1):
    Km = KMeans(n_clusters=i )
    %time Km.fit(X)
    silhouette_avg = metrics.silhouette_score(X ,Km.labels_)
    score.append(silhouette_avg)
    print("For n_clusters =", i,
          "The average silhouette_score is :", silhouette_avg)

# print(f"V-measure  : {metrics.v_measure_score(labels , Km.labels_):.2f} ")
# print(f"Silhouette : {metrics.silhouette_score(X ,Km.labels_ , sample_size=1000):.4f}")
```

```
 Wall time: 22.4 s
 For n_clusters = 2 The average silhouette_score is : 0.0015261972309054347
 Wall time: 45 s
 For n_clusters = 3 The average silhouette_score is : 0.0017983348908773146
 Wall time: 1min 6s
 For n_clusters = 4 The average silhouette_score is : 0.0018570520623438999
 Wall time: 59.3 s
 For n_clusters = 5 The average silhouette_score is : 0.0022122953712079728
 Wall time: 1min 10s
 For n_clusters = 6 The average silhouette_score is : 0.0026199475624456843
 Wall time: 1min 27s
 For n_clusters = 7 The average silhouette_score is : 0.0029534257434974767
 Wall time: 1min 44s
 For n_clusters = 8 The average silhouette_score is : 0.003022154407938752
 Wall time: 1min 48s
 For n_clusters = 9 The average silhouette_score is : 0.0031377974831349412
 Wall time: 2min
 For n_clusters = 10 The average silhouette_score is : 0.0035428899816962373
 Wall time: 2min 2s
 For n_clusters = 11 The average silhouette_score is : 0.0037668693967347396
 Wall time: 2min 32s
 For n_clusters = 12 The average silhouette_score is : 0.003850687475108001
 Wall time: 1min 59s
 For n_clusters = 13 The average silhouette_score is : 0.004313527188796505
 Wall time: 2min 18s
 For n_clusters = 14 The average silhouette_score is : 0.004581693623427375
 Wall time: 2min 23s
 For n_clusters = 15 The average silhouette_score is : 0.004529035063960511
 Wall time: 2min 36s
 For n_clusters = 16 The average silhouette_score is : 0.00499744247563358
 Wall time: 2min 42s
 For n_clusters = 17 The average silhouette_score is : 0.005233398749474703
 Wall time: 2min 41s
 For n_clusters = 18 The average silhouette_score is : 0.005466850993714914
 Wall time: 2min 47s
```

```
For n_clusters = 19 The average silhouette_score is : 0.005889176721638981
Wall time: 3min 1s
For n_clusters = 20 The average silhouette_score is : 0.0059830681322464475
```

```
k = score.index(max(score)) +2
k
```

```
20
```

3.3 Topic Terms

We want to explore each cluster to understand what news articles are in the cluster, what terms are associated with the cluster. This will require a bit of hacking.

1. Use TfidfVectorizer.get feature names to extract words associated with each dimension of the text vector.
2. Extract cluster's centroids using kmeans.cluster centers .
3. For each centroid, print the top 15 words that have the highest frequency.

```
vectorizer.get_feature_names()
```

```
['00',
 '000',
 '0000',
 '00000',
 '000000',
 '00000000',
 '0000000004',
 '0000000005',
 '00000000b',
 '00000001',
 '00000001b',
 '0000000667',
 '00000010',
 '00000010b',
 '00000011',
 '00000011b',
 '0000001200',
 '00000074',
 '00000093',
 '000000e5',
 '00000100',
 '00000100b',
 '00000101',
 '00000101b',
 '00000110',
 '00000110b',
 '00000111',
 '00000111b',
 '00000315',
 '000005102000',
 '00000510200001',
 '000007',
 '00000ee5',
```

```
 '00001000',
 '00001000b',
 '00001001',
 '00001001b',
 '00001010',
 '00001010b',
 '00001011',
 '00001011b',
 '000010af',
 '00001100',
 '00001100b',
 '00001101',
 '00001101b',
 '00001110',
 '00001110b',
 '00001111',
 '00001111b',
 '000021',
 '000042',
 '000062david42',
 '000094',
 '0000vec',
 '0001',
 '00010000',
 '00010000b',
 '00010001',
```

```
# ค่า k จริงๆ ที่เราจะใช้เราจะต้องดูตามความเหมาะสม เดี๋ยวจะวน loop เพื่อหาค่า k ที่ดีที่สุด
centroids = Km.cluster_centers_.argsort()[:,::-1]
terms = vectorizer.get_feature_names()

for i in range(k):
    print(f"Cluster : {i}")
    for j in centroids[i,:15]:
        print(f'{terms[j]}')
    print("\n")
```

```
Cluster : 0
uk
ac
mathew
mantis
demon
university
writes
subject
lines
organization
cam
com
article
dcs
edu


Cluster : 1
cs
edu
colorado
```

```
university
writes
article
nyx
organization
posting
subject
nntp
host
lines
berkeley
utexas


Cluster : 2
game
team
games
hockey
edu
ca
year
players
season
play
win
baseball
espn
nhl
player


Cluster : 3
stratus
sw
cdt
com
```

```
data[350]
```

```
'From: lhenso@unf6.cis.unf.edu (Larry Henson)\nSubject: IBM link to Imagewriter -- HE
```

```
test_data = vectorizer.transform([data[400]])
Km = KMeans(n_clusters=20)
model = Km.fit(X)
clu = model.predict(test_data)[0]
clu
```

```
5
```