Experiment No-1

Aim:

1. Extract sample document and apply following preprocessing methods.: Tokenization, POS Tagging stop words removal, stemming & lemmatization
2. Create representation of document by calculating Term frequency and inverse document frequency.

Theory:

Extracting sample document

pip install pyPDF2 – for PDF files

pip install python-docx – for docx files

- nltk library - Natural language toolkit. It is platform used for building programs for text analysis
- Python tokenization basically refers to splitting up a larger body of text into smaller lines, words or even creating words for non-english languages.

* Tokenization - basically refers to splitting up a larger body of text into smaller lines, words or even creating words for a non-english language.
  1) Line tokenization
  2) Non-english tokenization
  3) Word tokenization

* Pos tagging - Basically, the goal of a Pos tagger is to assign linguistic information to sub-sentential units, such units are called as tokens (eg. punctuation).

* Stop words removal
- w for w in wordlist if not w in stop-words.

* Stemming and Lemmatization
- used to prepare text, words and documents for further processing

- Language we speak and write are made up of several words often derived from one another. This is called inflected language.

- The degree of inflection may be higher or lower in a language.
- An inflected words will have a common root.
eg: playing
play
played
} common root from 'play'

* Stemming
- Method of normalization of words in NLP. It is a technique in which a set of words in a sentence are converted into a sequence to shorten its lookup.

* Lemmitization
- process of finding the leme of word, depends on its meaning & context

* Representation of document by calculating term frequency as Inverse document frequency.

Terminology : t - term (word)
d - document (set of words)
N - count of corpus.
Corpus - total document set.

a) Term frequency (TF)
- The weight of a term that occurs in a document is simply proportional to term frequency.

$$tf(t,d) = \frac{\text{count of } t \text{ in } d}{\text{no. of words in } d}$$

b) Inverse Document frequency (IDF)
- IDF is inverse of document frequency which measures informativeness of term t.

$$idf(t) = N / df$$

During query time, when a word which is not in vocab occur, df will be 0. So, we add 1 to the denomination.

TF-IDF is

$$idf(t) = \log(N/(df+1))$$

$$tf\text{-}idf(t,d) = tf(t,d) * \log\left(\frac{N}{df+1}\right)$$

Conclusion: Successfully performed preprocessing methods like Tokenization, POS Tagging, stop words removal, steming & Lemmetization.

Code:

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


pip install PyPDF2


# In[2]:


pip install python-docx


# In[102]:


# importing required modules
import PyPDF2

# creating a pdf file object
pdfFileObj = open('F:\example.pdf', 'rb')

# creating a pdf reader object
pdfReader = PyPDF2.PdfFileReader(pdfFileObj)

# printing number of pages in pdf file
print(pdfReader.numPages)

# creating a page object
pageObj = pdfReader.getPage(0)

# extracting text from page
print(pageObj.extractText())

# closing the pdf file object
pdfFileObj.close()


# In[103]:
```

```
# import docx NOT python-docx
import docx

# create an instance of a word document
doc = docx.Document()

# add a heading of level 0 (largest heading)
doc.add_heading('Heading for the document', 0)

# add a paragraph and store
# the object in a variable
doc_para = doc.add_paragraph('Your paragraph goes here, ')

# add a run i.e, style like
# bold, italic, underline, etc.
doc_para.add_run('hey there, bold here').bold = True
doc_para.add_run(', and ')
doc_para.add_run('these words are italic').italic = True

# add a page break to start a new page
doc.add_page_break()

# add a heading of level 2
doc.add_heading('Heading level 2', 2)

# pictures can also be added to our word document
# width is optional
doc.add_picture('F:\Vijay.jpg')

# now save the document to a location
doc.save('path_to_document')


# In[104]:


pip install nltk


# In[105]:


import nltk
nltk.download()
```

```
nltk.download('punkt')


# In[106]:


#Sentence Tokenization

sentence_data = "The First sentence is about Python. The Second: about Django. You can
learn Python,Django and Data Ananlysis here. "
nltk_tokens = nltk.sent_tokenize(sentence_data)
print (nltk_tokens)


# In[107]:


#Non English language Tokenization

german_tokenizer = nltk.data.load('tokenizers/punkt/german.pickle')
german_tokens=german_tokenizer.tokenize('Wie geht es Ihnen?  Gut, danke.')
print(german_tokens)


# In[108]:


#Word Tokenization

word_data = "It originated from the idea that there are readers who prefer learning new skills
from the comforts of their drawing rooms"
nltk_tokens = nltk.word_tokenize(word_data)
print (nltk_tokens)


# In[109]:


#Word Tokenization

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize

#Dummy text
```

```
txt = "He is a boy. "      "She is a girl"

word_tokens = word_tokenize(txt)

print(word_tokens)


# In[110]:


#Part of Speech (POS) tagging

import nltk
from nltk.tokenize import word_tokenize
text = word_tokenize("Hello welcome to the world of to learn Categorizing and POS Tagging
with NLTK and Python")
nltk.pos_tag(text)


# In[111]:


import nltk
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')


# In[112]:


from nltk.corpus import stopwords
print(stopwords.words('english'))


# In[113]:


#Stopwords removal from sentence

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

example_sent = """This is a sample sentence,
                  showing off the stop words filtration."""
```

```
stop_words = set(stopwords.words('english'))

word_tokens = word_tokenize(example_sent)

filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]

filtered_sentence = []

for w in word_tokens:
        if w not in stop_words:
        filtered_sentence.append(w)

print("Tokenized:", word_tokens)
print("Stop Words Removed:", filtered_sentence)


# In[114]:


#Stopwords from input file

import io
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# word_tokenize accepts
# a string as an input, not a file.
stop_words = set(stopwords.words('english'))
file1 = open("text.txt")

# Use this to read file content as a stream:
line = file1.read()
words = line.split()
for r in words:
        if not r in stop_words:
        appendFile = open('filteredtext.txt','a')
        appendFile.write(" "+r)
        appendFile.close()


# In[115]:
```

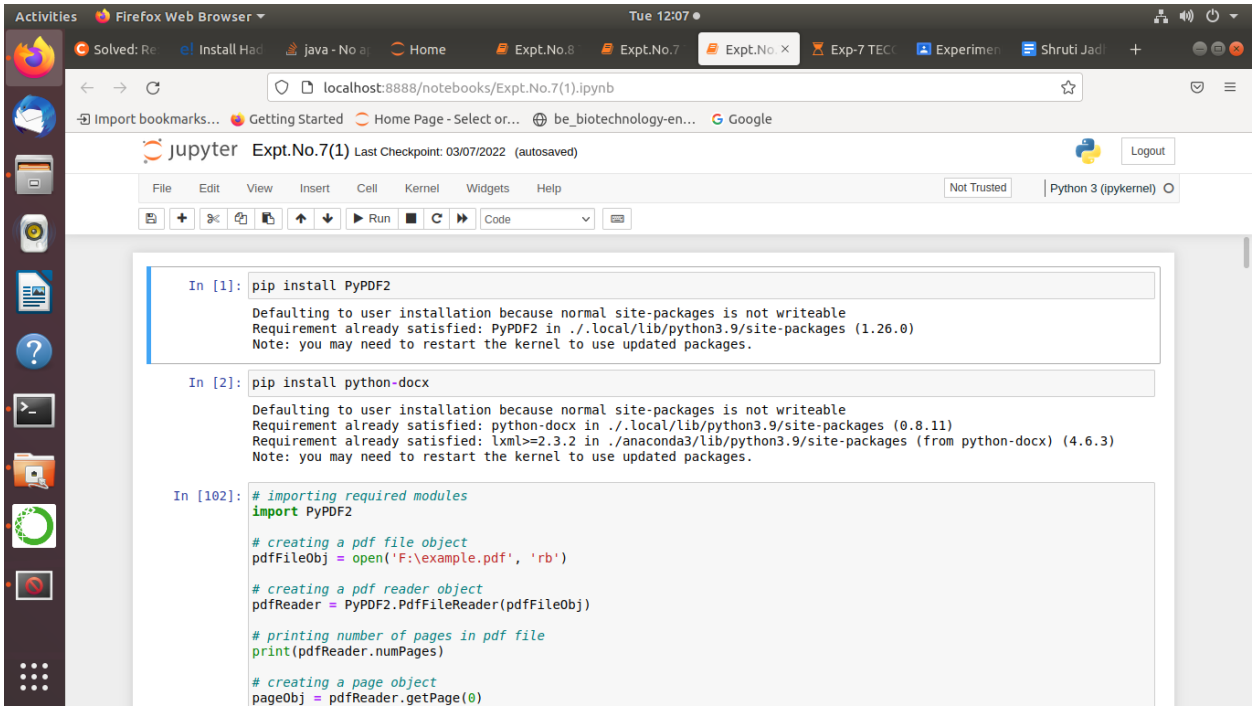
```
#Stemming

import nltk
from nltk.stem.porter import PorterStemmer
porter_stemmer = PorterStemmer()

word_data = "It vijaying vijayed vijays eats skills originated from the idea that there are readers
who prefer learning new skills from the comforts of their drawing rooms"
# First Word tokenization
nltk_tokens = nltk.word_tokenize(word_data)
#Next find the roots of the word
for w in nltk_tokens:
        print("Actual: %s  Stem: %s"  % (w,porter_stemmer.stem(w)))


# In[116]:


#Lemmatization

import nltk
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()

word_data = "It studies studying vijaying vijayed vijays skills originated from the idea that there
are readers who prefer learning new skills from the comforts of their drawing rooms"
nltk_tokens = nltk.word_tokenize(word_data)
for w in nltk_tokens:
        print("Actual: %s  Lemma: %s"  % (w,wordnet_lemmatizer.lemmatize(w)))


# In[117]:


#Expt.No.7 2nd Operation

import pandas as pd
import sklearn as sk
import math


# In[118]:
```

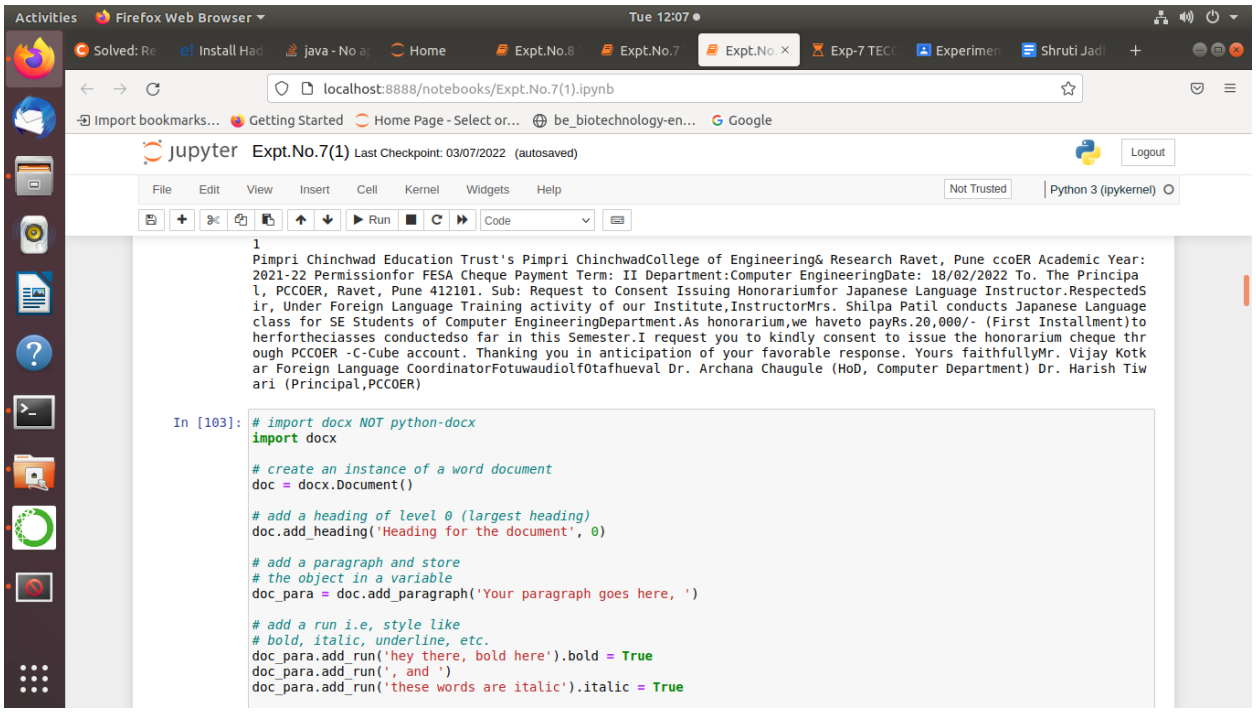
```
first_sentence = "Data Science is the best job of the 21st century"
second_sentence = "Machine learning is the key for data science"

#split so each word have their own string
first_sentence = first_sentence.split(" ")
second_sentence = second_sentence.split(" ")#join them to remove common duplicate words
total= set(first_sentence).union(set(second_sentence))
print(total)


# In[119]:


#count the words

wordDictA = dict.fromkeys(total, 0)
wordDictB = dict.fromkeys(total, 0)
for word in first_sentence:
        wordDictA[word]+=1

for word in second_sentence:
        wordDictB[word]+=1

pd.DataFrame([wordDictA, wordDictB])


# In[120]:


#Compute Term Frequency(TF)

def computeTF(wordDict, doc):
        tfDict = {}
        corpusCount = len(doc)
        for word, count in wordDict.items():
        tfDict[word] = count/float(corpusCount)
        return(tfDict)

#running our sentences through the tf function:
tfFirst = computeTF(wordDictA, first_sentence)
tfSecond = computeTF(wordDictB, second_sentence)

#Converting to dataframe for visualization
```

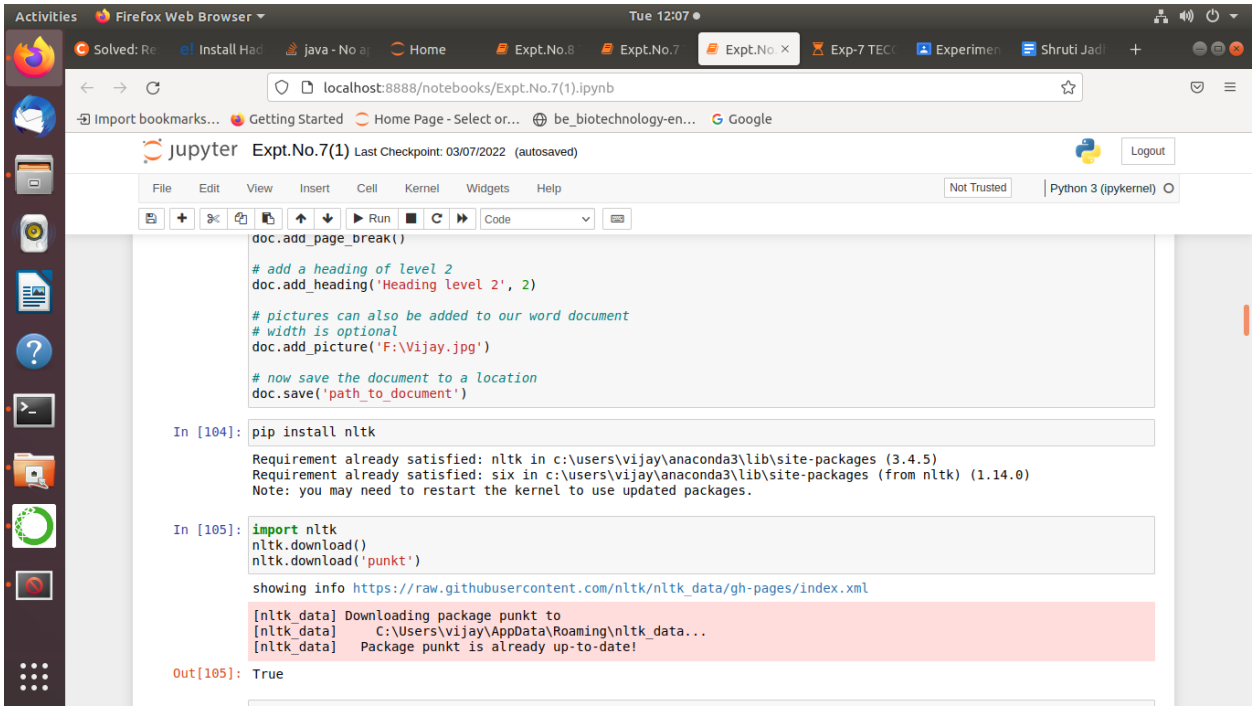
```
pd.DataFrame([tfFirst, tfSecond])


# In[121]:


#Compute Inverse Document Frequency(IDF)

def computeIDF(docList):
        idfDict = {}
        N = len(docList)

        idfDict = dict.fromkeys(docList[0].keys(), 0)
        for word, val in idfDict.items():
        idfDict[word] = math.log10(N / (float(val) + 1))

        return(idfDict)

#inputing our sentences in the log file
idfs = computeIDF([wordDictA, wordDictB])


# In[122]:


#Compute Term Frequency(TF) - Inverse Document Frequency(IDF)

def computeTFIDF(tfBow, idfs):
        tfidf = {}
        for word, val in tfBow.items():
        tfidf[word] = val*idfs[word]
        return(tfidf)

#running our two sentences through the IDF:
idfFirst = computeTFIDF(tfFirst, idfs)
idfSecond = computeTFIDF(tfSecond, idfs)

#putting it in a dataframe
pd.DataFrame([idfFirst, idfSecond])


# In[123]:
```

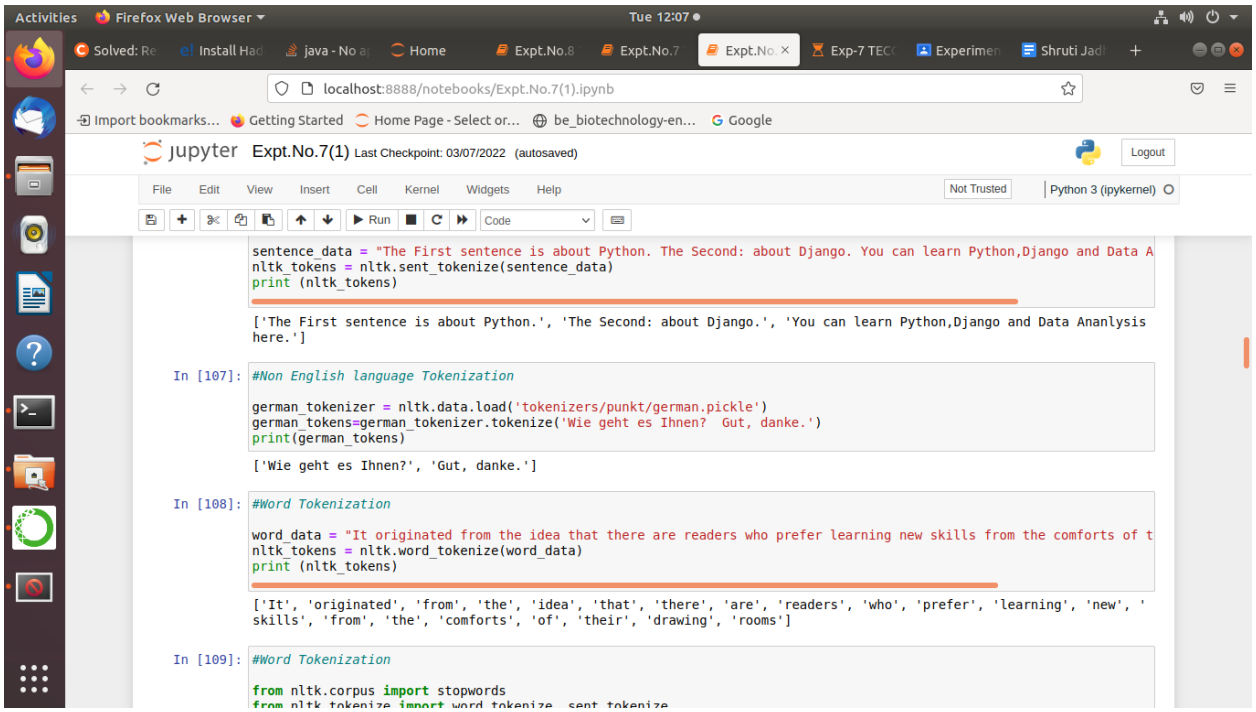
```
#Compute TF-IDF

#first step is to import the library
from sklearn.feature_extraction.text import TfidfVectorizer

#for the sentence, make sure all words are lowercase or you will run #into error. for simplicity, I just made the same sentence all #lowercase
firstV= "Data Science is the sexiest job of the 21st century"
secondV= "machine learning is the key for data science"

#calling the TfidfVectorizer
vectorize= TfidfVectorizer()

#fitting the model and passing our sentences right away:
response= vectorize.fit_transform([firstV, secondV])

print(response)


# In[ ]:
```

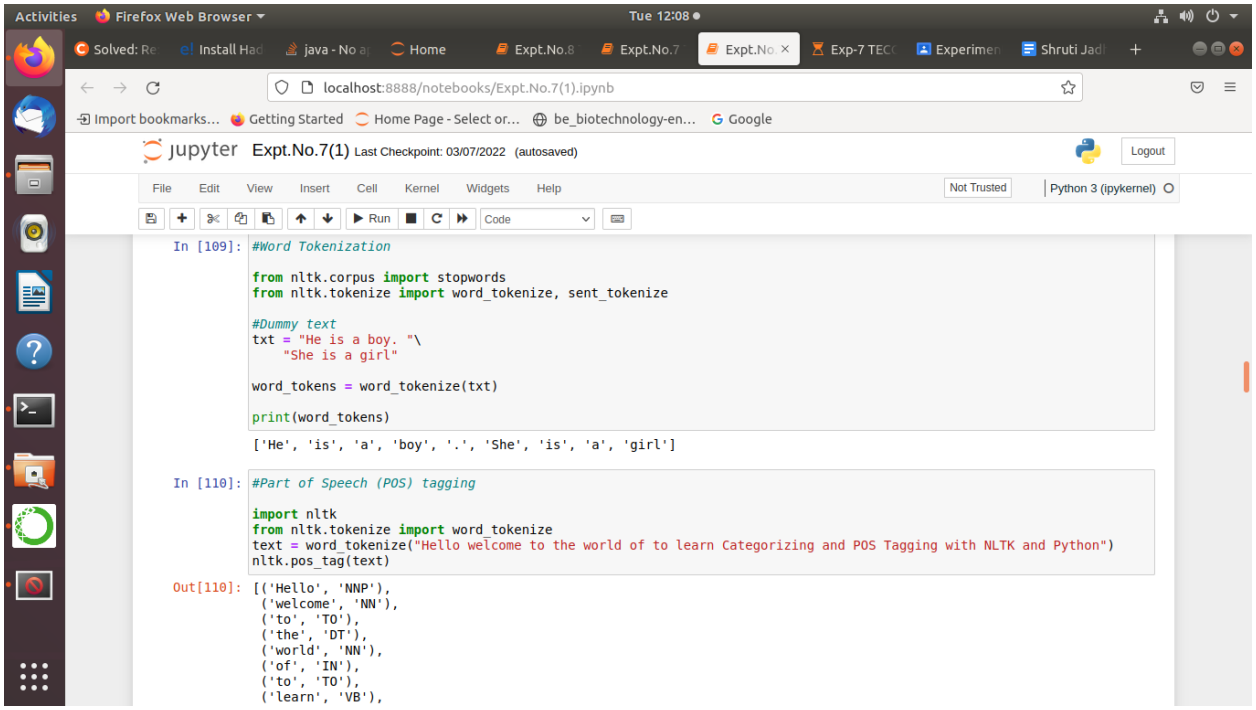
Output Screenshots:

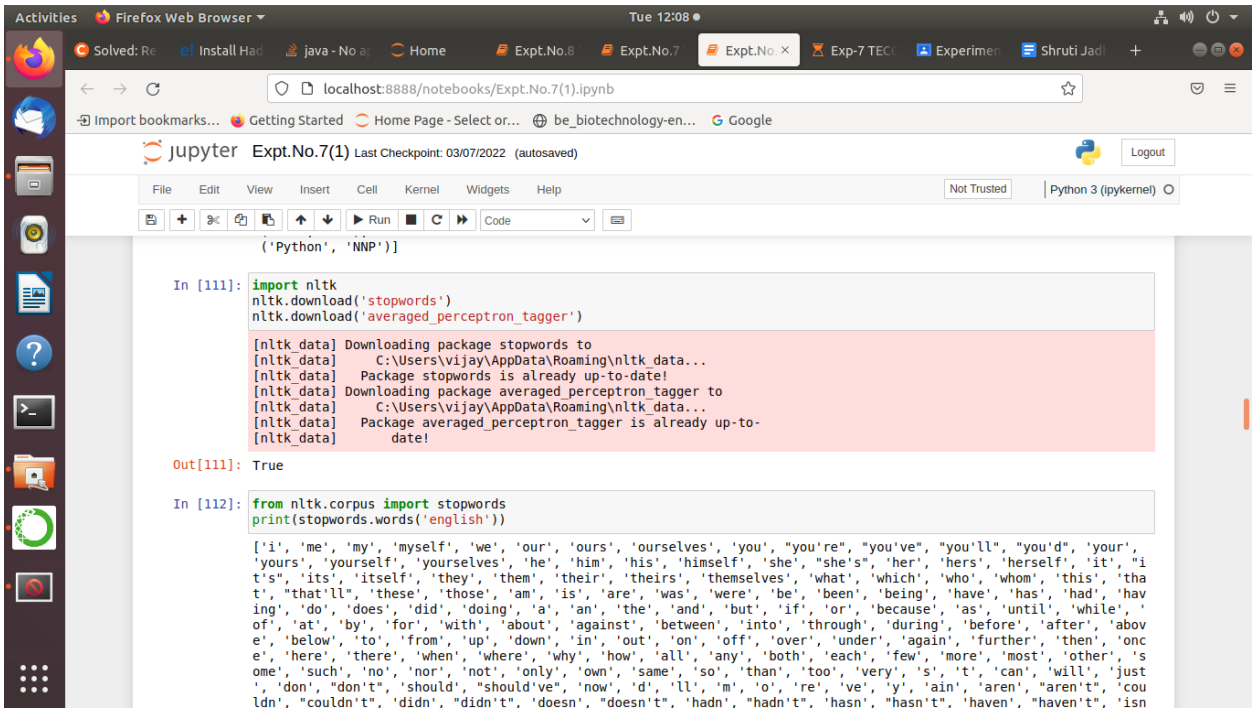

Activities
Firefox Web Browser
Tue 12:07
localhost:8888/notebooks/Expt.No.7(1).ipynb
jupyter Expt.No.7(1) Last Checkpoint: 03/07/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [1]: pip install PyPDF2
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: PyPDF2 in ./.local/lib/python3.9/site-packages (1.26.0)
Note: you may need to restart the kernel to use updated packages.
In [2]: pip install python-docx
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: python-docx in ./.local/lib/python3.9/site-packages (0.8.11)
Requirement already satisfied: lxml>=2.3.2 in ./anaconda3/lib/python3.9/site-packages (from python-docx) (4.6.3)
Note: you may need to restart the kernel to use updated packages.
In [102]:
# importing required modules
import PyPDF2
# creating a pdf file object
pdfFileObj = open('F:\example.pdf', 'rb')
# creating a pdf reader object
pdfReader = PyPDF2.PdfFileReader(pdfFileObj)
# printing number of pages in pdf file
print(pdfReader.numPages)
# creating a page object
pageObj = pdfReader.getPage(0)

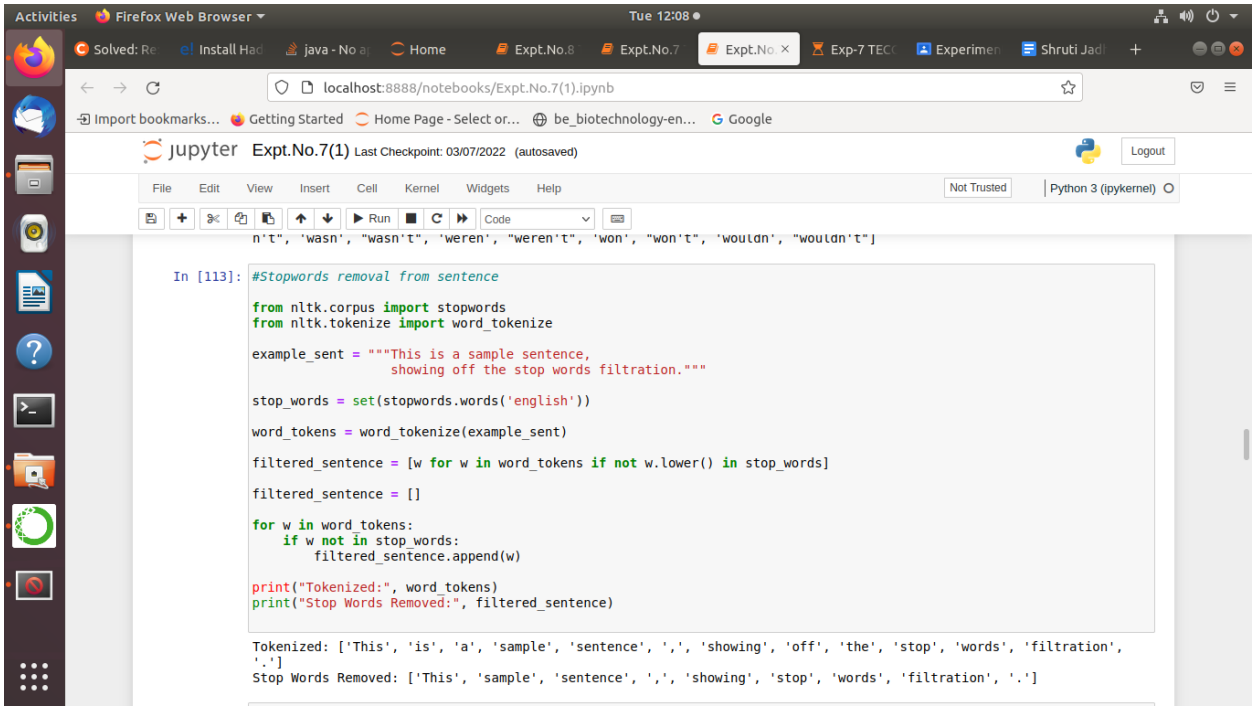

Activities
Firefox Web Browser
Tue 12:07
localhost:8888/notebooks/Expt.No.7(1).ipynb
jupyter Expt.No.7(1) Last Checkpoint: 03/07/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
1
Pimpri Chinchwad Education Trust's Pimpri ChinchwadCollege of Engineering& Research Ravet, Pune ccoER Academic Year: 2021-22 Permissionfor FESA Cheque Payment Term: II Department:Computer EngineeringDate: 18/02/2022 To. The Principal, PCCOER, Ravet, Pune 412101. Sub: Request to Consent Issuing Honorariumfor Japanese Language Instructor.RespectedSir, Under Foreign Language Training activity of our Institute,InstructorMrs. Shilpa Patil conducts Japanese Language class for SE Students of Computer EngineeringDepartment.As honorarium,we haveto payRs.20,000/- (First Installment)to herfortheciasses conductedso far in this Semester.I request you to kindly consent to issue the honorarium cheque through PCCOER -C-Cube account. Thanking you in anticipation of your favorable response. Yours faithfullyMr. Vijay Kotkar Foreign Language CoordinatorFotuwaudiolfOtafhueval Dr. Archana Chaugule (HoD, Computer Department) Dr. Harish Tiwari (Principal,PCCOER)
In [103]:
# import docx NOT python-docx
import docx
# create an instance of a word document
doc = docx.Document()
# add a heading of level 0 (largest heading)
doc.add_heading('Heading for the document', 0)
# add a paragraph and store
# the object in a variable
doc_para = doc.add_paragraph('Your paragraph goes here, ')
# add a run i.e, style like
# bold, italic, underline, etc.
doc_para.add_run('hey there, bold here').bold = True
doc_para.add_run(', and ')
doc_para.add_run('these words are italic').italic = True

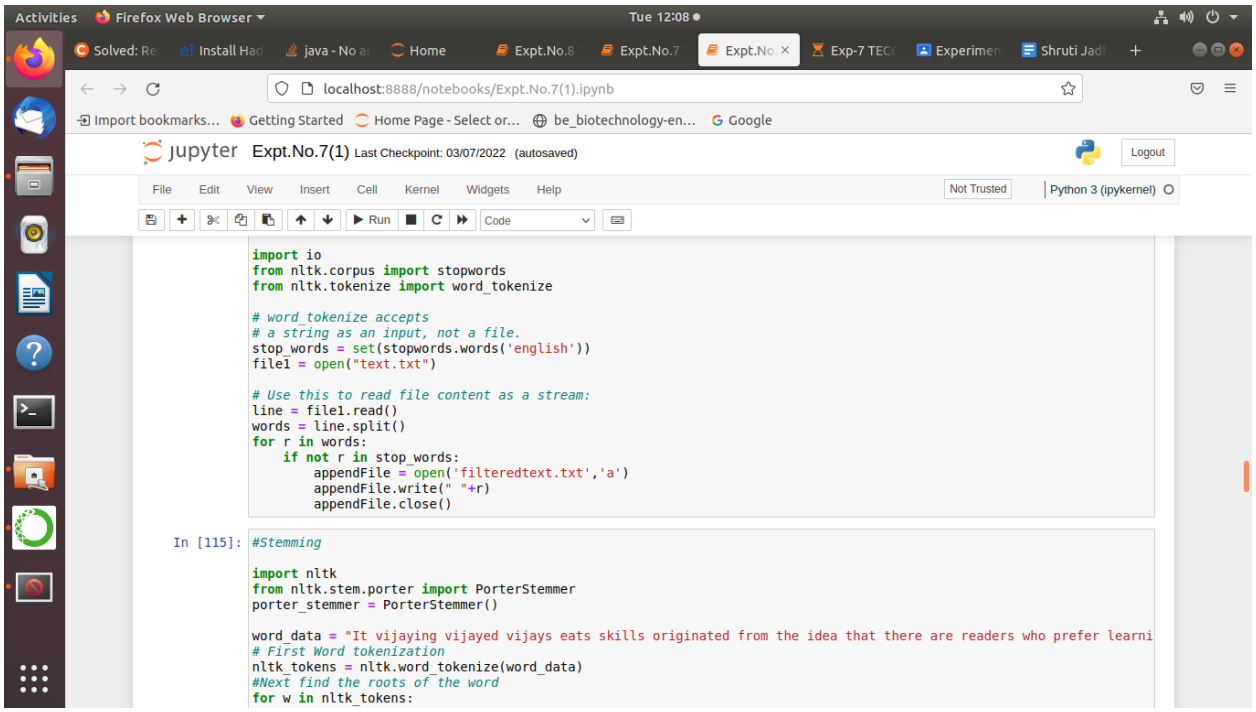

Activities
Firefox Web Browser
Tue 12:07
localhost:8888/notebooks/Expt.No.7(1).ipynb
jupyter Expt.No.7(1) Last Checkpoint: 03/07/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
doc.add_page_break()
# add a heading of level 2
doc.add_heading('Heading level 2', 2)
# pictures can also be added to our word document
# width is optional
doc.add_picture('F:\Vijay.jpg')
# now save the document to a location
doc.save('path_to_document')
In [104]:
pip install nltk
Requirement already satisfied: nltk in c:\users\vijay\anaconda3\lib\site-packages (3.4.5)
Requirement already satisfied: six in c:\users\vijay\anaconda3\lib\site-packages (from nltk) (1.14.0)
Note: you may need to restart the kernel to use updated packages.
In [105]:
import nltk
nltk.download()
nltk.download('punkt')
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
[nltk_data] Downloading package punkt to
[nltk_data] C:\Users\vijay\AppData\Roaming\nltk_data...
[nltk_data] Package punkt is already up-to-date!
Out[105]:
True

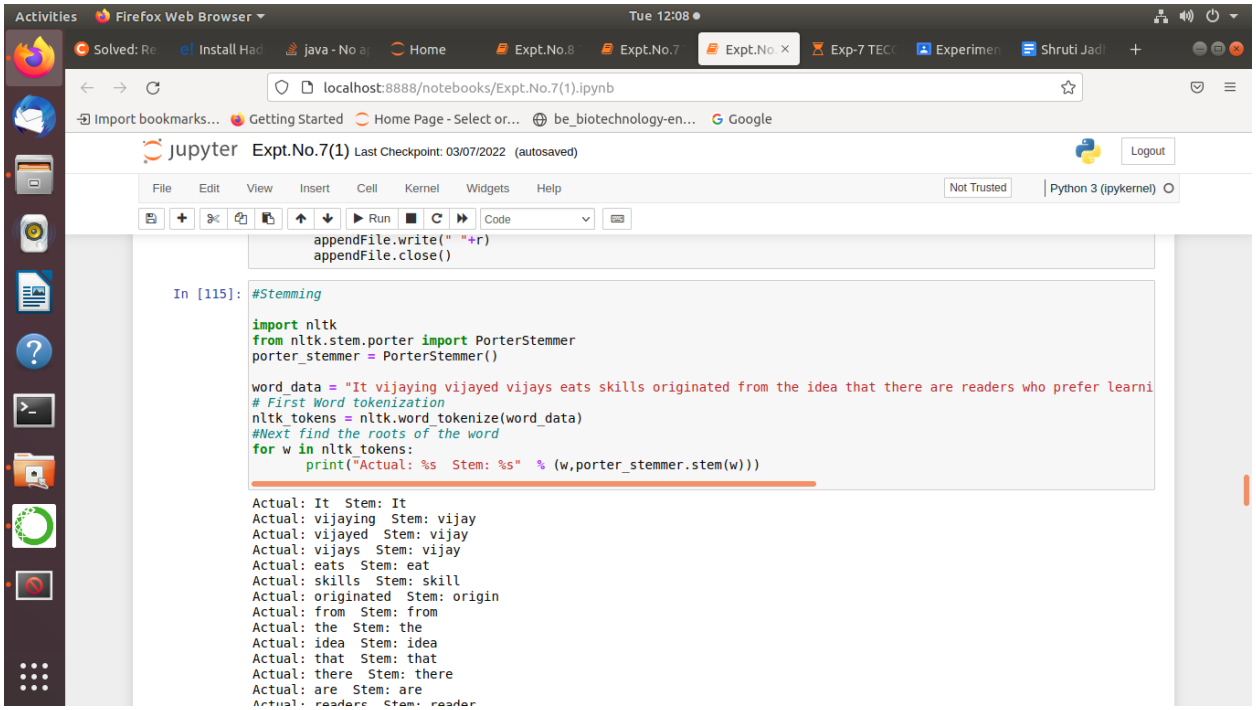

Activities
Firefox Web Browser
Tue 12:07
localhost:8888/notebooks/Expt.No.7(1).ipynb
jupyter Expt.No.7(1) Last Checkpoint: 03/07/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
sentence_data = "The First sentence is about Python. The Second: about Django. You can learn Python,Django and Data A
nltk_tokens = nltk.sent_tokenize(sentence_data)
print (nltk_tokens)
['The First sentence is about Python.', 'The Second: about Django.', 'You can learn Python,Django and Data Ananlysis here.']
In [107]:
#Non English language Tokenization
german_tokenizer = nltk.data.load('tokenizers/punkt/german.pickle')
german_tokens=german_tokenizer.tokenize('Wie geht es Ihnen? Gut, danke.')
print(german_tokens)
['Wie geht es Ihnen?', 'Gut, danke.']
In [108]:
#Word Tokenization
word_data = "It originated from the idea that there are readers who prefer learning new skills from the comforts of t
nltk_tokens = nltk.word_tokenize(word_data)
print (nltk_tokens)
['It', 'originated', 'from', 'the', 'idea', 'that', 'there', 'are', 'readers', 'who', 'prefer', 'learning', 'new', 'skills', 'from', 'the', 'comforts', 'of', 'their', 'drawing', 'rooms']
In [109]:
#Word Tokenization
from nltk.corpus import stopwords

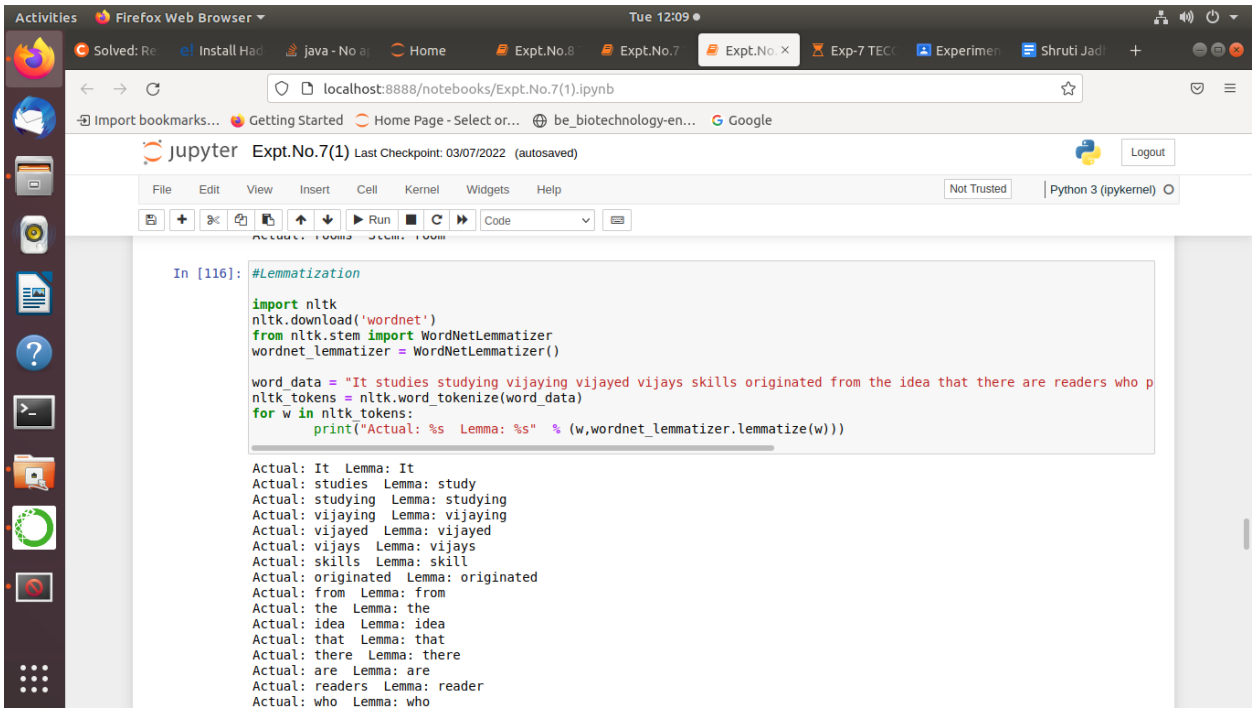

Activities
Firefox Web Browser
Tue 12:08
localhost:8888/notebooks/Expt.No.7(1).ipynb
jupyter
Expt.No.7(1) Last Checkpoint: 03/07/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [109]:
#Word Tokenization
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
#Dummy text
txt = "He is a boy. "\
"She is a girl"
word_tokens = word_tokenize(txt)
print(word_tokens)
['He', 'is', 'a', 'boy', '.', 'She', 'is', 'a', 'girl']
In [110]:
#Part of Speech (POS) tagging
import nltk
from nltk.tokenize import word_tokenize
text = word_tokenize("Hello welcome to the world of to learn Categorizing and POS Tagging with NLTK and Python")
nltk.pos_tag(text)
Out[110]:
[('Hello', 'NNP'),
('welcome', 'NN'),
('to', 'TO'),
('the', 'DT'),
('world', 'NN'),
('of', 'IN'),
('to', 'TO'),
('learn', 'VB'),

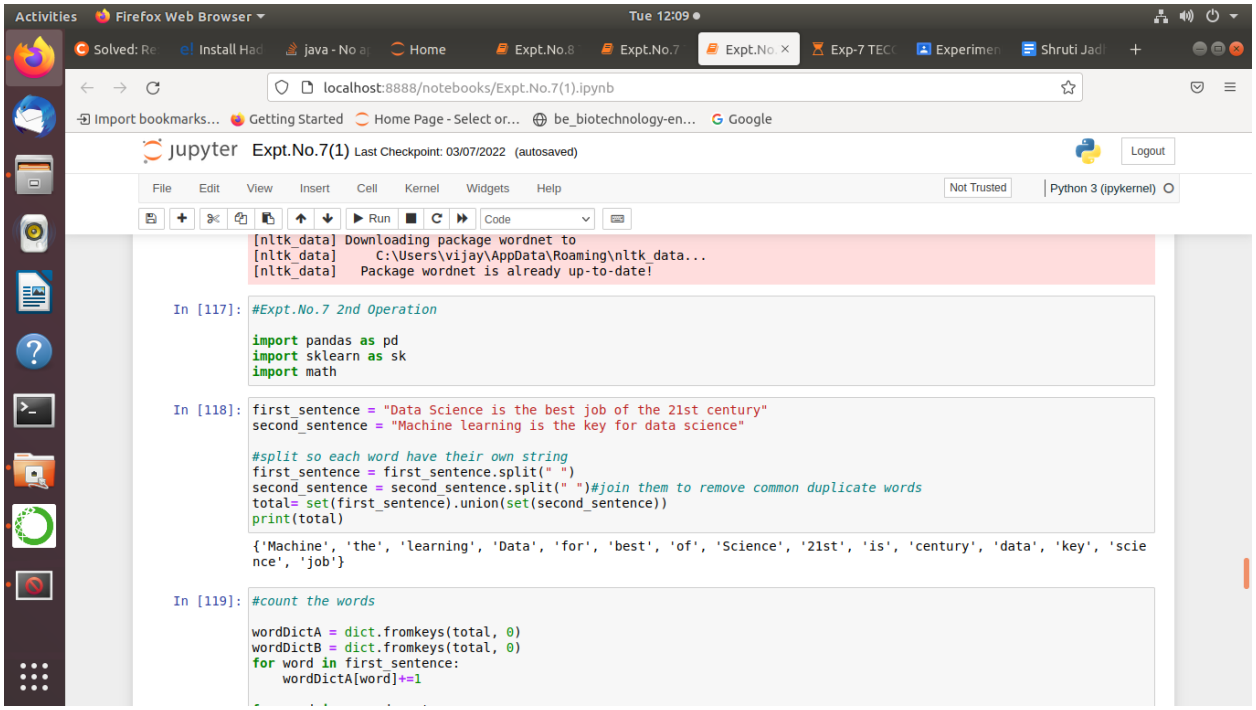

Activities
Firefox Web Browser
Tue 12:08
localhost:8888/notebooks/Expt.No.7(1).ipynb
jupyter
Expt.No.7(1) Last Checkpoint: 03/07/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
('Python', 'NNP')]
In [111]:
import nltk
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')
[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\vijay\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data] C:\Users\vijay\AppData\Roaming\nltk_data...
[nltk_data] Package averaged_perceptron_tagger is already up-to-
[nltk_data] date!
Out[111]:
True
In [112]:
from nltk.corpus import stopwords
print(stopwords.words('english'))
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your',
'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "i
t's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'tha
t', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'hav
ing', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', '
of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'abov
e', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'onc
e', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 's
ome', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just
', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'cou
ldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn

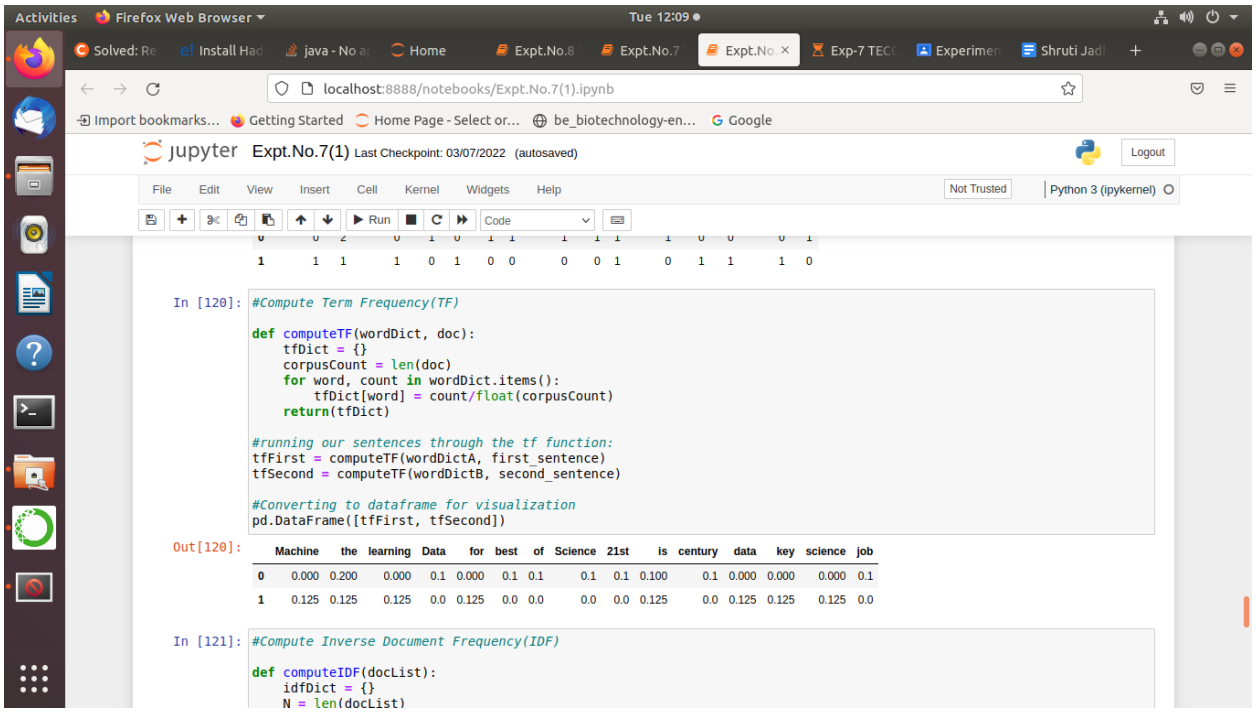

Activities
Firefox Web Browser
Tue 12:08
localhost:8888/notebooks/Expt.No.7(1).ipynb
jupyter
Expt.No.7(1)
Last Checkpoint: 03/07/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
In [113]:
#Stopwords removal from sentence
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
example_sent = """This is a sample sentence,
showing off the stop words filtration."""
stop_words = set(stopwords.words('english'))
word_tokens = word_tokenize(example_sent)
filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]
filtered_sentence = []
for w in word_tokens:
if w not in stop_words:
filtered_sentence.append(w)
print("Tokenized:", word_tokens)
print("Stop Words Removed:", filtered_sentence)
Tokenized: ['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off', 'the', 'stop', 'words', 'filtration',
'.']
Stop Words Removed: ['This', 'sample', 'sentence', ',', 'showing', 'stop', 'words', 'filtration', '.']

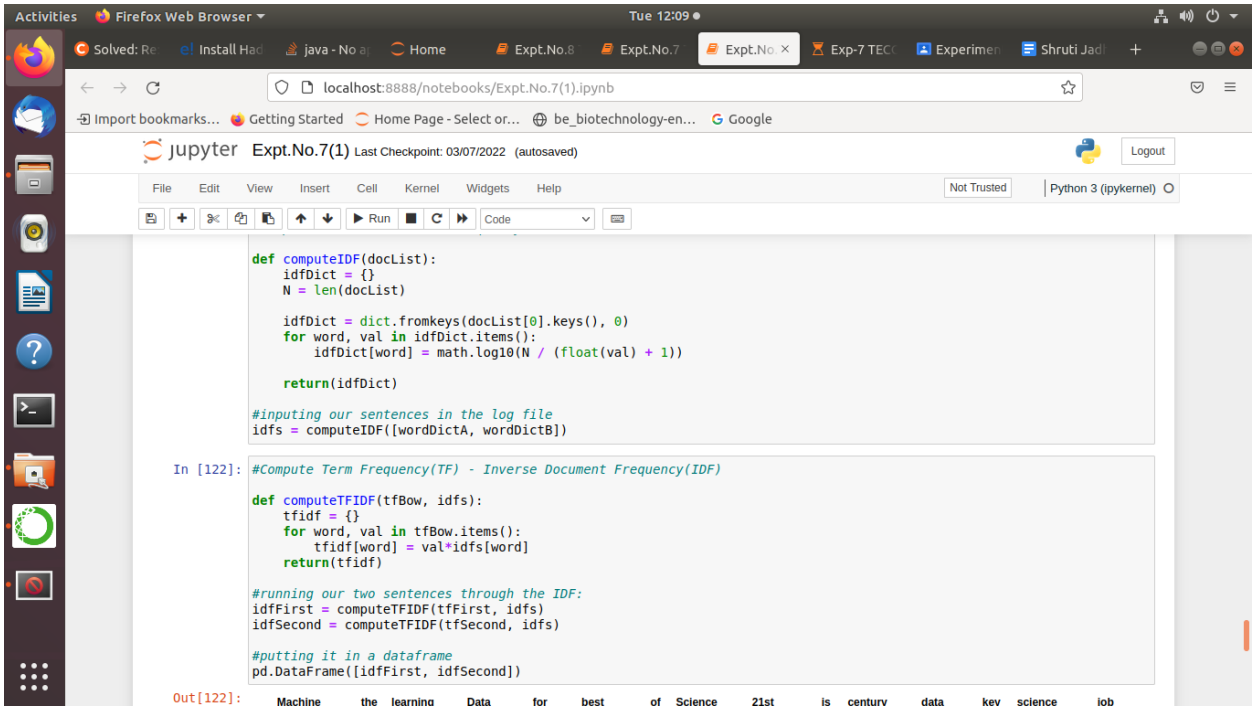

Activities
Firefox Web Browser
Tue 12:08
localhost:8888/notebooks/Expt.No.7(1).ipynb
jupyter
Expt.No.7(1)
Last Checkpoint: 03/07/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
import io
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
# word_tokenize accepts
# a string as an input, not a file.
stop_words = set(stopwords.words('english'))
file1 = open("text.txt")
# Use this to read file content as a stream:
line = file1.read()
words = line.split()
for r in words:
if not r in stop_words:
appendFile = open('filteredtext.txt','a')
appendFile.write(" "+r)
appendFile.close()
In [115]:
#Stemming
import nltk
from nltk.stem.porter import PorterStemmer
porter_stemmer = PorterStemmer()
word_data = "It vijaying vijayed vijays eats skills originated from the idea that there are readers who prefer learni
# First Word tokenization
nltk_tokens = nltk.word_tokenize(word_data)
#Next find the roots of the word
for w in nltk_tokens:

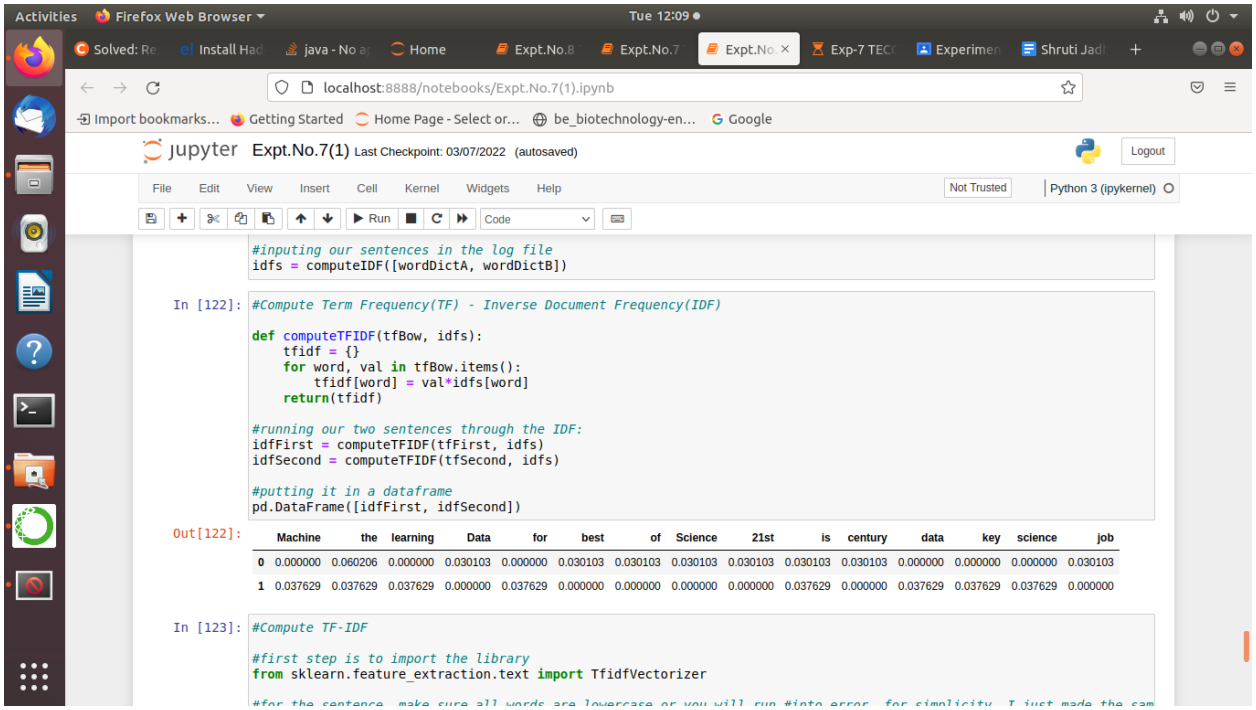

Activities
Firefox Web Browser
Tue 12:08
localhost:8888/notebooks/Expt.No.7(1).ipynb
jupyter
Expt.No.7(1) Last Checkpoint: 03/07/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
appendFile.write(" "+r)
appendFile.close()
In [115]:
#Stemming
import nltk
from nltk.stem.porter import PorterStemmer
porter_stemmer = PorterStemmer()
word_data = "It vijaying vijayed vijays eats skills originated from the idea that there are readers who prefer learni
# First Word tokenization
nltk_tokens = nltk.word_tokenize(word_data)
#Next find the roots of the word
for w in nltk_tokens:
print("Actual: %s Stem: %s" % (w,porter_stemmer.stem(w)))
Actual: It Stem: It
Actual: vijaying Stem: vijay
Actual: vijayed Stem: vijay
Actual: vijays Stem: vijay
Actual: eats Stem: eat
Actual: skills Stem: skill
Actual: originated Stem: origin
Actual: from Stem: from
Actual: the Stem: the
Actual: idea Stem: idea
Actual: that Stem: that
Actual: there Stem: there
Actual: are Stem: are

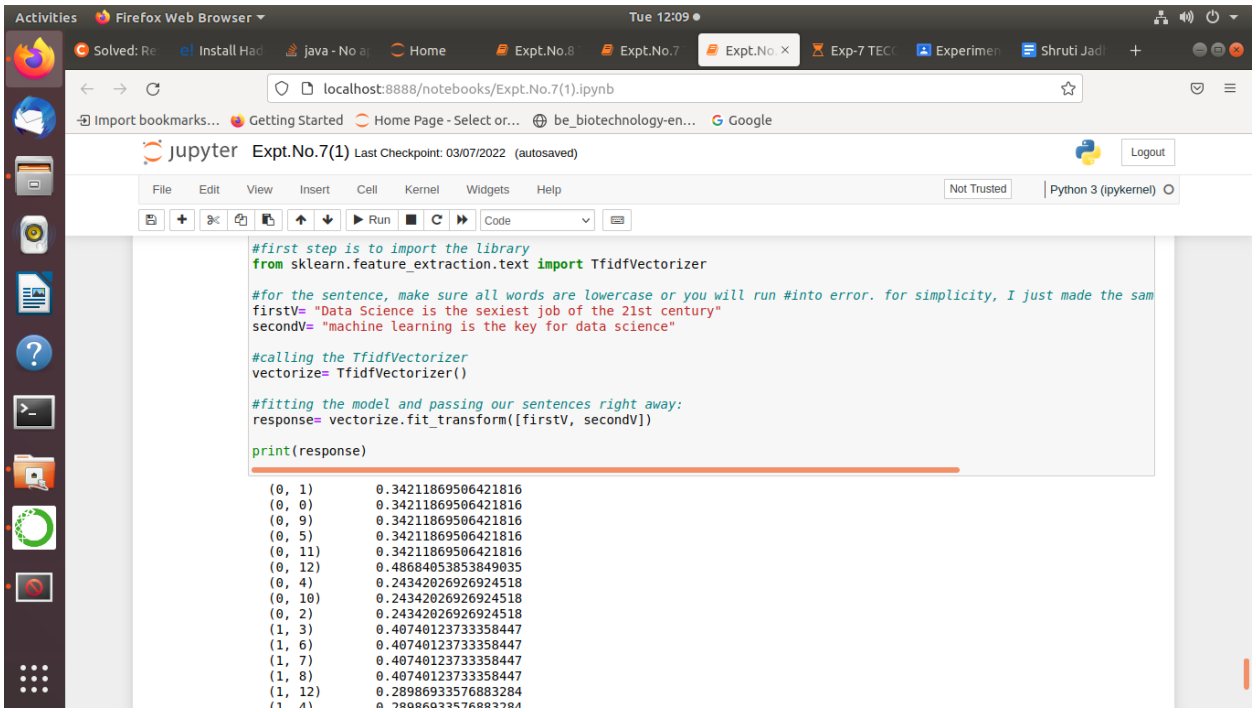

Activities
Firefox Web Browser
Tue 12:09
localhost:8888/notebooks/Expt.No.7(1).ipynb
jupyter
Expt.No.7(1) Last Checkpoint: 03/07/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [116]:
#Lemmatization
import nltk
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
word_data = "It studies studying vijaying vijayed vijays skills originated from the idea that there are readers who p
nltk_tokens = nltk.word_tokenize(word_data)
for w in nltk_tokens:
print("Actual: %s Lemma: %s" % (w,wordnet_lemmatizer.lemmatize(w)))
Actual: It Lemma: It
Actual: studies Lemma: study
Actual: studying Lemma: studying
Actual: vijaying Lemma: vijaying
Actual: vijayed Lemma: vijayed
Actual: vijays Lemma: vijays
Actual: skills Lemma: skill
Actual: originated Lemma: originated
Actual: from Lemma: from
Actual: the Lemma: the
Actual: idea Lemma: idea
Actual: that Lemma: that
Actual: there Lemma: there
Actual: are Lemma: are
Actual: readers Lemma: reader
Actual: who Lemma: who


Activities
Firefox Web Browser
Tue 12:09
localhost:8888/notebooks/Expt.No.7(1).ipynb
jupyter Expt.No.7(1) Last Checkpoint: 03/07/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
[nltk_data] Downloading package wordnet to
[nltk_data] C:\Users\vijay\AppData\Roaming\nltk_data...
[nltk_data] Package wordnet is already up-to-date!
In [117]:
#Expt.No.7 2nd Operation
import pandas as pd
import sklearn as sk
import math
In [118]:
first_sentence = "Data Science is the best job of the 21st century"
second_sentence = "Machine learning is the key for data science"
#split so each word have their own string
first_sentence = first_sentence.split(" ")
second_sentence = second_sentence.split(" ")#join them to remove common duplicate words
total= set(first_sentence).union(set(second_sentence))
print(total)
{'Machine', 'the', 'learning', 'Data', 'for', 'best', 'of', 'Science', '21st', 'is', 'century', 'data', 'key', 'scie
nce', 'job'}
In [119]:
#count the words
wordDictA = dict.fromkeys(total, 0)
wordDictB = dict.fromkeys(total, 0)
for word in first_sentence:
wordDictA[word]+=1


Activities
Firefox Web Browser
Tue 12:09
localhost:8888/notebooks/Expt.No.7(1).ipynb
jupyter Expt.No.7(1) Last Checkpoint: 03/07/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
1 1 1 1 0 1 0 0 0 0 1 0 1 1 1 0
In [120]:
#Compute Term Frequency(TF)
def computeTF(wordDict, doc):
tfDict = {}
corpusCount = len(doc)
for word, count in wordDict.items():
tfDict[word] = count/float(corpusCount)
return(tfDict)
#running our sentences through the tf function:
tfFirst = computeTF(wordDictA, first_sentence)
tfSecond = computeTF(wordDictB, second_sentence)
#Converting to dataframe for visualization
pd.DataFrame([tfFirst, tfSecond])
Out[120]:
Machine the learning Data for best of Science 21st is century data key science job
0 0.000 0.200 0.000 0.1 0.000 0.1 0.1 0.1 0.1 0.100 0.1 0.000 0.000 0.000 0.1
1 0.125 0.125 0.125 0.0 0.125 0.0 0.0 0.0 0.0 0.125 0.0 0.125 0.125 0.125 0.0
In [121]:
#Compute Inverse Document Frequency(IDF)
def computeIDF(docList):
idfDict = {}
N = len(docList)


Activities
Firefox Web Browser
Tue 12:09
localhost:8888/notebooks/Expt.No.7(1).ipynb
jupyter
Expt.No.7(1)
Last Checkpoint: 03/07/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
def computeIDF(docList):
idfDict = {}
N = len(docList)
idfDict = dict.fromkeys(docList[0].keys(), 0)
for word, val in idfDict.items():
idfDict[word] = math.log10(N / (float(val) + 1))
return(idfDict)
#inputing our sentences in the log file
idfs = computeIDF([wordDictA, wordDictB])
In [122]:
#Compute Term Frequency(TF) - Inverse Document Frequency(IDF)
def computeTFIDF(tfBow, idfs):
tfidf = {}
for word, val in tfBow.items():
tfidf[word] = val*idfs[word]
return(tfidf)
#running our two sentences through the IDF:
idfFirst = computeTFIDF(tfFirst, idfs)
idfSecond = computeTFIDF(tfSecond, idfs)
#putting it in a dataframe
pd.DataFrame([idfFirst, idfSecond])
Out[122]:
Machine the learning Data for best of Science 21st is century data key science job


Activities
Firefox Web Browser
Tue 12:09
localhost:8888/notebooks/Expt.No.7(1).ipynb
jupyter Expt.No.7(1) Last Checkpoint: 03/07/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
#inputing our sentences in the log file
idfs = computeIDF([wordDictA, wordDictB])
In [122]:
#Compute Term Frequency(TF) - Inverse Document Frequency(IDF)
def computeTFIDF(tfBow, idfs):
tfidf = {}
for word, val in tfBow.items():
tfidf[word] = val*idfs[word]
return(tfidf)
#running our two sentences through the IDF:
idfFirst = computeTFIDF(tfFirst, idfs)
idfSecond = computeTFIDF(tfSecond, idfs)
#putting it in a dataframe
pd.DataFrame([idfFirst, idfSecond])
Out[122]:
Machine the learning Data for best of Science 21st is century data key science job
0 0.000000 0.060206 0.000000 0.030103 0.000000 0.030103 0.030103 0.030103 0.030103 0.030103 0.030103 0.030103 0.000000 0.000000 0.000000 0.030103
1 0.037629 0.037629 0.037629 0.000000 0.037629 0.000000 0.000000 0.000000 0.000000 0.037629 0.000000 0.037629 0.037629 0.037629 0.000000
In [123]:
#Compute TF-IDF
#first step is to import the library
from sklearn.feature_extraction.text import TfidfVectorizer


Activities
Firefox Web Browser
Tue 12:09
localhost:8888/notebooks/Expt.No.7(1).ipynb
jupyter Expt.No.7(1) Last Checkpoint: 03/07/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
#first step is to import the library
from sklearn.feature_extraction.text import TfidfVectorizer
#for the sentence, make sure all words are lowercase or you will run #into error. for simplicity, I just made the sam
firstV= "Data Science is the sexiest job of the 21st century"
secondV= "machine learning is the key for data science"
#calling the TfidfVectorizer
vectorize= TfidfVectorizer()
#fitting the model and passing our sentences right away:
response= vectorize.fit_transform([firstV, secondV])
print(response)
(0, 1) 0.34211869506421816
(0, 0) 0.34211869506421816
(0, 9) 0.34211869506421816
(0, 5) 0.34211869506421816
(0, 11) 0.34211869506421816
(0, 12) 0.48684053853849035
(0, 4) 0.24342026926924518
(0, 10) 0.24342026926924518
(0, 2) 0.24342026926924518
(1, 3) 0.40740123733358447
(1, 6) 0.40740123733358447
(1, 7) 0.40740123733358447
(1, 8) 0.40740123733358447
(1, 12) 0.28986933576883284