```
nltk.download()
```

```
    |   Unzipping grammars/spanish_grammars.zip.
    | Downloading package state_union to /root/nltk_data...
    |   Unzipping corpora/state_union.zip.
    | Downloading package stopwords to /root/nltk_data...
    |   Unzipping corpora/stopwords.zip.
    | Downloading package subjectivity to /root/nltk_data...
    |   Unzipping corpora/subjectivity.zip.
    | Downloading package swadesh to /root/nltk_data...
    |   Unzipping corpora/swadesh.zip.
    | Downloading package switchboard to /root/nltk_data...
    |   Unzipping corpora/switchboard.zip.
    | Downloading package tagsets to /root/nltk_data...
    |   Unzipping help/tagsets.zip.
    | Downloading package timit to /root/nltk_data...
    |   Unzipping corpora/timit.zip.
    | Downloading package toolbox to /root/nltk_data...
    |   Unzipping corpora/toolbox.zip.
    | Downloading package treebank to /root/nltk_data...
    |   Unzipping corpora/treebank.zip.
    | Downloading package twitter_samples to /root/nltk_data...
    |   Unzipping corpora/twitter_samples.zip.
    | Downloading package udhr to /root/nltk_data...
    |   Unzipping corpora/udhr.zip.
    | Downloading package udhr2 to /root/nltk_data...
    |   Unzipping corpora/udhr2.zip.
    | Downloading package unicode_samples to /root/nltk_data...
    |   Unzipping corpora/unicode_samples.zip.
    | Downloading package universal_tagset to /root/nltk_data...
    |   Unzipping taggers/universal_tagset.zip.
    | Downloading package universal_treebanks_v20 to
    |     /root/nltk_data...
    | Downloading package vader_lexicon to /root/nltk_data...
    | Downloading package verbnet to /root/nltk_data...
    |   Unzipping corpora/verbnet.zip.
    | Downloading package verbnet3 to /root/nltk_data...
    |   Unzipping corpora/verbnet3.zip.
    | Downloading package webtext to /root/nltk_data...
    |   Unzipping corpora/webtext.zip.
    | Downloading package wmt15_eval to /root/nltk_data...
    |   Unzipping models/wmt15_eval.zip.
    | Downloading package word2vec_sample to /root/nltk_data...
    |   Unzipping models/word2vec_sample.zip.
    | Downloading package wordnet to /root/nltk_data...
    | Downloading package wordnet2021 to /root/nltk_data...
    | Downloading package wordnet31 to /root/nltk_data...
    | Downloading package wordnet_ic to /root/nltk_data...
    |   Unzipping corpora/wordnet_ic.zip.
    | Downloading package words to /root/nltk_data...
    |   Unzipping corpora/words.zip.
    | Downloading package ycoe to /root/nltk_data...
    |   Unzipping corpora/ycoe.zip.
    |
  Done downloading collection all

---------------------------------------------------------------------------
    d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
---------------------------------------------------------------------------
```

```
    Downloader> q
    True
```

```
import csv
import re
import nltk
import string
import math
import numpy as np
import sklearn
from nltk.corpus import stopwords
from pprint import pprint
from termcolor import colored
from sklearn.metrics import precision_recall_fscore_support
```

```
#tempaltes for printing
printReverseGreen = lambda x : print(colored(x,'green',attrs=['reverse','bold']
printReverseRed = lambda x : print(colored(x,'red',attrs=['reverse','bold']))
```

```
def printDataset(data,row):
    for i in range(row):
        printReverseRed('Row '+str(i))
        for attr in data[i]:
            print(attr)
```

```
def createDataset(fname):
    #reading dataset
    lol=list(csv.reader(open(fname), delimiter=':'))
    tarClass=list(set([e[0] for e in lol]))
    tarClass.sort()
    tarClass_map=dict(zip(tarClass,range(len(tarClass))))
    printReverseGreen("Mapping for target class")
    print(tarClass_map)
    #Mapping target class to numeric value
    data=[[tarClass_map[row[0]]," ".join(row[1].split()[1:]),None,None,None,None
    #removing punctuation and numeric value from question and generating vocab
    vocab=[]
    for row in data:
        questn=[word.lower() for word in re.sub('[^a-zA-Z]', ' ',row[1]).split(
        row[2]=questn
        row[3]=len(questn)
        vocab.extend(questn)
    #creating ngram
    ngram=list(nltk.ngrams(vocab,1))
    #counting a ngram
    ngramFreq={}
    for e in ngram:
        if " ".join(e) in ngramFreq:ngramFreq[" ".join(e)]+=1
        else:ngramFreq[" ".join(e)]=1
    #sorting in decreasing order
    ngramFreq=sorted(ngramFreq.items(),key=lambda x:x[1],reverse=True)
    #selecting top 500 from sorted list
    moFreq500=[e[0] for e in ngramFreq[:500]]
    #converting ngramFreq again back to dictionary from list of tuples
```

```
    #converting ngramFreq again back to dictionary from list of tuples
    ngramFreq=dict(ngramFreq)
    printReverseGreen("Datatset Attribute")
    print("-"*120)
    print("| Targte class | Raw Question | Question After preprocessing | lengt
    print("-"*120)
    printReverseGreen("first 5 rows from dataset after processing the question"
    printDataset(data,5)
    #extracting lexical and syntactic data
    for row in data:
        lexical=[]
        syntax=[]
        for word in row[2]:
            if word in moFreq500:
                lexical.append(word)
                syntax.append(nltk.pos_tag([word])[0][1])
        row[4]=lexical
        row[5]=syntax
    printReverseGreen("first 5 rows from dataset after extracting lexical and sy
    printDataset(data,5)
    return tarClass_map,ngramFreq,moFreq500,data
```

```
def probWordGivenTarClass(data,word,column_no,tarClass,vocab_length):
    tarClass_rows=[questn for questn in data if questn[0]==tarClass]
    count_word_given_tarClass=sum([row[column_no].count(word) for row in tarCla
    return (count_word_given_tarClass+1)/(len(tarClass_rows)+vocab_length)
```

```
def probWordsGivenTarClasses(tarClasses,column_no,data,words):
    map_probWordGivenTarClass={}
    for word in words:
        map={}
        for tarClass in tarClasses:
            map[tarClass]=probWordGivenTarClass(data,word,column_no,tarClass,le
        map_probWordGivenTarClass[word]=map
    return map_probWordGivenTarClass
```

```
def trainCalculateColProbs(data,mpwgtc,mppgtc):
    for row in data:
        row[4]=np.product([mpwgtc[word][row[0]] for word in row[4]])
        row[5]=np.product([mppgtc[word][row[0]] for word in row[5]])
    printReverseGreen("first 5 rows from training dataset after probability cal
    printDataset(data,5)
    return data
```

```
def testCalculateColProbs(data,mpwgtc,mppgtc):
    for row in data:
        row[4]=np.product([max(mpwgtc[word].values()) if word in mpwgtc else 1
        row[5]=np.product([max(mppgtc[word].values()) if word in mppgtc else 1
    printReverseGreen("first 5 rows from testing dataset after probability calc
    printDataset(data,5)
    return data

```

```
def bestSplit(data,algorithm):
    transpose_data=np.array(data).T.tolist()
    impurity_attribute=[]
    #print(len(data))
    for i in range(1,len(transpose_data)):
        data=transpose_data[i]
        data_target=list(zip(data,transpose_data[0]))
        unique_data=list(set(data))
        unique_data.sort()
        #print("unique_data",unique_data)
        split_position=[(unique_data[i]+unique_data[i+1])/2 for i in range(len(u
        if len(unique_data)==1:
            split_position=[unique_data[0]]
        #print("split positions",split_position)
        impurity=[]
        minGiniSplit=1
        for position in split_position:
            le=[]
            le_target_frequency={}
            gt=[]
            gt_target_frequency={}
            for row in data_target:
                if row[0]<=position:le.append(row)
                else:gt.append(row)
            for target in tarClass_map.values():
                le_target_frequency[target]=0
                gt_target_frequency[target]=0
            for row in le:
                le_target_frequency[row[1]]+=1
            for row in gt:
                gt_target_frequency[row[1]]+=1
            le.append(None)
            gt.append(None)
            #gini
            if algorithm=="gini":
                algorithm_le=1-sum([(e[1]/len(le))**2 for e in le_target_freque
                algorithm_gt=1-sum([(e[1]/len(gt))**2 for e in gt_target_freque
                algorithm_split=(len(le)/(len(le)+len(gt)))*algorithm_le+(len(g
            #entropy
            if algorithm=="entropy":
                algorithm_le=sum([-1*(e[1]/len(le))*math.log2((e[1]+1)/len(le))
                algorithm_gt=sum([-1*(e[1]/len(gt))*math.log2((e[1]+1)/len(gt))
                algorithm_split=(len(le)/(len(le)+len(gt)))*algorithm_le+(len(g
            #classification error
            if algorithm=="ce":
                algorithm_le=1-max([e[1]/len(le) for e in le_target_frequency.i
                algorithm_gt=1-max([e[1]/len(gt) for e in gt_target_frequency.i
                algorithm_split=(len(le)/(len(le)+len(gt)))*algorithm_le+(len(g
            tarClass=None
            '''
            if int(algorithm_split)==0:
                for e in tarClass_map.values():
                    if le_target_frequency[e]+gt_target_frequency[e]==len(data_
                        tarClass=e
```

```
            '''
            target_frequency=[[e,le_target_frequency[e]+gt_target_frequency[e]]
            tarClass=sorted(target_frequency,key=lambda x:x[1],reverse=True)[0]
            impurity.append([i,position,algorithm_split,tarClass])
        #print("impurity",impurity)
        #print("tarClass",tarClass)
        impurity_attribute.append(sorted(impurity,key=lambda x:x[2])[0])
        #print("impurity attribute",impurity_attribute)
        #if tarClass:break
    return sorted(impurity_attribute,key=lambda x:x[2])[0]
```

```
class node:
    def __init__(self,data):
        self.attr=None
        self.val=None
        self.impurity=None
        self.data=data
        self.left=None
        self.right=None
        self.tarClass=None
```

```
def bulidTree(train_data,algorithm):
    start=node(train_data)
    level=[start]
    cnt=0
    while level:
        cur=level[0]
        #if len(cur.data)>1:
        decision=bestSplit(cur.data,algorithm)
        cur.attr=decision[0]
        cur.val=decision[1]
        cur.impurity=decision[2]
        cur.tarClass=decision[3]
        #if len(cur.data)==1:
        #cur.tarClass=cur.data[0][0]
        #cur.impurity=0
        if cur.impurity!=0:
            left=[row for row in cur.data if row[cur.attr]<=cur.val]
            right=[row for row in cur.data if row[cur.attr]>cur.val]
            if left:
                cur.left=node(left)
                level.append(cur.left)
            if right:
                cur.right=node(right)
                level.append(cur.right)
        level.pop(0)
        cnt+=1
        if cnt==20000:
           break
    return start
```

```
def decisionTreeClassifier(start,attributes):
```

```
    cur=start
    tarClass=None
    while cur and cur.attr:
        #print("current attribute",cur.attr)
        if attributes[cur.attr-1]<=cur.val:
            tarClass=cur.tarClass
            cur=cur.left
        else:
            tarClass=cur.tarClass
            cur=cur.right
    return tarClass
```

```
#creation of train dataset
tarClass_map,train_ngramFreq,train_moFreq500,train_data=createDataset("dt_train
```

```
None
Row 2
1
How can I find a list of celebrities ' real names ?
['how', 'can', 'find', 'list', 'of', 'celebrities', 'real', 'names']
8
None
None
Row 3
2
What fowl grabs the spotlight after the Chinese Year of the Monkey ?
['what', 'fowl', 'grabs', 'the', 'spotlight', 'after', 'the', 'chinese', 'y
12
None
None
Row 4
0
What is the full form of .com ?
['what', 'is', 'the', 'full', 'form', 'of', 'com']
7
None
None
first 5 rows from dataset after extracting lexical and syntactic data

Row 0
1
How did serfdom develop in and then leave Russia ?
['how', 'did', 'serfdom', 'develop', 'in', 'and', 'then', 'leave', 'russia'
9
['how', 'did', 'in', 'and', 'then']
['WRB', 'VBD', 'IN', 'CC', 'RB']
Row 1
2
What films featured the character Popeye Doyle ?
['what', 'films', 'featured', 'the', 'character', 'popeye', 'doyle']
7
['what', 'the', 'character']
['WP', 'DT', 'NN']
Row 2
1
How can I find a list of celebrities ' real names ?
['how', 'can', 'find', 'list', 'of', 'celebrities', 'real', 'names']
8
['how', 'can', 'find', 'list', 'of', 'real', 'names']
['WRB', 'MD', 'VB', 'NN', 'IN', 'JJ', 'NNS']
```

```
['WRB', 'MD', 'VB', 'NN', 'IN', 'JJ', 'NNS']
Row 3
2
What fowl grabs the spotlight after the Chinese Year of the Monkey ?
['what', 'fowl', 'grabs', 'the', 'spotlight', 'after', 'the', 'chinese', 'ye
12
['what', 'the', 'after', 'the', 'chinese', 'year', 'of', 'the']
['WP', 'DT', 'IN', 'DT', 'JJ', 'NN', 'IN', 'DT']
Row 4
0
What is the full form of .com ?
['what', 'is', 'the', 'full', 'form', 'of', 'com']
7
['what', 'is', 'the', 'full', 'form', 'of', 'com']
```

```
#uniques pos from train data
unique_pos=set([e for row in train_data for e in row[-1]])
```

```
#mapper for word given all target classes
map_probWordGivenTarClass=probWordsGivenTarClasses(sorted(tarClass_map.values()),
```

```
#mapper for pos given all target classes
map_prob_pos_given_tarClass=probWordsGivenTarClasses(sorted(tarClass_map.values
```

```
#calculate the probabilities for lexical and syntactical attribute
train_data_after_prob_cal=trainCalculateColProbs(train_data,map_probWordGivenTa
```

```
first 5 rows from training dataset after probability calculations
Row 0
1
How did serfdom develop in and then leave Russia ?
['how', 'did', 'serfdom', 'develop', 'in', 'and', 'then', 'leave', 'russia']
9
7.734196724847904e-08
0.00013266783798205007
Row 1
2
What films featured the character Popeye Doyle ?
['what', 'films', 'featured', 'the', 'character', 'popeye', 'doyle']
7
0.0013101844897959183
0.7824579360530675
Row 2
1
How can I find a list of celebrities ' real names ?
['how', 'can', 'find', 'list', 'of', 'celebrities', 'real', 'names']
8
2.7404778811929026e-13
2.4588954000112153e-05
Row 3
2
What fowl grabs the spotlight after the Chinese Year of the Monkey ?
['what', 'fowl', 'grabs', 'the', 'spotlight', 'after', 'the', 'chinese', 'yea
12
4.671329086249706e-10
0.08633452119520049
```

```
Row 4
0
What is the full form of .com ?
['what', 'is', 'the', 'full', 'form', 'of', 'com']
7
1.75915223862936e-12
0.02025786748867755
```

```
#selecting only target class,length,lexical,synatax attribute
train_data=[[row[0],row[3],row[4],row[5]] for row in train_data_after_prob_cal]
```

```
#creation of test dataset
tarClass_map,test_ngramFreq,test_moFreq500,test_data=createDataset("dt_test.csv
```

```
None
Row 2
3
Who was Galileo ?
['who', 'was', 'galileo']
3
None
None
Row 3
1
What is an atom ?
['what', 'is', 'an', 'atom']
4
None
None
Row 4
5
When did Hawaii become a state ?
['when', 'did', 'hawaii', 'become', 'state']
5
None
None
first 5 rows from dataset after extracting lexical and syntactic data
Row 0
5
How far is it from Denver to Aspen ?
['how', 'far', 'is', 'it', 'from', 'denver', 'to', 'aspen']
8
['how', 'far', 'is', 'it', 'from', 'denver', 'to', 'aspen']
['WRB', 'RB', 'VBZ', 'PRP', 'IN', 'NN', 'TO', 'VB']
Row 1
4
What county is Modesto , California in ?
['what', 'county', 'is', 'modesto', 'california', 'in']
6
['what', 'county', 'is', 'modesto', 'california', 'in']
['WP', 'NN', 'VBZ', 'NN', 'NN', 'IN']
Row 2
3
Who was Galileo ?
['who', 'was', 'galileo']
3
['who', 'was', 'galileo']
['WP', 'VBD', 'NN']
```

```
['WP', 'VBD', 'NN']
Row 3
1
What is an atom ?
['what', 'is', 'an', 'atom']
4
['what', 'is', 'an', 'atom']
['WP', 'VBZ', 'DT', 'NN']
Row 4
5
When did Hawaii become a state ?
['when', 'did', 'hawaii', 'become', 'state']
5
['when', 'did', 'hawaii', 'become', 'state']
['WRB', 'VBD', 'NN', 'NN', 'NN']
```

```
#unique pos from test data
unique_pos=set([e for row in test_data for e in row[-1]])
```

```
#calculate the probabilities for lexical and syntactical attribute
test_data_after_prob_cal=testCalculateColProbs(test_data,map_probWordGivenTarCl
actual_target=np.array(test_data_after_prob_cal).T.tolist()[0]
test_predicted_algorithm=[]
```

```
first 5 rows from testing dataset after probability calculations
Row 0
5
How far is it from Denver to Aspen ?
['how', 'far', 'is', 'it', 'from', 'denver', 'to', 'aspen']
8
5.4853999922335294e-08
0.0008350159563586373
Row 1
4
What county is Modesto , California in ?
['what', 'county', 'is', 'modesto', 'california', 'in']
6
1.1856409585819994e-06
1.5584426076095899
Row 2
3
Who was Galileo ?
['who', 'was', 'galileo']
3
0.058518982365854205
0.469523748848022
Row 3
1
What is an atom ?
['what', 'is', 'an', 'atom']
4
0.005470750385762961
0.7515745108399025
Row 4
5
When did Hawaii become a state ?
['when', 'did', 'hawaii', 'become', 'state']
5
```

```
2.907471246516451e-06
0.6002377132007415
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: VisibleDeprec
  This is separate from the ipykernel package so we can avoid doing imports u
```

```
for algorithm in ["gini","entropy","ce"]:
    printReverseRed(algorithm)
    #build tree
    printReverseGreen("Building Tree ...")
    start=bulidTree(train_data,algorithm)
#     prediction
    printReverseGreen("Predicting test data ...")
    test_predicted=[]
    for questn in test_data_after_prob_cal:
        test_predicted.append(decisionTreeClassifier(start,[questn[3],questn[4]
    cnt=0
    for i in range(len(actual_target)):
        if actual_target[i]==test_predicted[i]:
            cnt+=1
    print("Accuracy",cnt/len(test_predicted)*100)
    #calculating precision,recall,fscore
    metric=precision_recall_fscore_support(actual_target,test_predicted,labels=
    printReverseGreen(" "*10+"| ".join([e[0]+" "+str(e[1]) for e in tarClass_ma
    metric_name=['precision','recall','f-score','support']
    for i  in range(len(metric)):
        print(metric_name[i],metric[i].tolist())
    test_predicted_algorithm.append(test_predicted)
```

```
gini
Building Tree ...
Predicting test data ...
Accuracy 15.0
          ABBR 0| DESC 1| ENTY 2| HUM 3| LOC 4| NUM 5
precision [0.0, 0.18, 0.11920529801324503, 0.14213197969543148, 0.17142857142
recall [0.0, 0.06521739130434782, 0.19148936170212766, 0.4307692307692308, 0.
f-score [0.0, 0.09574468085106382, 0.1469387755102041, 0.2137404580152672, 0.
support [9, 138, 94, 65, 81, 113]
entropy
Building Tree ...
Predicting test data ...
Accuracy 16.6
          ABBR 0| DESC 1| ENTY 2| HUM 3| LOC 4| NUM 5
precision [0.0, 0.23255813953488372, 0.14507772020725387, 0.1568627450980392,
recall [0.0, 0.07246376811594203, 0.2978723404255319, 0.36923076923076925, 0.
f-score [0.0, 0.11049723756906078, 0.19512195121951217, 0.22018348623853212,
support [9, 138, 94, 65, 81, 113]
ce
Building Tree ...
Predicting test data ...
Accuracy 18.6
          ABBR 0| DESC 1| ENTY 2| HUM 3| LOC 4| NUM 5
precision [0.015625, 0.35135135135135137, 0.19402985074626866, 0.15, 0.202898
recall [0.1111111111111111, 0.18840579710144928, 0.2765957446808511, 0.276923
f-score [0.0273972602739726, 0.24528301886792453, 0.2280701754385965, 0.19459
support [9, 138, 94, 65, 81, 113]
```

```
#Observe how many samples are mis-classified using gini index based
#model but correctly classified by mis-classification error and
#cross-entropy based model.
false_gini_true_entropy=0
false_gini_true_ce=0
```

```
for i in range(len(test_predicted)):
    if test_predicted_algo[0][i]!=actual_target[i] and test_predicted_algo[1][i
        false_gini_true_entropy+=1
    if test_predicted_algo[0][i]!=actual_target[i] and test_predicted_algo[2][i
        false_gini_true_ce+=1
```

```
print("samples misclassfied by Gini but correctly classified by entropy",false_g
print("samples misclassfied by Gini but correctly classified by Classification E
```

```
samples misclassfied by Gini but correctly classified by entropy 38
samples misclassfied by Gini but correctly classified by Classification Error
```