In [1]:
```
import pandas as pd
```

In [2]:
```
df=pd.read_csv("f:/dataset/sentiment/Restaurant_Reviews.txt",sep="\t")
df
```

Out[2]:

	Review	Liked
0	Wow... Loved this place.	1
1	Crust is not good.	0
2	Not tasty and the texture was just nasty.	0
3	Stopped by during the late May bank holiday of...	1
4	The selection on the menu was great and so wer...	1
...	...	...
995	I think food should have flavor and texture an...	0
996	Appetite instantly gone.	0
997	Overall I was not impressed and would not go b...	0
998	The whole experience was underwhelming, and I ...	0
999	Then, as if I hadn't wasted enough of my life ...	0

1000 rows × 2 columns

review sentiment f5o@od is # good & good! good food is tasty good Quality is Good good food is not good not good servi89ce is poor not good it is to_o costly not good che^ap quality not good

In [30]:
```
doc1='f5o@od is # good & good!'
doc2='& Food # is * tasty'
doc3='Quality is Good'
doc4='food is not good'
doc5='servi89ce is Poor poor means very poor'
doc6='it is to_o costly'
doc7='che^ap quality'

corpus=[doc1,doc2,doc3,doc4,doc5,doc6,doc7]
target=['pos','pos','pos','neg','neg','neg','neg']
print(corpus)
```

```
['f5o@od is # good & good!', '& Food # is * tasty', 'Quality is Good', 'food is not goo
d', 'servi89ce is Poor poor means very poor', 'it is to_o costly', 'che^ap quality']
```

In [7]:
```
#Text Preprocessing

#step-1 convert corpus to lowercase
corpus1=list(map(str.lower,corpus))
print(corpus1)

#step-2 remove punctutations(symblos like _%^&()!@#$)
import re
def removePunc(doc):
    newdoc=re.sub("[^a-z ]","",doc)
    return newdoc

print(removePunc('f5o@od is # good & good!'))

corpus2=list(map(removePunc,corpus1))
```

```
print(corpus2)

#step-3 remove stopwords(words having no sentiment) like it,is,was,did,has,have,
```

```
['f5o@od is # good & good!', '& food # is * tasty', 'quality is good', 'food is not goo
d', 'servi89ce is poor poor means very poor', 'it is to_o costly', 'che^ap quality']
food is  good  good
['food is  good  good', ' food  is  tasty', 'quality is good', 'food is not good', 'serv
ice is poor poor means very poor', 'it is too costly', 'cheap quality']
```

In [9]:
```
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
```

In [12]:
```
print(len(ENGLISH_STOP_WORDS))
print(ENGLISH_STOP_WORDS)
```

```
318
frozenset({'besides', 'everywhere', 'seems', 'ourselves', 'anyhow', 'your', 'thru', 'int
erest', 'below', 'now', 'an', 'wherein', 'whole', 'very', 'anyone', 'hereafter', 'unde
r', 'describe', 'cannot', 'might', 'somewhere', 'mostly', 'along', 'being', 'most', 'per
haps', 'nine', 'least', 'whether', 'off', 'everyone', 'without', 'never', 'twenty', 'thi
s', 'whereby', 'yourselves', 'beforehand', 'would', 'yours', 'could', 'whose', 'sincer
e', 'it', 'whereupon', 'become', 'though', 'done', 'other', 'we', 'anyway', 'and', 'exce
pt', 'upon', 'up', 'meanwhile', 'find', 'six', 'be', 'noone', 'mine', 'part', 'well', 'i
ndeed', 'too', 'both', 'with', 'several', 'alone', 'yet', 'when', 'am', 'even', 'its',
'thick', 'nowhere', 'get', 'is', 'former', 'than', 'whereafter', 'before', 'there', 'sam
e', 'may', 'fire', 'somehow', 'third', 'during', 'about', 'please', 'every', 'couldnt',
'through', 'more', 'back', 'hasnt', 'himself', 'yourself', 'itself', 'keep', 'each', 'so
metime', 'afterwards', 'although', 'nothing', 'latter', 'or', 'themselves', 'per', 'ther
eby', 'beyond', 'none', 'us', 'therein', 'formerly', 'empty', 'con', 'de', 'see', 'whil
e', 'no', 'next', 'hers', 'if', 'those', 'by', 'few', 'these', 'together', 'serious', 'm
ill', 'down', 'so', 'otherwise', 'will', 'whither', 'herein', 'ours', 'has', 'latterly',
'are', 'behind', 'wherever', 'amongst', 'seemed', 'in', 'our', 'then', 'less', 'rather',
'many', 'move', 'first', 'anywhere', 'into', 'from', 'herself', 'made', 'should', 'amoun
t', 'her', 'until', 'nobody', 'were', 'others', 'front', 'which', 'becoming', 'where',
'either', 'have', 'seeming', 'three', 'against', 'hereupon', 'two', 'how', 'because', 'n
or', 'the', 'toward', 'me', 'can', 'etc', 'fifty', 'ten', 'thin', 'thus', 'to', 'cry',
'fill', 'she', 'not', 'as', 'around', 'for', 'sixty', 'i', 'hereby', 'cant', 'one', 'm
y', 'since', 'go', 'already', 'towards', 'all', 'whoever', 'any', 'top', 'but', 'amongs
t', 'what', 'was', 'show', 'detail', 'over', 'you', 'on', 'else', 'further', 'eg', 'elev
en', 'four', 'whenever', 'system', 'another', 'un', 'always', 'ie', 'twelve', 'almost',
'hundred', 'that', 'thereafter', 'him', 'eight', 're', 'last', 'he', 'who', 'found', 'na
mely', 'often', 'nevertheless', 'own', 'been', 'neither', 'hence', 'whereas', 'ever', 'v
ia', 'due', 'also', 'they', 'do', 'some', 'whatever', 'whence', 'only', 'had', 'onto',
'must', 'again', 'a', 'something', 'thereupon', 'moreover', 'becomes', 'at', 'seem', 'be
side', 'once', 'after', 'side', 'five', 'why', 'put', 'bill', 'call', 'however', 'ltd',
'co', 'whom', 'within', 'elsewhere', 'inc', 'of', 'give', 'here', 'still', 'enough', 'th
eir', 'full', 'throughout', 'anything', 'became', 'fifteen', 'them', 'name', 'myself',
'above', 'someone', 'forty', 'out', 'bottom', 'thence', 'much', 'sometimes', 'across',
'such', 'his', 'among', 'therefore', 'everything', 'between', 'take'})
```

In [18]:
```
spwords=list(ENGLISH_STOP_WORDS)
spwords.remove('not')
```

In [23]:
```
def removespwords(doc):
    wordslist=doc.split()
    newdoc=""
    for word in wordslist:
        if word not in spwords:
            newdoc=newdoc+word+" "
    return newdoc.strip()

removespwords('he and she likes quality of the food')
```

Out[23]:
```
'likes quality food'
```

In [24]:
```
corpus3=list(map(removespwords,corpus2))
print(corpus3)
```

```
['food good good', 'food tasty', 'quality good', 'food not good', 'service poor poor mea
ns poor', 'costly', 'cheap quality']
```

In [29]:
```
#step-4 extract features(each unique is a feature)

#food,good,tasty,quality,not,service,poor,means,costly,cheap
#cheap,costly,food,good,means,not,poor,quality,service,tasty

#step-5 obtain vector representation of each document and get feature matrix of docs
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer()
X=cv.fit_transform(corpus3) #first extract features then returns sparse matrix for docs
print(cv.get_feature_names_out())
print(X)
X1=X.toarray()
print(X1)
```

```
['cheap' 'costly' 'food' 'good' 'means' 'not' 'poor' 'quality' 'service'
 'tasty']
  (0, 2)	1
  (0, 3)	2
  (1, 2)	1
  (1, 9)	1
  (2, 3)	1
  (2, 7)	1
  (3, 2)	1
  (3, 3)	1
  (3, 5)	1
  (4, 8)	1
  (4, 6)	3
  (4, 4)	1
  (5, 1)	1
  (6, 7)	1
  (6, 0)	1
[[0 0 1 2 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 1]
 [0 0 0 1 0 0 0 1 0 0]
 [0 0 1 1 0 1 0 0 0 0]
 [0 0 0 0 1 0 3 0 1 0]
 [0 1 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 1 0 0]]
```

In [49]:
```
#model training
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import RandomForestClassifier

model=KNeighborsClassifier()
model=LogisticRegression()
model=DecisionTreeClassifier()
model=SVC()
model=AdaBoostClassifier()
model=RandomForestClassifier()
model.fit(X,target)
print(model.predict(cv.transform(['good quality is not food'])))
```

```
['neg']
```

In [53]:
```
sample='Food quality is not good$'
sample1=sample.lower()
```

```
sample2=removePunc(sample1)
sample3=removespwords(sample2)
print(sample3)
sample4=cv.transform([sample3])
print(model.predict(sample4))
```

```
food quality not good
['neg']
```

In [ ]:

```
#text cleaning
#feature extraction
#vectorization
#model training
```