TF-IDF [Intuition]

→ Term frequency & Inverse document frequency.

$$\text{Term frequency (TF)} = \frac{\text{No. of repetition of words in sentences}}{\text{No of words in sentence.}}$$

$$\text{Inverse Document frequency (IDF)} = \log\left(\frac{\text{No. of sentences}}{\text{No. of sentences containing words}}\right)$$

#. finally $\boxed{TF \times IDF}$ is used to generate vector

S_1	good	boy	
S_2	good	girl	
S_3	good	boy	girl.

word	frequ
good	3
boy	2
girl	2

TF

vectors →

~~Bow~~

TF-IDF →

	S_1	S_2	S_3
good	$1/2$	$1/2$	$1/3$
boy	$1/2$	0	$1/3$
girl	0	$1/2$	$1/3$

IDF

words	IDF
good	$\log(3/3) = 0$
boy	$\log(3/2)$
girl	$\log(3/2$

	f_1	f_2	f_3	op
⇒	good	boy	girl	
S_1	0	$1/2 \log(3/2)$	0	
S_2	0	0	$1/2 \log(3/2)$	
S_3	0	$1/3 \log(3/2$	$1/3 \log(3/2)$	

↑ used to train my algo as it is dependent feature.

TF-IDF

```
import nltk
Pagraph = ___________

# cleaning
import re
~~from Stem nltk.stem import PorterStemmer~~
from nltk.stem import WordNetLammatizer
from nltk.corpus import stopwords

Stemmer = PorterStemmer()
Lammatizer = WordNet Lemmatizer()
Sentences = nltk.sent_tokenize (paragraph)
Corpus = [ ]

for i in range(len(sentences)):
    revies = re.sub('[^a-zA-Z]', ' ', sentences[i]))
    review = review.lower()
    review = review.split()
    review = [stemmer.stem (word) for
         word in review if word not in set(stopwords.
                  words ('english'))].
     review = ' '.join(review).
     corpus.append (review)

# Calculate the TF-IDF .
from sklearn.feature_extraction.text import TFidfVectorizer.
cv = Tfidfvectorizer()
x = cv.fit_transform(corpus).toarray() .
```