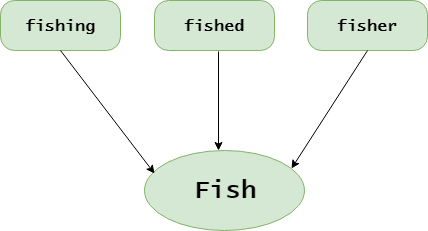
**Step 1:** Import dataset with setting delimiter as ‘\t’ as columns are separated as tab space. Reviews and their category(0 or 1) are not separated by any other symbol but with tab space as most of the other symbols are is the review (like $ for price, ….!, etc) and the algorithm might use them as delimiter, which will lead to strange behavior (like errors, weird output) in output.

|  |
| --- |
| # Importing Libraries  import numpy as np  import pandas as pd    # Import dataset  dataset = pd.read\_csv('Restaurant\_Reviews.tsv', delimiter = '\t') |

**Step 2:** Text Cleaning or Preprocessing

* **Remove Punctuations, Numbers**: Punctuations, Numbers doesn’t help much in processong the given text, if included, they will just increase the size of bag of words that we will create as last step and decrase the efficency of algorithm.
* **Stemming**: Take roots of the word  
  
* **Convert each word into its lower case**: For example, it useless to have same words in different cases (eg ‘good’ and ‘GOOD’).

# library to clean data

import re

# Natural Language Tool Kit

import nltk

nltk.download('stopwords')

# to remove stopword

from nltk.corpus import stopwords

# for Stemming propose

from nltk.stem.porter import PorterStemmer

# Initialize empty array

# to append clean text

corpus = []

# 1000 (reviews) rows to clean

for i in range(0, 1000):

# column : "Review", row ith

review = re.sub('[^a-zA-Z]', ' ', dataset['Review'][i])

# convert all cases to lower cases

review = review.lower()

# split to array(default delimiter is " ")

review = review.split()

# creating PorterStemmer object to

# take main stem of each word

ps = PorterStemmer()

# loop for stemming each word

# in string array at ith row

review = [ps.stem(word) for word in review

if not word in set(stopwords.words('english'))]

# rejoin all string array elements

# to create back into a string

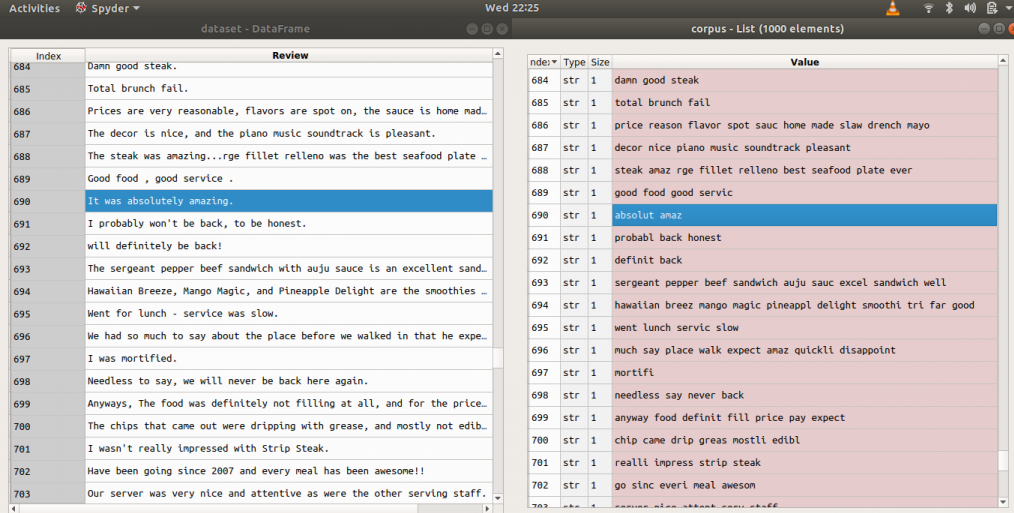
review = ' '.join(review)

# append each string to create

# array of clean text

corpus.append(review)

**Examples:** Before and after applying above code (reviews = > before, corpus => after)



**Step 3:**Tokenization, involves splitting sentences and words from the body of the text.

**Step 4:** Making the bag of words via sparse matrix

* Take all the different words of reviews in the dataset without repeating of words.
* One column for each word, therefore there are going to be many columns.
* Rows are reviews
* If word is there in row of dataset of reviews, then the count of word will be there in row of bag of words under the column of the word.

**Examples:** Let’s take a dataset of reviews of only two reviews

Input : "dam good steak", "good food good servic"

Output :

https://media.geeksforgeeks.org/wp-content/uploads/eg.png

For this purpose we need CountVectorizer class from sklearn.feature\_extraction.text.  
We can also set a max number of features (max no. features which help the most via attribute “max\_features”). Do the training on the corpus and then apply the same transformation to the corpus “.fit\_transform(corpus)” and then convert it into an array. If review is positive or negative that answer is in the second column of the dataset[:, 1] : all rows and 1st column (indexing from zero).

# Creating the Bag of Words model

from sklearn.feature\_extraction.text import CountVectorizer

# To extract max 1500 feature.

# "max\_features" is attribute to

# experiment with to get better results

cv = CountVectorizer(max\_features = 1500)

# X contains corpus (dependent variable)

X = cv.fit\_transform(corpus).toarray()

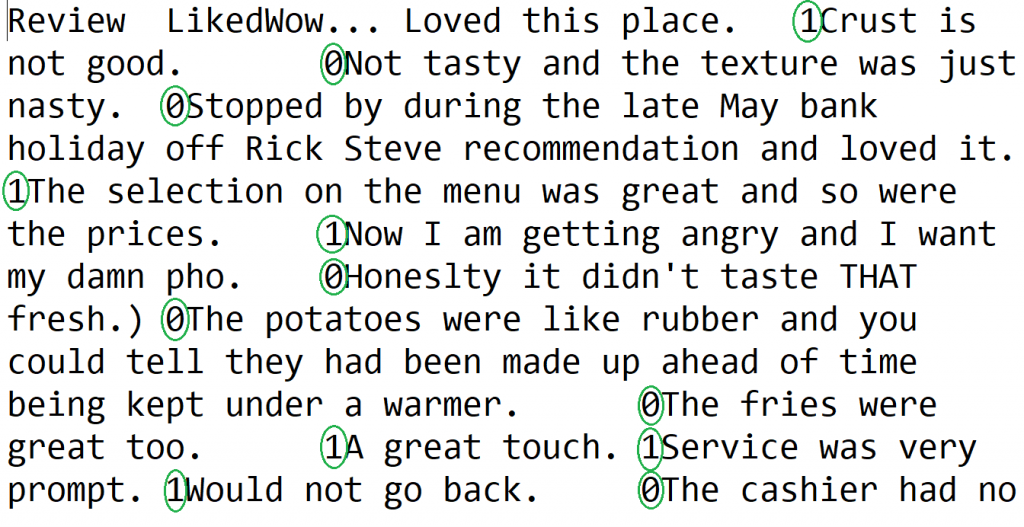
# y contains answers if review

# is positive or negative

y = dataset.iloc[:, 1].values

***Description of the dataset to be used:***

* *Columns seperated by \t (tab space)*
* *First column is about reviews of people*
* *In second column, 0 is for negative review and 1 is for positive review*



**Step 5 :** Splitting Corpus into Training and Test set. For this, we need class train\_test\_split from sklearn.cross\_validation. Split can be made 70/30 or 80/20 or 85/15 or 75/25, here I choose 75/25 via “test\_size”.  
X is the bag of words, y is 0 or 1 (positive or negative).

from sklearn.model\_selection import train\_test\_split

# Splitting the dataset into

# the Training set and Test set

from sklearn.cross\_validation import train\_test\_split

# experiment with "test\_size"

# to get better results

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25)

**Step 6:**Fitting a Predictive Model (here random forest)

* Since Random fored is ensemble model (made of many trees) from sklearn.ensemble, import RandomForestClassifier class
* With 501 tree or “n\_estimators” and criterion as ‘entropy’
* Fit the model via .fit() method with attributes X\_train and y\_train

# Fitting Random Forest Classification

# to the Training set

from sklearn.ensemble import RandomForestClassifier

# n\_estimators can be said as number of

# trees, experiment with n\_estimators

# to get better results

model = RandomForestClassifier(n\_estimators = 501, criterion = 'entropy')

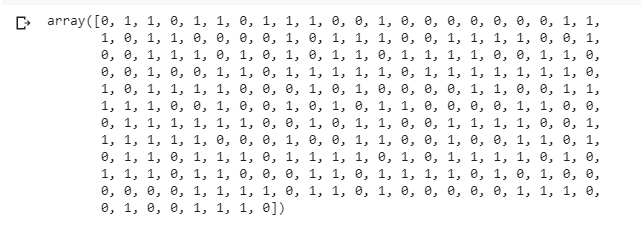
model.fit(X\_train, y\_train)

**Step 7:**Pridicting Final Results via using .predict() method with attribute X\_test

# Predicting the Test set results

y\_pred = model.predict(X\_test)

y\_pred

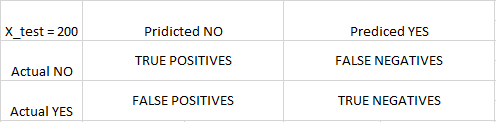


**Note:**Accuracy with random forest was 72%.(It may be different when performed experiment with different test size, here = 0.25).

**Step 8:**To know the accuracy, confusion matrix is needed.

Confusion Matrix is a 2X2 Matrix.

***TRUE POSITIVE :****measures the proportion of actual positives that are correctly identified.****TRUE NEGATIVE :****measures the proportion of actual positives that are not correctly identified.****FALSE POSITIVE :****measures the proportion of actual negatives that are correctly identified.****FALSE NEGATIVE :****measures the proportion of actual negatives that are not correctly identified.*

**Note :**True or False refers to the assigned classification being Correct or Incorrect, while Positive or Negative refers to assignment to the Positive or the Negative Category  


# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

cm

