import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import re

import nltk

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

ps = PorterStemmer()

nltk.download('stopwords')

dataset = pd.read\_csv('Restaurant\_Reviews.tsv', delimiter = '\t')

dataset['Review'][0]

clean\_reviews = []

for i in range(1000):

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

text = text.lower()

text = text.split()

#t = [word for word in text if not word in set(stopwords.words('english'))]

text = [ps.stem(word) for word in text if not word in set(stopwords.words('english'))]

text = ' '.join(text)

clean\_reviews.append(text)

from sklearn.feature\_extraction.text import CountVectorizer

cv = CountVectorizer(max\_features = 800)

X = cv.fit\_transform(clean\_reviews)

X = X.toarray()

y = dataset['Liked'].values

from sklearn.naive\_bayes import GaussianNB

nb = GaussianNB()

nb.fit(X, y)

nb.score(X, y)

print(cv.get\_feature\_names())

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()

knn.fit(X, y)

knn.score(X, y)

from sklearn.linear\_model import LogisticRegression

log\_reg = LogisticRegression()

log\_reg.fit(X, y)

log\_reg.score(X, y)

from sklearn.tree import DecisionTreeClassifier

dtf = DecisionTreeClassifier()

dtf.fit(X, y)

dtf.score(X, y)

from sklearn.svm import SVC

svm = SVC()

svm.fit(X, y)

svm.score(X, y)

from sklearn.ensemble import VotingClassifier

vc = VotingClassifier([('LR', log\_reg),

('DT', dtf),

('NB', nb),

('SVM', svm),

('KNN', knn)])

vc.fit(X, y)

vc.score(X, y)

from sklearn.ensemble import BaggingClassifier

bag = BaggingClassifier(nb, n\_estimators = 3)

bag.fit(X, y)

bag.score(X, y)

from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n\_estimators = 5)

rf.fit(X, y)

rf.score(X, y)