```
# NAME: OMKAR R SUNKERSETT
# UNIVERSITY OF MICHIGAN
# COURSE: SI 630 NLP

import nltk, string, time
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
from sklearn.naive_bayes import BernoulliNB, MultinomialNB
from sklearn.linear_model import LogisticRegression, SGDClassifier, Perceptron
from sklearn.neural_network import MLPClassifier
from nltk import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
from sklearn.metrics import f1_score

class LemmaTokenizer(object):
	def __init__(self):
		self.wnl = WordNetLemmatizer()
	def __call__(self, doc):
		return [self.wnl.lemmatize(t) for t in word_tokenize(doc)]

class LemmaTokenizerWithPOSTagging(object):
	def __init__(self):
		self.wnl = WordNetLemmatizer()
	def get_wordnet_pos(self, treebank_tag):
		if treebank_tag.startswith('J'):
			return wordnet.ADJ
		elif treebank_tag.startswith('V'):
			return wordnet.VERB
		elif treebank_tag.startswith('N'):
			return wordnet.NOUN
		elif treebank_tag.startswith('R'):
			return wordnet.ADV
		else:
			return wordnet.NOUN
	def __call__(self, doc):
		return [self.wnl.lemmatize(w, self.get_wordnet_pos(pt)) for (w,pt) in
nltk.pos_tag(doc)]


start_time = time.time()
full_df = pd.read_csv('/Users/omkarsunkersett/Downloads/yelp-dataset/yelp_review.csv',
keep_default_na=False)
full_df = full_df[['text','stars']]
full_df.loc[full_df['stars'] >= 4, 'sentiment'] = 1
full_df.loc[full_df['stars'] <= 3, 'sentiment'] = 0
full_df.sentiment = full_df.sentiment.astype(int)
print(full_df[(full_df['sentiment'] == 1)].count())
print(full_df[(full_df['sentiment'] == 0)].count())
full_df = pd.concat([full_df[(full_df['sentiment'] == 1)].sample(n=300000),
full_df[(full_df['sentiment'] == 0)].sample(n=300000)])
print(full_df[(full_df['sentiment'] == 1)].count())
print(full_df[(full_df['sentiment'] == 0)].count())
train_df, test_df = train_test_split(full_df, test_size = 0.7)
del full_df

elapsed_time = time.time() - start_time
print("Loaded data sets: "+str(elapsed_time/60) + " minutes")

# train_df['text'] = train_df['text'].str.replace("[" + string.punctuation + "]", "")
```

```
# train_df['text'] = train_df['text'].str.replace("#", "")
# train_df['text'] = train_df['text'].str.replace(r"http\S+", "")

# test_df['text'] = test_df['text'].str.replace("[" + string.punctuation + "]", "")
# test_df['text'] = test_df['text'].str.replace("#", "")
# test_df['text'] = test_df['text'].str.replace(r"http\S+", "")

# elapsed_time = time.time() - start_time
# print("Removed punctuation, hashtags & URLs: "+str(elapsed_time/60) + " minutes")

train_data = train_df['text'].tolist()
print(train_data[0])
train_labels = train_df['sentiment'].tolist()
del train_df

test_data = test_df['text'].tolist()
print(test_data[0])
test_labels = test_df['sentiment'].tolist()
del test_df

elapsed_time = time.time() - elapsed_time
print("Created memory sets: "+str(elapsed_time/60) + " minutes")

pred_random = list(np.random.choice(2, len(test_labels)))
print("Random Baseline F1 Score: " + str(f1_score(test_labels, pred_random,
average='binary')))
elapsed_time = time.time() - elapsed_time
print("Time for Calculating F1 Score Baseline: " + str(elapsed_time/60) + " minutes")

vectorizer = TfidfVectorizer(sublinear_tf=True, use_idf=True, analyzer='word')
#vectorizer = TfidfVectorizer(min_df=5, max_df=0.95, ngram_range=(2,4), sublinear_tf=True,
use_idf=True, analyzer='word', tokenizer=LemmaTokenizer())
#vectorizer = TfidfVectorizer(min_df=5, max_df=0.95, ngram_range=(2,4), sublinear_tf=True,
use_idf=True, analyzer='word', stop_words='english', tokenizer=LemmaTokenizer())
train_vectors = vectorizer.fit_transform(train_data)
test_vectors = vectorizer.transform(test_data)

vector_time = time.time() - elapsed_time
print("Created vectorizer: "+str(vector_time/60) + " minutes")

while (True):
	ans = int(input("Enter an option (9 to stop): "))
	if ans == 1:
		classifier = RandomForestClassifier()
	elif ans == 2:
		classifier = BernoulliNB()
	elif ans == 3:
		classifier = MultinomialNB()
	elif ans == 4:
		classifier = svm.LinearSVC()
	elif ans == 5:
		classifier = LogisticRegression()
	elif ans == 6:
		classifier = SGDClassifier()
	elif ans == 7:
		classifier = Perceptron()
	elif ans == 8:
		classifier = MLPClassifier()
	elif ans == 9:
		break
	classifier.fit(train_vectors, train_labels)
	elapsed_time = time.time() - elapsed_time
	print("Performed training: "+str(elapsed_time/60) + " minutes")
	prediction = classifier.predict(test_vectors)
```

```
        elapsed_time = time.time() - elapsed_time
        print("Performed testing: "+str(elapsed_time/60) + " minutes")
        print("F1 Score: " + str(f1_score(test_labels, prediction, average='binary')))
        elapsed_time = time.time() - elapsed_time
        print("Time for F1 Score Calculation: " + str(elapsed_time/60) + " minutes")

elapsed_time = time.time() - start_time
print("Total time taken: "+str(elapsed_time/60) + " minutes")
```