```
# -*- coding: utf-8 -*-
"""Export CSV.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1uNTFaEcHjLGUkGUojaKE8gCcieD9FaLQ

<center><h1>Mini Project 2 - Bernoulli Naïve Bayes</h1>
<h4>This file is for file exporting.</h4></center>

<h3>Team Members:</h3>
<center>
Yi Zhu, 260716006<br>
Fei Peng, 260712440<br>
Yukai Zhang, 260710915
</center>
"""

from google.colab import drive
drive.mount('/content/drive')

# make path = './' in-case you are running this locally
path = '/content/drive/My Drive/ECSE_551_F_2020/Mini_Project_02/'

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import Normalizer
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.feature_extraction import text
from sklearn import metrics
from sklearn.model_selection import GridSearchCV, cross_val_score, KFold
from sklearn.pipeline import make_pipeline

!pip install nltk
import nltk
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')

from nltk.stem import PorterStemmer
from nltk import word_tokenize
from nltk import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet

"""Additional classifiers:
1. Logistic Regression
2. Multinomial Naïve Bayes
3. Support Vector Machine
4. Random Forest
5. Decision Tree
6. Ada Boost
7. k-Neighbors
8. Neural Network
"""

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn import svm
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier

reddit_dataset = pd.read_csv(path+"train.csv")
reddit_test = pd.read_csv(path+"test.csv")

X = reddit_dataset['body']
y = reddit_dataset['subreddit']

"""# Define Vectorizer
### (To vectorize the text-based data to numerical features)

1. CountVectorizer
1) Use "CountVectorizer" to transform text data to feature vectors.
2) Normalize your feature vectors
"""

def count_vectorizer(X_train, X_test):
    vectorizer = CountVectorizer()
    vectors_train = vectorizer.fit_transform(X_train)
    vectors_test = vectorizer.transform(X_test)

    # z-score normalization
    normalizer_train = Normalizer().fit(X=vectors_train)
    vectors_train = normalizer_train.transform(vectors_train)
    vectors_test = normalizer_train.transform(vectors_test)

    return vectors_train, vectors_test

"""2. CountVectorizer with stop word
1) Use "CountVectorizer" with stop word to transform text data to vector.
2) Normalize your feature vectors
"""

def count_vec_with_sw(X_train, X_test):
    stop_words = text.ENGLISH_STOP_WORDS
    vectorizer = CountVectorizer(stop_words=stop_words)
    vectors_train_stop = vectorizer.fit_transform(X_train)
    vectors_test_stop = vectorizer.transform(X_test)

    # z-score normalization
    normalizer_train = Normalizer().fit(X=vectors_train_stop)
    vectors_train_stop= normalizer_train.transform(vectors_train_stop)
    vectors_test_stop = normalizer_train.transform(vectors_test_stop)

    return vectors_train_stop, vectors_test_stop

"""3. TF-IDF
1) use "TfidfVectorizer" to weight features based on your train set.
2) Normalize your feature vectors
"""

def tfidf_vectorizer(X_train, X_test):
    tf_idf_vectorizer = TfidfVectorizer()
    vectors_train_idf = tf_idf_vectorizer.fit_transform(X_train)
    vectors_test_idf = tf_idf_vectorizer.transform(X_test)

    # z-score normalization
    normalizer_train = Normalizer().fit(X=vectors_train_idf)
    vectors_train_idf= normalizer_train.transform(vectors_train_idf)
```

```
127     vectors_test_idf = normalizer_train.transform(vectors_test_idf)
128
129     return vectors_train_idf, vectors_test_idf
130
131 """4. CountVectorizer with stem tokenizer
132 1) Use "StemTokenizer" to transform text data to vector.
133 2) Normalize your feature vectors
134 """
135
136 class StemTokenizer:
137      def __init__(self):
138        self.wnl =PorterStemmer()
139      def __call__(self, doc):
140        return [self.wnl.stem(t) for t in word_tokenize(doc) if t.isalpha()]
141
142
143 def count_vec_stem(X_train, X_test):
144     vectorizer = CountVectorizer(tokenizer=StemTokenizer())
145     vectors_train_stem = vectorizer.fit_transform(X_train)
146     vectors_test_stem = vectorizer.transform(X_test)
147
148     # z-score normalization
149     normalizer_train = Normalizer().fit(X=vectors_train_stem)
150     vectors_train_stem= normalizer_train.transform(vectors_train_stem)
151     vectors_test_stem = normalizer_train.transform(vectors_test_stem)
152
153     return vectors_train_stem, vectors_test_stem
154
155 """5. CountVectorizer with lemma tokenizer
156 1) Use "LemmaTokenizer" to transform text data to vector.
157 2) Normalize your feature vectors
158 """
159
160 def get_wordnet_pos(word):
161     """Map POS tag to first character lemmatize() accepts"""
162     tag = nltk.pos_tag([word])[0][1][0].upper()
163     tag_dict = {"J": wordnet.ADJ,
164                 "N": wordnet.NOUN,
165                 "V": wordnet.VERB,
166                 "R": wordnet.ADV}
167     return tag_dict.get(tag, wordnet.NOUN)
168
169
170 class LemmaTokenizer:
171      def __init__(self):
172        self.wnl = WordNetLemmatizer()
173      def __call__(self, doc):
174        return [self.wnl.lemmatize(t,pos =get_wordnet_pos(t)) for t in
    word_tokenize(doc) if t.isalpha()]
175
176
177 def count_vec_lemma(X_train, X_test):
178     vectorizer = CountVectorizer(tokenizer=LemmaTokenizer())
179     vectors_train_lemma = vectorizer.fit_transform(X_train)
180     vectors_test_lemma = vectorizer.transform(X_test)
181
182     # z-score normalization
183     normalizer_train = Normalizer().fit(X=vectors_train_lemma)
184     vectors_train_lemma= normalizer_train.transform(vectors_train_lemma)
185     vectors_test_lemma = normalizer_train.transform(vectors_test_lemma)
186
187     return vectors_train_lemma, vectors_test_lemma
188
```

```
"""# Export csv"""

# test set id
X_id = reddit_test['id']
# test set features
X_test = reddit_test['body']

# vectorize the training and testing data
vectors_train, vectors_test = tfidf_vectorizer(X, X_test)
# perform MLP classification
clf = MLPClassifier(random_state=0, max_iter=1000, learning_rate="adaptive",
    learning_rate_init=0.0001).fit(vectors_train, y)

# predict the result
y_test = clf.predict(vectors_test)

# put the result into a pandas dataframe
result = {'id': X_id, 'subreddit': y_test}
df = pd.DataFrame(data=result)

# export to csv
df.to_csv('result.csv', index=False)

# download csv
from google.colab import files
files.download('result.csv')
```