# spam_ham.py

```
# -*- coding: utf-8 -*-
"""
Created on Fri Sep 16 13:57:26 2022

@author: Admin
"""


import os
import string
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import numpy as np
import pandas as pd

df = pd.read_csv('G:/dypiemr2/dypiemr22-
23/sem_I/BE_I/databse/spam_ham_dataset.csv')

import seaborn as sns
import matplotlib.pyplot as plt

df.head()

df = df.drop(['Unnamed: 0'], axis=1)
df.head()
print('Total %s data email'% len(df))

#total class memebers
df['label'].value_counts()

#show graph
df_label = sns.countplot(df['label'])
df_label.set_xticklabels(df['label'].unique())
plt.show()

#data preprocessing
#data text cleaning
# punchuations
punct = []
for char in string.punctuation:
punct.append(char)

import re

def cleaning(txt):
# case folding
text = txt.lower()

# remove multiple space, tabs, dan newlines
text = re.sub('\s+',' ',text)
# remove links
```

```
text = text.replace("http://", " ").replace("https://", " ")
# remove special characters
text = text.encode('ascii', 'replace').decode('ascii')
text = ' '.join(re.sub("([@#][A-Za-z0-9]+)|(\w+:\/\/\S+)"," ", text).split())
# remove punctuation
text = ''.join([word for word in text if word not in punct])
#remove single character
text = re.sub(r"\b[a-zA-Z]\b", "", text)
#remove numbers
text = re.sub(r"\d+", "", text)
#remove multiple spaces (again)
text = re.sub('\s+',' ',text)
return text

# call function for cleaning
# apply fungsi cleaning ke setiap text
df['text_cleaned'] = df['text'].apply(lambda x: cleaning(x))
df = df[['text', 'text_cleaned', 'label']]
df.head()

#compare
print(df['text'][0])
print(df['text_cleaned'][0])

# to remove stop words
import nltk
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
from nltk.corpus import stopwords
stop = stopwords.words('english')
df['text_cleaned'] = df['text_cleaned'].apply(lambda x: ' '.join([word for word
in x.split() if word not in stop]))

#lammitaization
lemmatizer = WordNetLemmatizer()
def get_wordnet_pos(word):
"""Map POS tag to first character lemmatize() accepts"""
tag = nltk.pos_tag([word])[0][1][0].upper()
tag_dict = {"J": wordnet.ADJ,
"N": wordnet.NOUN,
"V": wordnet.VERB,
"R": wordnet.ADV}
return tag_dict.get(tag, wordnet.NOUN)

def do_lemma(string):
lemmatized = ' '.join([lemmatizer.lemmatize(word, get_wordnet_pos(word)) for
word in nltk.word_tokenize(string)])
return lemmatized

df['text_cleaned'] = df['text_cleaned'].apply(lambda x: do_lemma(x))

df = df.drop(['text'], axis=1)
df = df.rename(columns = {'text_cleaned' : 'text'})
```

```
df.columns

from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer()
X = tfidf.fit_transform(df['text'])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

y = df['label']

from time import time
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
# defining the classifier
clf = KNeighborsClassifier(n_neighbors=5, metric='euclidean')

#predicting the time of train and testing
t0 = time()
clf.fit(X_train, y_train)
print("\nTraining time:", round(time()-t0, 3), "s\n")

#predicting the time of testing
t1 = time()
pred = clf.predict(X_test)
print("Predicting time:", round(time()-t1, 3), "s\n")

#calculating and printing the accuracy of the algorithm
print("Accuracy of KNN Algorithm: ", accuracy_score(pred,y_test))
# tested ok 16/09/2022
```