ДОДАТОК А

Вихідний код

```
import os

usr = os.getlogin()

os.chdir('/Users/'+usr+'/Desktop')

cwd = os.getcwd()

print('Working in ', cwd, '\n')


files = [f for f in os.listdir('.') if os.path.isfile(f)]

print('---\nTSV files currently on your desktop: \n')

for f in files:

if '.tsv' in f:

print(f)

data_file = input('\n---\nWhich file would like to analyze? \n\n')


file_prefix = data_file.split('.')

file_prefix = file_prefix[0]+'_'

print('\nFile exports will be prefixed with:', file_prefix)


import pandas

dataset = pandas.read_csv(data_file, delimiter = '\t')

dataset.head()

datacol = input('\n---\nWhich column contains the text data you would
like to analyze?\n\n')

freq = pandas.Series(' '.join(map(str,
dataset[datacol])).split()).value_counts()[:10]

freq

freq1 =  pandas.Series(' '.join(map(str,dataset

 [datacol])).split()).value_counts()[-10:]

freq1
```

```
import re
import nltk

nltk.download('stopwords')
nltk.download('wordnet')

from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import RegexpTokenizer
from nltk.stem.wordnet import WordNetLemmatizer

stop_words = set(stopwords.words("english"))
print(sorted(stop_words))

csw = set(line.strip() for line in open('custom-stopwords.txt'))
csw = [sw.lower() for sw in csw]
print(sorted(csw))
stop_words = stop_words.union(csw)
print(sorted(stop_words))
corpus = []
dataset['word_count'] = dataset[datacol].apply(lambda x:
len(str(x).split(" ")))
ds_count = len(dataset.word_count)
for i in range(0, ds_count):
text = re.sub('[^a-zA-Z]', ' ', str(dataset[datacol][i]))
text = text.lower()
text=re.sub("&lt;/?.*?&gt;"," &lt;&gt; ",text)
text=re.sub("(\\d|\\W)+"," ",text)
text = text.split()
ps=PorterStemmer()
lem = WordNetLemmatizer()
```

```
text = [lem.lemmatize(word) for word in text if not word in
stop_words]
text = " ".join(text)
corpus.append(text)

corpus[10]
from os import path
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import matplotlib.pyplot as plt
%matplotlib inline
wordcloud = WordCloud(
background_color='white',
stopwords=stop_words,
max_words=100,
max_font_size=50,
random_state=42
).generate(str(corpus))
print(wordcloud)
fig = plt.figure(1)
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
fig.savefig(file_prefix + "wordcloud.png", dpi=900)
from sklearn.feature_extraction.text import CountVectorizer
import re
cv=CountVectorizer(max_df=0.8,stop_words=stop_words,
max_features=10000, ngram_range=(1,3))
X=cv.fit_transform(corpus)
```

```
list(cv.vocabulary_.keys())[:10]
def get_top_n_words(corpus, n=None):
vec = CountVectorizer().fit(corpus)
bag_of_words = vec.transform(corpus)
sum_words = bag_of_words.sum(axis=0)
words_freq = [(word, sum_words[0, idx]) for word, idx in
vec.vocabulary_.items()]
words_freq =sorted(words_freq, key = lambda x: x[1],
reverse=True)
return words_freq[:n]
top_words = get_top_n_words(corpus, n=20)
top_df = pandas.DataFrame(top_words)
top_df.columns=["Keyword", "Frequency"]
print(top_df)
top_df.to_csv(file_prefix + '_top_words.csv')

import seaborn as sns
sns.set(rc={'figure.figsize':(13,8)})
g = sns.barplot(x="Keyword", y="Frequency", data=top_df,
palette="Blues_d")
g.set_xticklabels(g.get_xticklabels(), rotation=45)
g.figure.savefig(file_prefix + "_keyword.png", bbox_inches = "tight")
def get_top_n2_words(corpus, n=None):
vec1 = CountVectorizer(ngram_range=(2,2),
max_features=2000).fit(corpus)
bag_of_words = vec1.transform(corpus)
sum_words = bag_of_words.sum(axis=0)
words_freq = [(word, sum_words[0, idx]) for word, idx in
vec1.vocabulary_.items()]
```

```
words_freq =sorted(words_freq, key = lambda x: x[1],
reverse=True)
return words_freq[:n]
top2_words = get_top_n2_words(corpus, n=20)
top2_df = pandas.DataFrame(top2_words)
top2_df.columns=["Bi-gram", "Frequency"]
print(top2_df)
top2_df.to_csv(file_prefix + '_bigrams.csv')
import seaborn as sns
sns.set(rc={'figure.figsize':(13,8)})
h=sns.barplot(x="Bi-gram", y="Frequency", data=top2_df,
palette="Blues_d")
h.set_xticklabels(h.get_xticklabels(), rotation=75)
h.figure.savefig(file_prefix + "_bi-gram.png", bbox_inches = "tight")
def get_top_n3_words(corpus, n=None):
vec1 = CountVectorizer(ngram_range=(3,3),
max_features=2000).fit(corpus)
bag_of_words = vec1.transform(corpus)
sum_words = bag_of_words.sum(axis=0)
words_freq = [(word, sum_words[0, idx]) for word, idx in
vec1.vocabulary_.items()]
words_freq =sorted(words_freq, key = lambda x: x[1],
reverse=True)
return words_freq[:n]
top3_words = get_top_n3_words(corpus, n=20)
top3_df = pandas.DataFrame(top3_words)
top3_df.columns=["Tri-gram", "Frequency"]
print(top3_df)
top3_df.to_csv(file_prefix + '_trigrams.csv')
```

```
import seaborn as sns

sns.set(rc={'figure.figsize':(13,8)})

j=sns.barplot(x="Tri-gram", y="Frequency", data=top3_df,
palette="Blues_d")

j.set_xticklabels(j.get_xticklabels(), rotation=75)

j.figure.savefig(file_prefix + "_tri-gram.png", bbox_inches = "tight")

from sklearn.feature_extraction.text import TfidfTransformer

tfidf_transformer=TfidfTransformer(smooth_idf=True,use_idf=True)

tfidf_transformer.fit(X)

feature_names=cv.get_feature_names()


doc=corpus[ds_count-1]


tf_idf_vector=tfidf_transformer.transform(cv.transform([doc]))

from scipy.sparse import coo_matrix

def sort_coo(coo_matrix):

tuples = zip(coo_matrix.col, coo_matrix.data)

return sorted(tuples, key=lambda x: (x[1], x[0]), reverse=True)

def extract_topn_from_vector(feature_names, sorted_items, topn=25):

sorted_items = sorted_items[:topn]

score_vals = []

feature_vals = []

for idx, score in sorted_items:

score_vals.append(round(score, 3))

feature_vals.append(feature_names[idx])

results= {}

for idx in range(len(feature_vals)):

results[feature_vals[idx]]=score_vals[idx]

return results
```

```
sorted_items=sort_coo(tf_idf_vector.tocoo())
keywords=extract_topn_from_vector(feature_names,sorted_items,25)

print("\nAbstract:")
print(doc)
print("\nKeywords:")
for k in keywords:
print(k,keywords[k])

import csv
with open(file_prefix + 'td_idf.csv', 'w', newline="") as csv_file:
writer = csv.writer(csv_file)
writer.writerow(["Keyword", "Importance"])
for key, value in keywords.items():
writer.writerow([key, value])

from sklearn.ensemble import BaggingClassifier
from sklearn import datasets
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
import numpy as np

def ensembles_accuracy(data):
n_of_class = len(set(data.target))
acc_dict = dict()
X_train, X_test, y_train, y_test = train_test_split(
data.data, data.target, test_size=0.4, random_state=30)
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

```
acc_dict["One classificator"] = gnb.score(X_test, y_test)

bagging = BaggingClassifier(GaussianNB(), n_estimators=10)

bagging.fit(X_train, y_train)

acc_dict["BaggingClassifier"] = bagging.score(X_test, y_test)

cross_score, gnb_cross, Y_pred = cross_validation(data)

acc_dict["Cross-validation"] = cross_score

gnb_bagging = bag(data)

w_bag_score, Ypred_bag, Ytest_bag = weighted(data.data, data.target,
gnb_bagging, n_of_class)

acc_dict["Weighted BaggingClassifier"] = w_bag_score

w_cross_score, Ypred_cross, Ytest_cross = weighted(X_test, y_test,
gnb_cross, n_of_class)

acc_dict["Weighted cross-validation"] = w_cross_score

wm_bag_score = weighted_majority(gnb_bagging, Ypred_bag, Ytest_bag,
n_of_class)

acc_dict["Weighted Majority BaggingClassifier"] = wm_bag_score

wm_cross_score = weighted_majority(gnb_cross, Ypred_cross, Ytest_cross,
n_of_class)

acc_dict["Weighted Majority cross-validation"] = wm_cross_score


return acc_dict


def bag(data):

gnb_bagging = []

for i in range(10):

trainX, testX, trainY, testY = train_test_split(

data.data, data.target, test_size=0.4, random_state=i)

gnb = GaussianNB()

gnb.fit(trainX, trainY)

gnb_bagging.append(gnb)

Y_pred = np.empty((len(testX), 10))

for i in range(10):
```

```
Y_pred[:, i] = gnb_bagging[i].predict(testX)
return gnb_bagging
def weighted(X_test, y_test, gnb_list, n_of_class):
Xtest = X_test[:len(X_test) // 2, :]
Xvalid = X_test[len(X_test) // 2:, :]
Ytest = y_test[:len(X_test) // 2]
Yvalid = y_test[len(X_test) // 2:]
score_list = []
for gnb in gnb_list:
score_list.append(gnb.score(Xvalid, Yvalid))
Ypred = np.empty((len(Xtest), len(gnb_list)))
for i in range(len(gnb_list)):
Ypred[:, i] = gnb_list[i].predict(Xtest)
l = []
for row in range(len(Ypred)):
sum_array = np.zeros((n_of_class, 1))
for col in range(len(Ypred[row])):
for i in range(n_of_class):
if Ypred[row, col] == i:
sum_array[i] += 1 * score_list[col]
m = sum_array.max()
for i in range(len(sum_array)):
if m == sum_array[i]:
l.append(i)
score = sum(np.array(l) == Ytest) / len(Ytest)
return score, Ypred, Ytest
def cross_validation(data):
X_train, X_test, y_train, y_test = train_test_split(
data.data, data.target, test_size=0.4, random_state=30)
step = len(data.data) // 10
```

```
lenght = len(data.data)
gnb_cross = []
for i in range(15, lenght + step, step):
rez_x = np.concatenate((data.data[0:lenght - i, :],
data.data[lenght + step - i:lenght, :]), axis=0)
rez_y = np.concatenate((data.target[0:lenght - i],
data.target[lenght + step - i:lenght]), axis=0)
gnb = GaussianNB()
gnb.fit(rez_x, rez_y)
gnb_cross.append(gnb)
Y_pred = np.empty((len(X_test), 10))
for i in range(10):
Y_pred[:, i] = gnb_cross[i].predict(X_test)
score = sum(np.array(count(Y_pred)) == y_test) / len(y_test)
return score, gnb_cross, Y_pred

def count(data):
l = []
for d in data:
unique, counts = np.unique(d, return_counts=True)
count_dict = dict(zip(counts, unique))
max_key = max(count_dict.keys())
l.append(count_dict[max_key])
return l
```

ДОДАТОК Б

Відомість атестаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	112 с.
	Інші документи	
2.	Презентаційні матеріали	плакатів

.	.	Прізвище та ініціали.	Підп.	Дата	
Розробив		Рогач М.А.			Розробка інтелектуального персонального помічника в управлінні витратами часу на основі методів машинного навчання
Перевірив		Золотухін О. В.			
Н.контр.		Малєєва І.А.			
Затв.		Філатов В.О.			

Шифр групи	Код напр./спец.
СШІм-19-1	122
ХНУРЕ кафедра ШІ	