50.データの入手・整形

解答：

```
import pandas as pd
from sklearn.model_selection import train_test_split

# データの読込
df = pd.read_csv('./newsCorpora_re.csv', header=None, sep='\t', names=['ID', 'TITLE', 'URL', 'PUBLISHER', 'CATEGORY', 'STORY', 'HOSTNAME', 'TIMESTAMP'])

# データの抽出
df = df.loc[df['PUBLISHER'].isin(['Reuters', 'Huffington Post', 'Businessweek', 'Contactmusic.com', 'Daily Mail']), ['TITLE', 'CATEGORY']]

# データの分割
train, valid_test = train_test_split(df, test_size=0.2, shuffle=True, random_state=123, stratify=df['CATEGORY'])
valid, test = train_test_split(valid_test, test_size=0.5, shuffle=True, random_state=123, stratify=valid_test['CATEGORY'])

# データの保存
train.to_csv('./train.txt', sep='\t', index=False)
valid.to_csv('./valid.txt', sep='\t', index=False)
test.to_csv('./test.txt', sep='\t', index=False)

# 事例数の確認
print('【学習データ】')
print(train['CATEGORY'].value_counts())
print('【検証データ】')
print(valid['CATEGORY'].value_counts())
print('【評価データ】')
print(test['CATEGORY'].value_counts())
```

実行結果：

```
【学習データ】
b    4501
e    4235
t    1220
m     728
Name: CATEGORY, dtype: int64
【検証データ】
b    563
e    529
t    153
m     91
Name: CATEGORY, dtype: int64
【評価データ】
b    563
e    530
t    152
m     91
Name: CATEGORY, dtype: int64
```

（結果の一部）

まとめ：

まずデータを読み込み、分類する。

次に PUBLISHER タグを取り出したのは「Reuters」、「Huffington Post」、「Businessweek」、「Contactmusic.com」、「Daily Mail」の行は、'TITLE'，'CATEGORY'情報のみを保持します。

Train_test_split()を 2 回使用して、学習データ 80%、検証データ 10%、評価データ 10%に分割し、タブ区切りで、ファイルに保存する。

value_counts()を使用して、"CATEGORY"列の要素数を表示する。

5 1.特徴量抽出

解答：

```
import numpy as np

train=np.array(train)
valid=np.array(valid)
test=np.array(test)

import re
from sklearn.feature_extraction.text import TfidfVectorizer

def preprocessing(data):
 x=[]
 y=[]
 label={"b":0,"e":1,"t":2,"m":3} #英字ラベルを数値ラベルに変換

 for title,category in data:
  title=re.sub('[0-9]+', '0', title) # title の数字を 0 に変換
  x.append(title.lower())#title の文字を小文字に変換
  y.append(label[category]) #英字ラベルを数値ラベルに変換
 return x,y

X_train,Y_train=preprocessing(train) #X_....store title, Y_....store catagory
X_valid,Y_valid=preprocessing(valid)
X_test,Y_test=preprocessing(test)

tfidfvectorizer=TfidfVectorizer(min_df=0.001)# 最小データ精度の定義、小数点以下 3 桁
X_train=tfidfvectorizer.fit_transform(X_train).toarray()#単語ごとに出現する頻度を計算し、頻度を配列に書き込む
X_valid=tfidfvectorizer.transform(X_valid).toarray()
X_test=tfidfvectorizer.transform(X_test).toarray()

#ベクトルをデータフレームに変換
X_train=pd.DataFrame(X_train,columns=tfidfvectorizer.get_feature_names_out())
X_valid=pd.DataFrame(X_valid,columns=tfidfvectorizer.get_feature_names_out())
X_test=pd.DataFrame(X_test,columns=tfidfvectorizer.get_feature_names_out())

# 特徴量をファイルに保存する
X_train.to_csv("./X_train.txt",sep="\t",index=False)
X_valid.to_csv("./X_valid.txt",sep="\t",index=False)
X_test.to_csv("./X_test.txt",sep="\t",index=False)
```

実行結果：（結果の一部）

Column1	Column2
TITLE	CATEGORY
Mila Kunis' with Ashton Kutcher on Two And A Half Men, as they're 'expecting ...	e
Netflix Orders Jane Fonda, Lily Tomlin Sitcom, 'Grace And Frankie'	e
UPDATE 1-S&P lifts outlook on UK's top credit rating, but warns on EU exit	b
US STOCKS SNAPSHOT-S&P 500 ends above 2000 for first time	b
UPDATE 2-'X-Men' director Bryan Singer accused of drugging, raping teen	e
Peaches Geldof Funeral Set, But We Still Don't Know How She Died	e
Employers, Republicans see pro-union slant in US NLRB action	b
Pandora's rise in ad revenue helps beat expectations	b
Jay Z - Jay Z And Beyonce Trek On Track To Be Second Best-selling Tour Per ...	e
PRECIOUS-Gold above 6-week low; set for first monthly decline this year	b
Lead singer of heavy metal band As I Lay Dying sentenced to six years in prison ...	e
Lindsay Lohan slips into a wedding dress for 2 Broke Girls with Kat Dennings	e
Brad Pitt and Angelina Jolie to star together in first film in nearly a decade	e
WRAPUP 2-Amazon snaps up live video startup Twitch for $970 million	t
Lorillard Reaches Record on Fresh Reynolds Takeover Speculation	b
Kim Kardashian Is A Blonde Again (UPDATE)	e
Aspirin to Prevent Heart Attack Should Be Limited, FDA Says (2)	m
US STOCKS SNAPSHOT-Wall St ends higher; healthcare sector helps	b
Sir Mick Jagger - The Rolling Stones supporting Sir Mick Jagger	e

まとめ：

TF-IDF は、ファイルセットまたはコーパス内のファイルの 1 つに対する単語の重要度を評価する統計的な方法です。単語の重要性は、ファイルに現れる回数に比例して増加しますが、コーパスに現れる頻度に比例して減少します。

TF-IDF の主な思想は、ある単語が 1 つの文章に現れる頻度が TF が高く、他の文章にはあまり現れない場合、この単語やフレーズは分類に適したカテゴリ区分能力を持っていると考えている。

TfidfVectorizer.fit_transform() メソッド：引数にテキストデータを入力し、TF-IDF を使用して単語をベクトルに変換する。戻り値は scipy 型。

TfidfVectorizer.get_feature_names() メソッド：特徴量（単語）を取得する。

scipy.sparse.csr_matrix.toarray() メソッド：scipy オブジェクトを numpy オブジェクトに変換する。

np.savetxt() 関数：numpy オブジェクトをファイルに保存することができる。第一引数に、ファイル名を指定し、第二引数に保存したい numpy 配列データを指定する。

52.学習

解答：

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression(max_iter=200)
```

```
model.fit(X_train,Y_train)# train ファイル内のデータを用いてロジスティック回帰モデルを適用し、訓練を行う
```

実行結果：

```
LogisticRegression(max_iter=200)
```

まとめ：

クラスを import してロジステック回帰モデルを実装する。学習データをモデルに渡して学習を行う。

53.予測

解答：

```
def score(model,X):
 pred=model.predict(X)
 pred_proba=model.predict_proba(X)
 return pred,pred_proba
```

```
X_pred_valid,X_pred_proba_valid=score(model,X_valid)#使用上一题中准备好的逻辑回归预测模型对 valid 数据集进行预测，
X_pred_test,X_pred_proba_test=score(model,X_test)#使用上一题中准备好的逻辑回归预测模型对 test 数据集进行预测，
```

実行結果：

```
[2 1 0 ... 3 0 1] [[0.03342492 0.00177938 0.96216485 0.00263085]
 [0.03859564 0.94345968 0.01114548 0.0067992 ]
 [0.80881505 0.11111363 0.01870903 0.06136229]
 ...
 [0.2385465  0.09664149 0.08179416 0.58301785]
 [0.72216426 0.12517354 0.07295479 0.07970742]
 [0.32417943 0.55079134 0.09014124 0.034888  ]]
```

まとめ：

LogisticRegression.predict()メソッド：カテゴリのラベルを予測する。
LogisticRegression.predict_proba()メソッド：各カテゴリの予測確率を求める。

54.正確率の計測

解答：

```
from sklearn.metrics import accuracy_score
```

```
print(f"valid_accuracy: {accuracy_score(Y_valid, X_pred_valid)}")# accuracy _score 関数直接計算予測結果精度
print(f"valid_accuracy: {accuracy_score(Y_test, X_pred_test)}")
```

実行結果：

```
valid_accuracy: 0.8712574850299402
valid_accuracy: 0.8854790419161677
```

まとめ：

sklearn.metrics.accuracy_score()関数：第一引数にラベルを指定し、第二引数にモデルの予測結果を指定する。

55. 混同行列の作成

解答：

```
from sklearn.metrics import confusion_matrix

print(f"confusion matrix of X_pred_valid:\n{confusion_matrix(Y_valid, X_pred_valid)}\n")
print(f"confusion matrix of X_pred_test:\n{confusion_matrix(Y_test, X_pred_test)}")
```

実行結果：

```
confusion matrix of X_pred_valid:
[[515  33  13   2]
 [ 14 509   3   3]
 [ 26  28  97   2]
 [ 17  26   5  43]]

confusion matrix of X_pred_test: :
[[528  19  14   2]
 [ 11 516   2   1]
 [ 36  25  90   1]
 [ 12  29   1  49]]
```

まとめ：

混同行列：

　　混同行列は誤差行列とも呼ばれ、精度評価を表す標準形式であり、n行n列の行列形式で表される。具体的な評価指標には、全体精度、製図精度、ユーザ精度などがあり、これらの精度指標は画像分類の精度を異なる側面から反映している。人工知能では、混同行列（confusion matrix）は視覚化ツールであり、特に監督学習に用いられ、監督学習がない場合は一般的にマッチング行列と呼ばれている。画像精度評価では、主に分類結果と実際の測定値を比較するために用いられ、分類結果の精度を混同行列に表示することができる。混同行列は、各実測画像要素の位置と分類を分類画像中の対応する位置と分類と比較することによって計算される。

混同行列の各列は予測カテゴリを表し、各列の総数はそのカテゴリとして予測されたデータの数を表し、各行はデータの真のホーム・カテゴリを表し、各行のデータ総数はそのカテゴリのデータ・インスタンスの数を表します。各列の数値は、真のデータがクラスの数として予測されていることを示します：第 1 行第 1 列の 43 は、43 の実際に第 1 クラスに帰属するインスタンスが第 1 クラスとして予測されていることを示し、同様に、第 1 行第 2 列の 2 は、2 つの実際に第 1 クラスに帰属するインスタンスが第 2 クラスとして誤って予測されていることを示します。

		预测		
		类1	类2	类3
实际	类1	43	2	0
	类2	5	45	1
	类3	2	3	49

56. 適合率，再現率，F 1 スコアの計測

解答：

```
from sklearn.metrics import precision_score, recall_score, f1_score

def metrics(y_data, y_pred, ave=None):
  precision = precision_score(y_data, y_pred, average=ave)
  recall = recall_score(y_data, y_pred, average=ave)
  F1_score = f1_score(y_data, y_pred, average=ave)
  result="Precision{}\nRecall {}\nF1_score{}\n".format(precision, recall, F1_score)
  return result

print(f"【Category order】{model.classes_}\n\n{metrics(Y_test, X_pred_test)}")
print("【micro-average】\n", metrics(Y_valid, X_pred_valid, "macro"))
print("【macro-average】\n", metrics(Y_test, X_pred_test, "micro"))
```

実行結果：

```
【Category order】[0 1 2 3]

Precision: [0.89948893 0.87606112 0.8411215  0.9245283 ]
Recall : [0.93783304 0.97358491 0.59210526 0.53846154]
F1_score: [0.91826087 0.92225201 0.69498069 0.68055556]

【micro-average】
Precision: 0.8591025985730797
Recall : 0.7458624171054344
F1_score: 0.7845435280897002

【macro-average】
Precision: 0.8854790419161677
Recall : 0.8854790419161677
F1_score: 0.8854790419161677
```

まとめ：

sklearn.metrics.precision_score()関数：適合率が求まる。第一引数にラベルを指定し、第二引数にモデルの予測値を指定する。
sklearn.metrics.recall_score 関数：再現率が求まる。第一引数にラベルを指定し、第二引数にモデルの予測値を指定する。
sklearn.metrics.f1_score 関数：F1 スコアが求まる。第一引数にラベルを指定し、第二引数にモデルの予測値を指定する。
マクロ平均：カテゴリごとに評価指標を求め、全体の評価指標を平均する方法。上記 3 つの関数の average 引数に”macro”を指定することで求まる。
マイクロ平均：混合行列全体で TP、FP、FN を算出して、適合率、再現率、F 値を計算する方法。上記 3 つの関数の average 引数に”micro”を指定することで求まる。

57. 特徴量の重みの確認

解答：

```
import numpy as np

features = X_train.columns.values
for c, coef in zip(model.classes_, model.coef_):
  top_10 = pd.DataFrame(features[np.argsort(-
 coef)[:10]], columns=[f"Feature weights 10 most important（class: {c}）
 "], index=[i for i in range(1, 11)])
  worst_10 = pd.DataFrame(features[np.argsort(coef)[:10]], columns=[f"Feature weigh
 ts 10 least important（class: {c}）"], index=[i for i in range(1, 11)])
  print( top_10, "\n"),
  print(worst_10, "\n", "-"*70)
```

実行結果：

	Feature weights 10 most important（class: 0）
1	bank
2	stocks
3	fed
4	ecb
5	china
6	euro
7	obamacare
8	oil
9	yellen
10	ukraine

	Feature weights 10 most important（class: 1）
1	kardashian
2	chris
3	her
4	movie
5	paul
6	she
7	he
8	star
9	miley
10	thrones

	Feature weights 10 least important（class: 0）
1	video
2	she
3	ebola
4	her
5	the
6	microsoft
7	star
8	virus
9	apple
10	aereo

	Feature weights 10 least important（class: 1）
1	us
2	google
3	update
4	ceo
5	china
6	study
7	gm
8	facebook
9	apple
10	billion

	Feature weights 10 most important (class: 2)
1	google
2	facebook
3	apple
4	microsoft
5	climate
6	gm
7	nasa
8	tesla
9	comcast
10	earth

	Feature weights 10 most important (class: 3)
1	ebola
2	study
3	cancer
4	mers
5	drug
6	fda
7	cases
8	cigarettes
9	cdc
10	medical

	Feature weights 10 least important (class: 2)
1	stocks
2	her
3	fed
4	percent
5	drug
6	american
7	ukraine
8	still
9	cancer
10	men

	Feature weights 10 least important (class: 3)
1	facebook
2	gm
3	ceo
4	apple
5	bank
6	sales
7	game
8	google
9	deal
10	twitter

まとめ：

Pd.DataFrame(data, columns, index, dtype, copy)

data：データのセット（darray、series、map、lists、dict など）。

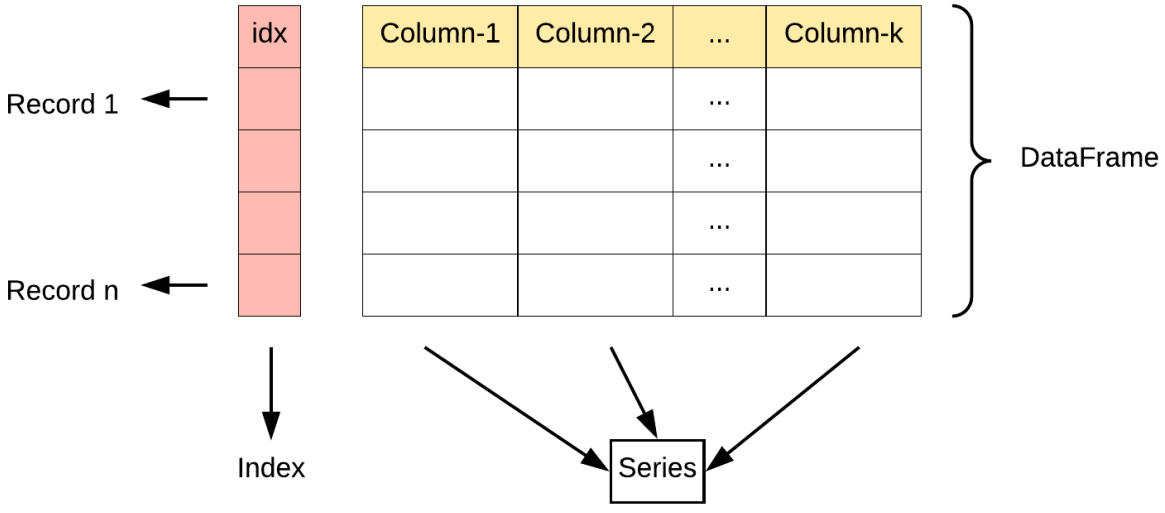
index：インデックス値、または行ラベルと呼ばれます。

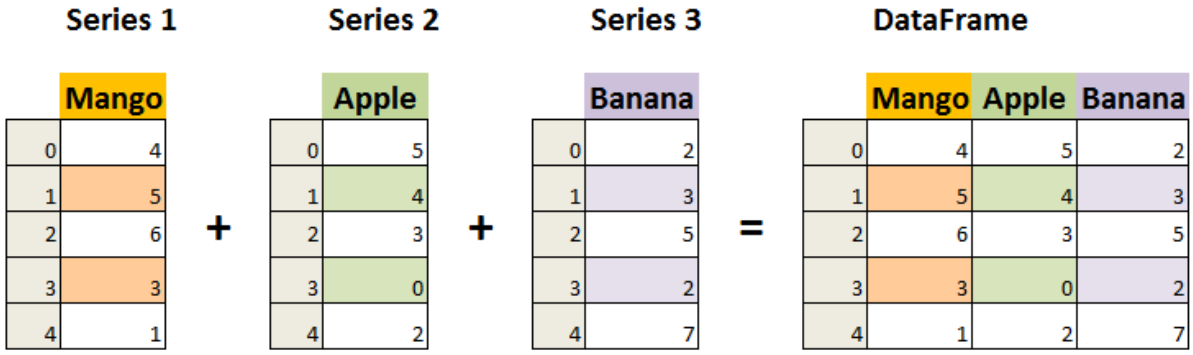
columns：カラムラベル、デフォルトは RangeIndex（0、1、2、…、n）です。

dtype：データ型。

copy：コピーデータ、デフォルトは False です。

DataFrame は、テーブル型のデータ構造であり、各カラムは異なる値タイプ（数値、文字列、ブール値）であることができる順序付けられたカラムのセットを含んでいます。DataFrame には行インデックスと列インデックスがあり、Series からなる辞書（1 つのインデックスで共通）として見ることができます。





58.正規化パラメータの変更

解答：

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

#モデル構築、学習（関数）
def LR_model_fit(x_data, y_data, c):

  model = LogisticRegression(C=c)#c：正規化係数λの逆数、float タイプ、既定値は 1.0 です。正の浮動小数点数でなければなりません。SVM のように、小さい数値ほど強い正規化を表す。
  model.fit(x_data, y_data)
  return model

#学習済みモデルを用いて予測する（関数）
def LR_pred(x_data, y_data, model):
  Y_pred_data = model.predict(x_data)
  accuracy = accuracy_score(Y_pred_data, y_data)
  return accuracy

train_acc = []
valid_acc = []
test_acc = []

#正則化パラメータ
c_list = np.linspace(0.001, 0.1, 5)#c_list は等差数列です
[0.001   0.02575 0.0505  0.07525 0.1   ]

for c in c_list:# 異なる正規化パラメータを試す
  model = LR_model_fit(X_train, Y_train, c)
  pre_train = LR_pred(X_train, Y_train, model)
  pre_valid = LR_pred(X_valid, Y_valid, model)
  pre_test = LR_pred(X_test, Y_test, model)

  train_acc.append(pre_train)
  valid_acc.append(pre_valid)
  test_acc.append(pre_test)
```

```
    print(f"【正則化パラメータ：{c}】\n")
    print(f"train_accuracy：{pre_train}")
    print(f"valid_accuracy：{pre_valid}")
    print(f"test_accuracy：{pre_test}\n")

#可視化
plt.plot(c_list, train_acc, label="train", marker="o")
plt.plot(c_list, valid_acc, label="valid", marker="o")
plt.plot(c_list, test_acc, label="test", marker="o")

plt.legend()
plt.grid(True)
plt.xlabel("Regularization")
plt.ylabel("Accuracy")
plt.show()
```

実行結果：

【正則化パラメータ：0.001】

train_accuracy：0.5057094721078248
valid_accuracy：0.5029940119760479
test_accuracy：0.5082335329341318

【正則化パラメータ：0.025750000000000002】

train_accuracy：0.7686259827779858
valid_accuracy：0.7634730538922155
test_accuracy：0.7687125748502994

【正則化パラメータ：0.0505】

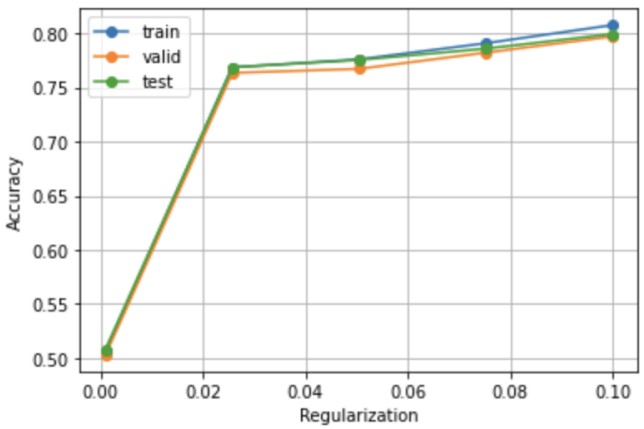
train_accuracy：0.7760202171471359
valid_accuracy：0.7672155688622755
test_accuracy：0.7754491017964071

【正則化パラメータ：0.07525000000000001】

train_accuracy：0.7909022837888431
valid_accuracy：0.782185628742515
test_accuracy：0.7859281437125748

【正則化パラメータ：0.1】

train_accuracy：0.8076563084986896
valid_accuracy：0.7971556886227545
test_accuracy：0.7994011976047904



まとめ：

LogisticRegression、全部で14のパラメータがあり、このサイト（https://scikitlearn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html）には公式の解説があります。

59.ハイパーパラメータの探索

解答：

```
from sklearn.model_selection import GridSearchCV

params = {"C": [0.001, 0.005, 10]}

# グリッドサーチを行う
gs_model = GridSearchCV(LogisticRegression(max_iter=1500),params, cv=5, verbose=1)
gs_model.fit(X_train, Y_train)

#最適なモデルを取得する
best_gs_model = gs_model.best_estimator_
print("\ntrain_score: {:.2%}".format(best_gs_model.score(X_train, Y_train)))
print("valid_score: {:.2%}".format(best_gs_model.score(X_valid, Y_valid)))
print("test_score: {:.2%}".format(best_gs_model.score(X_test, Y_test)))
```

実行結果：

```
Fitting 5 folds for each of 3 candidates, totalling 15 fits

train_score: 96.27%
valid_score: 87.87%
test_score: 88.55%
```

まとめ：

sklearn.model_selection.GridSearchCV()クラス：モデルの最適なハイパーパラメータを求めることができる。第一引数に、学習させたいモデルのインスタンスを指定する。第二引数に、使用したいハイパーパラメータの候補を辞書型で指定する。第三引数に、学習データを何分割して交差検証を行うのかを指定する。
GridSearchCV.fit()メソッド：モデルの学習を行う。第一引数に学習データを指定する。第二引数にラベルを指定する。
GridSearchCV.best_estimator_属性：最も精度の高かったハイパーパラメータのモデルを返す。
GridSearchCV.score()メソッド：第一引数に学習データを指定する。第二引数にモデルの予測値を指定する。デフォルトで正解率を返す。