#**CLASSIFICATION : TOPIC MODELING -TRUE vs FALSE vs OTHER vs MIXTURE :**

Membres: Hadjoudja Bachir (21811363), Zeggar Rym (21909615), Bendahmane Rania (21811387), Labiad Youcef (21710780).

```python
#les imports utilisés dans ce notebook
import sys
from numpy import vstack
import pandas as pd
from pandas import read_csv
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from torch.utils.data import Dataset
from torch.utils.data import DataLoader
from torch.utils.data import random_split
from torch import Tensor
from torch.nn import Linear
from torch.nn import ReLU
from torch.nn import Sigmoid
from torch.nn import Module
from torch.optim import SGD
from torch.nn import BCELoss
from torch.nn.init import kaiming_uniform_
from torch.nn.init import xavier_uniform_
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from pandas import read_csv
from sklearn.feature_extraction.text import TfidfVectorizer
from  sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
import pickle
import string

import nltk
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
from nltk import word_tokenize
from sklearn.pipeline import Pipeline

# librairie spacy
import spacy

# librairies de gensim
import gensim
from gensim.utils import simple_preprocess
from gensim.models import CoherenceModel
from gensim.models import Phrases
from gensim.models.phrases import Phraser
from gensim import corpora
from gensim import models

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

stop_words = set(stopwords.words('english'))

import sklearn
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_fscore_support as score
#from sklearn.linear  import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

# Importation des différentes librairies utiles pour le notebook
#Sickit learn met régulièrement à jour des versions et
#indique des futurs warnings.
#ces deux lignes permettent de ne pas les afficher.
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import sys
import pandas as pd
import numpy as np
import sklearn
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

#Sickit learn met régulièrement à jour des versions et indique des
futurs warnings.
#ces deux lignes permettent de ne pas les afficher.
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)

from sklearn.metrics._plot.confusion_matrix import
ConfusionMatrixDisplay
# fonction qui affiche le classification report et la matrice de
confusion
from sklearn import metrics
from sklearn.metrics import confusion_matrix , ConfusionMatrixDisplay
from sklearn.metrics import classification_report



import re
import spacy
import gensim
import string
import nltk
from nltk.corpus import stopwords
from nltk.corpus import wordnet
import gensim
from gensim.utils import simple_preprocess
from gensim.models import Phrases
from gensim.models.phrases import Phraser
from gensim import corpora
from gensim import models
nltk.download('wordnet')
nltk.download('stopwords')
import gensim
from gensim import corpora
import gensim
from gensim.models import Phrases
from gensim.models.phrases import Phraser
stop_words = set(stopwords.words('english'))

from sklearn.model_selection import GridSearchCV
from sklearn.datasets import fetch_20newsgroups
```

```python
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from tabulate import tabulate


from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
import time
import numpy as np

from sklearn.metrics._plot.confusion_matrix import
ConfusionMatrixDisplay
# fonction qui affiche le classification report et la matrice de
confusion
from sklearn import metrics
from sklearn.metrics import confusion_matrix , ConfusionMatrixDisplay
from sklearn.metrics import classification_report

import sklearn
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.linear_model  import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier


# Importation des différentes librairies utiles pour le notebook

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import sys
import pandas as pd
import numpy as np
import sklearn
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

#Sickit learn met régulièrement à jour des versions et indique des
futurs warnings.
#ces deux lignes permettent de ne pas les afficher.
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

autorisation

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

```
Mounted at /content/gdrive/
```

chemin spécifique Google Drive

```
my_local_drive='/content/gdrive/My Drive/Colab Notebooks'
# Ajout du path pour les librairies, fonctions et données
sys.path.append(my_local_drive)
# Se positionner sur le répertoire associé
%cd $my_local_drive
%ls
```

```
%pwd
```

```
/content/gdrive/My Drive/Colab Notebooks
 avecscaler.pkl
 Classification_de_données_textuelles2023.ipynb
'Copie de TRUE_vs_FALSE_vs_OTHER_vs_MIXTURE_TOPIC_MODELLING.ipynb'
 Dataset/
 firstmodel.pkl
'Ingénierie_des_données_textuelles2023 (1).ipynb'
 Ingénierie_des_données_textuelles2023.ipynb
 MyNLPUtilities.py
 newsTrain2.csv
 newsTrain_-_newsTrain.csv
 penguins.csv
 penguins.csv.1
 pkl_modelNB.sav
 Premières_Classifications.ipynb
'Projet ML FakeNEWS_TRUE FALSE_TEXT.ipynb'
'Projet ML FakeNEWS_TRUE FALSE_TEXT+TITRE.ipynb'
'Projet ML FakeNEWS_TRUE FALSE_TITRE.ipynb'
 __pycache__/
 ReviewsLabelled.csv
 ReviewsLabelled.csv.1
 ReviewsLabelled.csv.2
 ReviewsLabelled.csv.3
 ReviewsLabelled.csv.4
 ReviewsLabelled.csv.5
 SentimentModel.pkl
 StopWordsFrench.csv
 StopWordsFrench.csv.1
 StopWordsFrench.csv.2
 StopWordsFrench.csv.3
 StopWordsFrench.csv.4
 Topics_extraction.ipynb
 TP1_HAI817I.ipynb
 TP2_HAI817I.ipynb
'TRUE FALSE_TOPIC_MODELLING.ipynb'
'TRUE FALSE_vs_OTHER.ipynb'
'TRUE FALSE_vs_OTHER_TOPIC_MODELLING.ipynb'
'TRUE_vs_FALSE_vs_OTHER_vs_MIXTURE_TOPIC_MODELLING (1).ipynb'
```

```
 TRUE_vs_FALSE_vs_OTHER_vs_MIXTURE_TOPIC_MODELLING.ipynb
 Visualisation_Donnees_2D_3D.ipynb
```

```
{"type":"string"}
```

La fonction qui sera utilisée pour les prétraitements: MyCleanText

- Mettre le texte en minuscule
- Se débarasser des stopwords
- Se débarasser des nombres
- Stemmatisation
- Lemmatisation ..

La fonction MyshowAllScores prend le y_test et le y_predict, affiche l'accuracy et le classification report avec la matrice de confusion.

```
#...................................................Fonction
MyCleanText ..........................................................
...................
# mettre en minuscule
#enlever les stopwords
#se debarasser des nombres
#stemmatisation
#lemmatisation
#.....................................................................
......................................................................
...........


nltk.download('wordnet')
nltk.download('stopwords')
nltk.download('punkt')
#liste des stopwords en anglais
stop_words = set(stopwords.words('english'))

def MyCleanText(X,
                lowercase=False, #mettre en minuscule
                removestopwords=False, #supprimer les stopwords
                removedigit=False, #supprimer les nombres
                getstemmer=False, #conserver la racine des termes
                getlemmatisation=False #lemmatisation des termes
                ):
  #conversion du texte d'entrée en chaîne de caractères
    sentence=str(X)
    #suppression des caractères spéciaux
    sentence = re.sub(r'[^\w\s]',' ', sentence)
    # suppression de tous les caractères uniques
    sentence = re.sub(r'\s+[a-zA-Z]\s+', ' ', sentence)
    # substitution des espaces multiples par un seul espace
    sentence = re.sub(r'\s+', ' ', sentence, flags=re.I)
    # decoupage en mots
    tokens = word_tokenize(sentence)
    if lowercase:
          tokens = [token.lower() for token in tokens]

    # suppression ponctuation
    table = str.maketrans('', '', string.punctuation)
    words = [token.translate(table) for token in tokens]

    # suppression des tokens non alphabetique ou numerique
    words = [word for word in words if word.isalnum()]

    # suppression des tokens numerique
    if removedigit:
        words = [word for word in words if not word.isdigit()]

    # suppression des stopwords
    if removestopwords:
        words = [word for word in words if not word in stop_words]

    # lemmatisation
    if getlemmatisation:
        lemmatizer=WordNetLemmatizer()
        words = [lemmatizer.lemmatize(word)for word in words]


    # racinisation
    if getstemmer:
        ps = PorterStemmer()
        words=[ps.stem(word) for word in words]

    sentence= ' '.join(words)

    return sentence

def MyshowAllScores(y_test,y_pred):
  classes= np.unique(y_test)
  print("Accuracy : %0.3f"%(accuracy_score(y_test,y_pred)))
  print("Classification Report")
  print(classification_report(y_test,y_pred,digits=5))
  cnf_matrix = confusion_matrix(y_test,y_pred)
  disp=ConfusionMatrixDisplay(cnf_matrix,display_labels=classes)
  disp.plot()
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

- La classe TextNormalizer qui contiendra la fonction MyCleanText.
- Fit_transform de mon corpus propre.

```
#..................................................Etape 1 :
prétraitement du
texte ................................................................
...................................
#...................................................Class
TextNormalizer  ......................................................
.......................
#fit_transform de mon corpus propre
#.....................................................................
......................................................................
...........

from sklearn.base import BaseEstimator, TransformerMixin

class TextNormalizer(BaseEstimator, TransformerMixin):
    def __init__(self,
                 removestopwords=False, # suppression des stopwords
                 lowercase=False,# passage en minuscule
                 removedigit=False, # supprimer les nombres
                 getstemmer=False,# racinisation des termes
                 getlemmatisation=False # lemmatisation des termes
                ):

        self.lowercase=lowercase
        self.getstemmer=getstemmer
        self.removestopwords=removestopwords
        self.getlemmatisation=getlemmatisation
        self.removedigit=removedigit

    def transform(self, X, **transform_params):
        # Nettoyage du texte
        X=X.copy() # pour conserver le fichier d'origine
        return [MyCleanText(text,lowercase=self.lowercase,
                            getstemmer=self.getstemmer,
                            removestopwords=self.removestopwords,
                            getlemmatisation=self.getlemmatisation,
                            removedigit=self.removedigit) for text in
X]

    def fit(self, X, y=None, **fit_params):
        return self

    def fit_transform(self, X, y=None, **fit_params):
        return self.fit(X).transform(X)

    def get_params(self, deep=True):
        return {
            'lowercase':self.lowercase,
            'getstemmer':self.getstemmer,
            'removestopwords':self.removestopwords,
            'getlemmatisation':self.getlemmatisation,
            'removedigit':self.removedigit
        }

    def set_params (self, **parameters):
        for parameter, value in parameters.items():
            setattr(self,parameter,value)
        return self
```

##**Etape 1 : Préparer les données**

- Load et preparer les données à partir des 2 fichiers csv

```
dftrain1 = pd.read_csv("/content/gdrive/MyDrive/Colab
Notebooks/newsTrain2.csv", names=['id','text','title','rating'],
header=0,sep=',', encoding='utf8')
dftrain1.reset_index(drop = True, inplace = True)

dftrain2 = pd.read_csv("/content/gdrive/MyDrive/Colab
Notebooks/newsTrain_-_newsTrain.csv",
names=['id','text','title','rating'], header=0,sep=',',
encoding='utf8')
dftrain2.reset_index(drop = True, inplace = True)


# concaténer les deux dataframes en ajoutant les lignes du deuxième à
la fin du premier
dftrainbase = pd.concat([dftrain1, dftrain2], ignore_index=True)
dftrain=dftrainbase


print("Echantillon de mon dataset \n")
print(dftrain.sample(n=10))
print("\n")
print("Quelques informations importantes \n")
dftrain.info()
print("\n")
X_text=dftrain["text"]
X_title=dftrain["title"]


print("le texte est")
display(X_text)
print("\n")
print("le titre est")
display(X_title)
print("\n")
y=dftrain.iloc[0:,-1]
print("voici la dernière case de rating")
display(y)
print("\n")
print("la taille de Xt est",X_text.shape)
print("\n")
print(" y EST " ,y)
print("\n")
y = y.str.lower()
print("Les valeurs de true et false sont:\n", y.value_counts())
```

```
Echantillon de mon dataset

            id                                               text  \
21    4461868e  If you still think mail-in ballots are a safe ...
1283  2d0f41d8  WHEELING — At least $53 million a week in ille...
902   d9cd4895  The hypocritical Lib Dems want to ignore the r...
2176  b49f74e3  According to the latest FOX News poll Presiden...
577   a5e0c051  Milwaukee emerged as America's fourth-most imp...
2455  71f6e8fd  This is an archived article and the informatio...
2386  ea95b7e8  PUPILS aged just five have been accused of sex...
2145  e44784a7  Kindly Share This Story: By David Royal A Twit...
1897  3c8eac5c  TALLAHASSEE, Fla. (WCTV) - Wednesday, Presiden...
2038  f2cf61ca  An 18-year-old United States citizen has been ...

                                                  title   rating
21    Dem Senator Knope Caught With 8000 Mail-In Bal...    FALSE
1283  Rep. David McKinley Calls for Increase in Bord...  mixture
902   Climate Alarmists Caught Manipulating Temperat...    FALSE
2176  Send This to Anyone Who Wants to Know WTF Is U...  mixture
577   Kremlin: Putin’s 'heaven' remark symbolism, Ru...     TRUE
2455  NASA releases time-lapse of the disappearing A...  mixture
2386  Pervs' aged five School sex crime claims trebl...    other
2145  History-making Olympics' Theresa May leads tri...    FALSE
1897  Florida Senator Rick Scott on Paris Climate Ag...  mixture
2038                                                NaN     TRUE


Quelques informations importantes

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2528 entries, 0 to 2527
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      2528 non-null   object
 1   text    2528 non-null   object
 2   title   2482 non-null   object
 3   rating  2528 non-null   object
dtypes: object(4)
memory usage: 79.1+ KB


le texte est

0       Distracted driving causes more deaths in Canad...
1       Missouri politicians have made statements afte...
2       Home Alone 2: Lost in New York is full of viol...
3       But things took a turn for the worse when riot...
4       It’s no secret that Epstein and Schiff share a...
                              ...
2523    More than four million calls to the taxman are...
2524    More under-18s are being taken to court for se...
2525    The Government’s much vaunted Help to Buy Isa ...
2526    The late Robin Williams once called cocaine “G...
2527    The late Robin Williams once called cocaine “G...
Name: text, Length: 2528, dtype: object



le titre est

0       You Can Be Fined $1,500 If Your Passenger Is U...
1           Missouri lawmakers condemn Las Vegas shooting
2       CBC Cuts Donald Trump's 'Home Alone 2' Cameo O...
3       Obama’s Daughters Caught on Camera Burning US ...
4       Leaked Visitor Logs Reveal Schiff’s 78 Visits ...
                              ...
2523    Taxman fails to answer four million calls a ye...
2524    Police catch 11-year-olds being used to sell d...
2525    Help to Buy Isa scandal: 500,000 first-time bu...
2526             A coke-snorting generation of hypocrites
2527             A coke-snorting generation of hypocrites
Name: title, Length: 2528, dtype: object



voici la dernière case de rating

0         FALSE
1       mixture
2       mixture
3         FALSE
4         FALSE
         ...
2523       TRUE
2524       TRUE
2525      FALSE
```

```
2526       TRUE
2527       TRUE
Name: rating, Length: 2528, dtype: object
```

```
la taille de Xt est (2528,)
```

```
 y EST  0         FALSE
1       mixture
2       mixture
3         FALSE
4         FALSE
         ...
2523       TRUE
2524       TRUE
2525      FALSE
2526       TRUE
2527       TRUE
Name: rating, Length: 2528, dtype: object
```

```
Les valeurs de true et false sont:
 false      1156
mixture     716
true        422
other       234
Name: rating, dtype: int64
```

Le jeu de données étant déséquilibré, on a pensé à appliquer le downsampling pour équilibrer nos données. on séléctionne des lignes aléatoirement de TRUE,FALSE, OTHER et MIXTURE de telle sorte que le nombre de lignes de chacune soit = au nbr de lignes de la classe qui a le plus petit nombre de lignes. et on mélange le DataFrame.

```python
# Compter le nombre d'observations dans chaque catégorie
false_count = dftrain['rating'].value_counts()['FALSE']
mixture_count = dftrain['rating'].value_counts()['mixture']
true_count = dftrain['rating'].value_counts()['TRUE']
other_count = dftrain['rating'].value_counts()['other']

# Trouver le nombre minimum d'observations parmi les catégories
min_count = min(false_count, mixture_count, true_count, other_count)

# Sous-échantillonner les catégories pour équilibrer les quantités
false_sampled = dftrain[dftrain['rating'] ==
'FALSE'].sample(min_count, random_state=42)
mixture_sampled = dftrain[dftrain['rating'] ==
'mixture'].sample(min_count, random_state=42)
true_sampled = dftrain[dftrain['rating'] == 'TRUE'].sample(min_count,
random_state=42)
other_sampled = dftrain[dftrain['rating'] ==
'other'].sample(min_count, random_state=42)
print(false_sampled.shape)
print(true_sampled.shape)

# Concaténer les échantillons pour obtenir un nouveau dataframe
équilibré
dftrain = pd.concat([false_sampled, mixture_sampled, true_sampled,
other_sampled])

# Mélanger aléatoirement les données
dftrain = dftrain.sample(frac=1, random_state=42)

X_text=dftrain.iloc[0:,1:2]
X_title=dftrain.iloc[0:,2:3]

print("le texte est")
display(X_text)
print("le titre est")
display(X_title)

y=dftrain.iloc[0:,-1]
print("le y est")
display(y)
print("la taille de X_text est",X_text.shape)
print("la taille de y_train est " ,y.shape)
print("les valeurs de TRUE et FALSE maintenant sont
" ,y.value_counts())
```

```
(234, 4)
(234, 4)
le texte est

                                                   text
2504  Hillary Clinton's plane passes over Manhattan ...
261   Rashida Tlaib is busy at work during a nationa...
46    Natural News The oldest magazine in the United...
1546  Ministers are undermining trust in foreign aid...
1781  Today the Education Policy Institute's Indepen...
...                                                 ...
1543  The bombshell claim comes from over 20 hours o...
422   This is a rush transcript from Fox News Sunday...
1102  The use of cocaine in Britain has doubled in s...
2382  A ndy Murray served up an ace to John Inverdal...
1325  Though the whole world relies on RT-PCR to “di...

[936 rows x 1 columns]

le titre est
```

```
                                                  title
2504  Hillary Clinton Boards The Climate Crisis Trai...
261   Tlaib Files Lawsuit to Ban the American Flag i...
46    Still think 5G is harmless? Scientific America...
1546  Ministers are undermining trust in foreign aid...
1781  Apocalyptic Sea-Level Rise—Just a Thing of the...
...                                                 ...
1543  Breaking: Breonna Taylor’s boyfriend says SHE ...
422   Pruitt defends decision to withdraw from Paris...
1102  Britain's cocaine use doubles in last seven ye...
2382  Andy Murray aces John Inverdale after BBC pres...
1325   COVID19 PCR Tests are Scientifically Meaningless

[936 rows x 1 columns]

le y est

2504    mixture
261       FALSE
46        FALSE
1546       TRUE
1781       TRUE
         ...
1543      FALSE
422     mixture
1102      other
2382    mixture
1325      FALSE
Name: rating, Length: 936, dtype: object

la taille de X_text est (936, 1)
la taille de y_train est  (936,)
les valeurs de TRUE et FALSE maintenant sont  mixture    234
FALSE      234
TRUE       234
other      234
Name: rating, dtype: int64
```

Installation des librairies qu'on utilise pour le topic modeling

```
!pip install pyLDAvis
!pip install -U gensim
!pip install --upgrade numpy
!pip uninstall numpy
!pip install numpy
```

```
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Collecting pyLDAvis
  Downloading pyLDAvis-3.4.1-py3-none-any.whl (2.6 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.6/2.6 MB 28.1 MB/s eta
```

```
0:00:00
ent already satisfied: joblib>=1.2.0 in
/usr/local/lib/python3.10/dist-packages (from pyLDAvis) (1.2.0)
Requirement already satisfied: gensim in
/usr/local/lib/python3.10/dist-packages (from pyLDAvis) (4.3.1)
Collecting funcy
  Downloading funcy-2.0-py2.py3-none-any.whl (30 kB)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.10/dist-packages (from pyLDAvis) (3.1.2)
Requirement already satisfied: scipy in
/usr/local/lib/python3.10/dist-packages (from pyLDAvis) (1.10.1)
Collecting numpy>=1.24.2
  Downloading numpy-1.24.3-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 54.3 MB/s eta
0:00:00
anylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.3 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.3/12.3 MB 61.6 MB/s eta
0:00:00
ent already satisfied: scikit-learn>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from pyLDAvis) (1.2.2)
Requirement already satisfied: numexpr in
/usr/local/lib/python3.10/dist-packages (from pyLDAvis) (2.8.4)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.10/dist-packages (from pyLDAvis) (67.7.2)
Requirement already satisfied: tzdata>=2022.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=2.0.0->pyLDAvis)
(2023.3)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=2.0.0->pyLDAvis)
(2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas>=2.0.0->pyLDAvis)
(2.8.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0-
>pyLDAvis) (3.1.0)
Requirement already satisfied: smart-open>=1.8.1 in
/usr/local/lib/python3.10/dist-packages (from gensim->pyLDAvis)
(6.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->pyLDAvis)
(2.1.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2-
>pandas>=2.0.0->pyLDAvis) (1.16.0)
Installing collected packages: funcy, numpy, pandas, pyLDAvis
  Attempting uninstall: numpy
    Found existing installation: numpy 1.22.4
    Uninstalling numpy-1.22.4:
      Successfully uninstalled numpy-1.22.4
  Attempting uninstall: pandas
    Found existing installation: pandas 1.5.3
    Uninstalling pandas-1.5.3:
      Successfully uninstalled pandas-1.5.3
ERROR: pip's dependency resolver does not currently take into account
all the packages that are installed. This behaviour is the source of
the following dependency conflicts.
tensorflow 2.12.0 requires numpy<1.24,>=1.22, but you have numpy
1.24.3 which is incompatible.
numba 0.56.4 requires numpy<1.24,>=1.18, but you have numpy 1.24.3
which is incompatible.
google-colab 1.0.0 requires pandas~=1.5.3, but you have pandas 2.0.1
which is incompatible.
Successfully installed funcy-2.0 numpy-1.24.3 pandas-2.0.1 pyLDAvis-
3.4.1
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: gensim in
/usr/local/lib/python3.10/dist-packages (4.3.1)
Requirement already satisfied: numpy>=1.18.5 in
/usr/local/lib/python3.10/dist-packages (from gensim) (1.24.3)
Requirement already satisfied: smart-open>=1.8.1 in
/usr/local/lib/python3.10/dist-packages (from gensim) (6.3.0)
Requirement already satisfied: scipy>=1.7.0 in
/usr/local/lib/python3.10/dist-packages (from gensim) (1.10.1)
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: numpy in
/usr/local/lib/python3.10/dist-packages (1.24.3)
Found existing installation: numpy 1.24.3
Uninstalling numpy-1.24.3:
  Would remove:
    /usr/local/bin/f2py
    /usr/local/bin/f2py3
    /usr/local/bin/f2py3.10
    /usr/local/lib/python3.10/dist-packages/numpy-1.24.3.dist-info/*
    /usr/local/lib/python3.10/dist-packages/numpy.libs/libgfortran-
040039e1.so.5.0.0

/usr/local/lib/python3.10/dist-packages/numpy.libs/libopenblas64_p-r0-
15028c96.3.21.so
    /usr/local/lib/python3.10/dist-packages/numpy.libs/libquadmath-
96973f99.so.0.0.0
    /usr/local/lib/python3.10/dist-packages/numpy/*
Proceed (Y/n)? y
  Successfully uninstalled numpy-1.24.3
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Collecting numpy
```

```
  Using cached numpy-1.24.3-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
Installing collected packages: numpy
ERROR: pip's dependency resolver does not currently take into account
all the packages that are installed. This behaviour is the source of
the following dependency conflicts.
tensorflow 2.12.0 requires numpy<1.24,>=1.22, but you have numpy
1.24.3 which is incompatible.
numba 0.56.4 requires numpy<1.24,>=1.18, but you have numpy 1.24.3
which is incompatible.
google-colab 1.0.0 requires pandas~=1.5.3, but you have pandas 2.0.1
which is incompatible.
Successfully installed numpy-1.24.3
```

Dans cette cellule on trouve les définitions de toutes les fonctions qu'on utilise pour le topic modeling:

- MyCleanTextsforLDA pour le nettoyage du text, elle renvoie les bigrammes, corpus bow, corpus tfIdf, le dictionnaire des mots
- dominant_topic qui extrait le topic dominant du corpus
- format_topics_sentence elle renvoie un DataFrame qui contient le topic dominant de chaque document du corpus, ses keywords, et le pourcentage de sa contribution dans le document
- compute_coherences_values pour calculer la cohérence
- MyGridSearchLda elle applique différentes valeurs pour num_topics, eta et alpha et renvoie un DataFrame trié par ordre décroissant de cohérence
- get_best_coherence_values pour tester différents nombre de topics et choisir le meilleur compromis

```
nlp = spacy.load("en_core_web_sm", disable=['parser', 'ner'])
#nlp = spacy.load('en', disable=['parser', 'ner'])

def MyCleanTextsforLDA(texts,
                      min_count=1, # nombre d'apparitions minimale
pour un bigram
                      threshold=2,
                      no_below=1, # nombre minimum d'apparitions pour
être dans le dictionnaire
                      no_above=0.5, # pourcentage maximal (sur la
taille totale du corpus) pour filtrer
                      stop_words=stop_words
                      ):

    allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']
    sentences=texts.copy()

    # suppression des caractères spéciaux
    sentences = [re.sub(r'[^\w\s]', ' ', str(sentence)) for sentence
in sentences]
```

```
    # suppression de tous les caractères uniques
    sentences = [re.sub(r'\s+[a-zA-Z]\s+', ' ', str(sentence)) for
sentence in sentences]
    # substitution des espaces multiples par un seul espace
    sentences = [re.sub(r'\s+', ' ', str(sentence), flags=re.I) for
sentence in sentences]

    # conversion en minuscule et split des mots dans les textes
    sentences = [sentence.lower().split() for sentence in sentences]

    # utilisation de spacy pour ne retenir que les allowed_postags
    texts_out = []
    for sent in sentences:
        if len(sent) < (nlp.max_length): # si le texte est trop grand
            doc = nlp(" ".join(sent))
            texts_out.append(" ".join([token.lemma_ for token in doc
if token.pos_ in allowed_postags]))
        else:
            texts_out.append(sent)
    sentences=texts_out

    # suppression des stopwords
    words =[[word for word in simple_preprocess(str(doc)) if word not
in stop_words] for doc in sentences]


    # recherche des bigrammes
    bigram = Phrases(words, min_count, threshold,delimiter=' ')
    bigram_phraser = Phraser(bigram)


    # sauvergarde des tokens et des bigrammes
    bigram_token = []
    for sent in words:
        bigram_token.append(bigram_phraser[sent])


    # creation du vocabulaire
    dictionary = gensim.corpora.Dictionary(bigram_token)


    # il est possible de filtrer des mots en fonction de leur
occurrence d'apparitions
    #dictionary.filter_extremes(no_below, no_above)
    # et de compacter le dictionnaire
    # dictionary.compactify()
    corpus = [dictionary.doc2bow(text) for text in bigram_token]

    # recuperaction du tfidf plutôt que uniquement le bag of words
    tfidf = models.TfidfModel(corpus)
    corpus_tfidf = tfidf[corpus]

    return corpus, corpus_tfidf, dictionary, bigram_token





def dominant_topic(model,corpus,num_topics):
    # recuperation du vecteur associé
    # creation d'un dictionnaire pour stocker les résultats
    topic_dictionary = {i: [] for i in range(num_topics)}
    topic_probability_scores =
model.get_document_topics(corpus,minimum_probability=0.000)
    if len(topic_probability_scores) ==1 : # il y a plusieurs
predictions on recupere la premiere
        row=topic_probability_scores[0]
    else: # on concatene les predictions
        tab=[]
        for j in range (len(topic_probability_scores)):
            tab.append(topic_probability_scores[j])
        row=tab
    # parcours des différents topics
    for (topic_num, prop_topic) in row:
            topic_dictionary[topic_num].append(prop_topic)
    # tri pour avoir le plus grand en premier
    list_proba=topic_dictionary
    topic_dictionary=sorted(topic_dictionary,
key=topic_dictionary.get,reverse = True)
    return topic_dictionary, list_proba


def format_topics_sentences(ldamodel, corpus, texts):
    # Initialisation du dataframe de sortie
    sent_topics_df = pd.DataFrame()

    # Recherche le topic dominant pour chaque document
    for i, row_list in enumerate(ldamodel[corpus]):
        row = row_list[0] if ldamodel.per_word_topics else row_list

        row = sorted(row, key=lambda x: (x[1]), reverse=True)
        # Donne le topic dominant, le pourcentage de contribution
        # et les mots clés pour chaque document
        for j, (topic_num, prop_topic) in enumerate(row):
            if j == 0:  # => topic dominant
                wp = ldamodel.show_topic(topic_num)
                topic_keywords = ", ".join([word for word, prop in
wp])
                #sent_topics_df =
sent_topics_df.append(pd.Series([int(topic_num), round(prop_topic,4),
topic_keywords]), ignore_index=True)
                temp_df = pd.DataFrame([[int(topic_num),
round(prop_topic, 4), topic_keywords]], columns=['topic_dominant',
'pourcentage_contrib', 'topic_keywords'])
                sent_topics_df = pd.concat([sent_topics_df, temp_df],
ignore_index=True)
            else:
                break
    sent_topics_df.columns = ['topic_dominant', 'pourcentage_contrib',
'topic_keywords']

    # Ajout du texte original à la fin de la sortie
    contents = pd.Series(texts)
    sent_topics_df = pd.concat([sent_topics_df, contents], axis=1)
    return(sent_topics_df)



# ce code est inspiré de
# https://towardsdatascience.com/evaluate-topic-model-in-python-
latent-dirichlet-allocation-lda-7d57484bb5d0
def compute_coherence_values(corpus, dictionary, listtokens, k, alpha,
eta):

    lda_model = gensim.models.LdaMulticore(corpus=corpus,
                                           id2word=dictionary,
                                           num_topics=k,
                                           random_state=100,
                                           chunksize=100,
                                           passes=10,
                                           alpha=alpha,
                                           eta=eta,
                                           per_word_topics=True)

    coherence_model_lda = CoherenceModel(model=lda_model,
texts=listtokens, dictionary=dictionary, coherence='c_v')

    return coherence_model_lda.get_coherence()

def MyGridSearchLda
(corpus,listtokens,dictionnary,nb_topics,alpha,eta,verbose=1):

    grid = {}
    model_results = {'topics': [],
                     'alpha': [],
                     'eta': [],
                     'coherence': []
                    }
    # iteration sur le nombre de topics
    for k in nb_topics:
        # iteration sur les valeurs d'alpha
        for a in alpha:
            # iteration sur les valeurs de eta
            for e in eta:
                # calcul du score de coherence
                cv = compute_coherence_values(corpus=corpus,
                                              dictionary=dictionary,
                                              listtokens=listtokens,
                                              k=k, alpha=a, eta=e)
                if verbose==1:
                    print ('topics:', k, ' alpha: %0.3f  eta: %0.3f
coherence: %0.3f'%(a,e,cv))

                # sauvegarde des résultats
                model_results['topics'].append(k)
                model_results['alpha'].append(a)
                model_results['eta'].append(e)
                model_results['coherence'].append(cv)

    df_result=pd.DataFrame(model_results)
    df_result = df_result.sort_values('coherence',ascending=False)
    df_result.reset_index(drop=True, inplace=True)
    return df_result



def get_best_coherence_values(corpus, dictionary, listtokens, start=5,
stop=15, step=2):
    coherence_values = []
    model_list = []
    for num_topics in range(start, stop, step):
        lda_model = gensim.models.LdaMulticore(corpus=corpus,
                                           id2word=dictionary,
                                           num_topics=num_topics,
                                           random_state=100,
                                           chunksize=100,
                                           passes=10,
                                           per_word_topics=True)

        coherence_model_lda = CoherenceModel(model=lda_model,
texts=listtokens, dictionary=dictionary, coherence='c_v')
        model_list.append(lda_model)
        coherence_values.append(coherence_model_lda.get_coherence())
    return model_list, coherence_values
```

On concatène les deux colonnes text et titre de note DataFrame dftrain

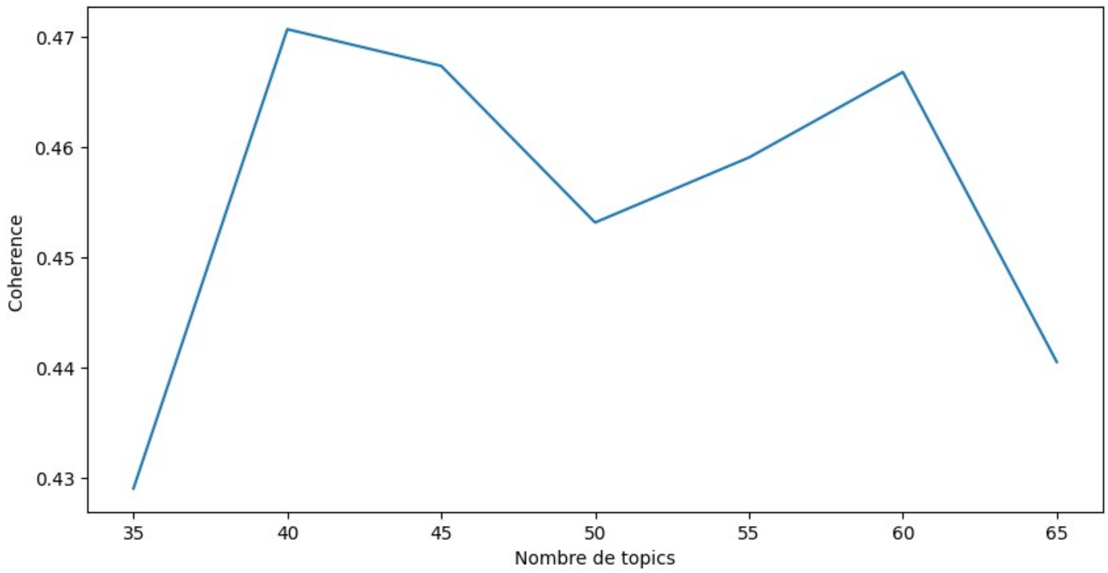
```
text_title = dftrain.apply(lambda x : '{}
{}'.format(x['text'],x['title']),axis=1)
dftrain['text_title'] = text_title
```

On commence par applique la fonction MyCleantextsforLDA sur la colonne text_title (combinaison des 2 colonnes) et puis on teste différentes valeurs pour pouvoir trouver le bon nombre de topics

```
import pyLDAvis
import pyLDAvis.gensim_models as gensimvis

dftrain.reset_index(drop = True, inplace = True)
display(dftrain)
dftrain_txt_ttl = dftrain.text_title
stop = stopwords.words('english')

# enrichissement des stopwords
stop.extend(['always','try','go','get','make','would','really',

'like','came','got','article','creativecommons','license','http'])
corpus, corpus_tfidf, dictionary,
bigram_token=MyCleanTextsforLDA(dftrain_txt_ttl)



# test sur un intervalle de 6 à 15 en utilisant le corpus Bow
start=35
stop=70
step=5
model_list, coherence_values =
get_best_coherence_values(dictionary=dictionary,
                                                        corpus=corpus,


listtokens=bigram_token,
                                                        start=start,
stop=stop, step=step)


# affichage du graphe associé à la recherche du nombre de topics
plt.figure(figsize=(10,5))
x = range(start, stop, step)
plt.plot(x, coherence_values)
plt.xlabel("Nombre de topics")
plt.ylabel("Coherence ")
plt.show()
```

```
           id                                                text  \
0    c9a710dc  Hillary Clinton's plane passes over Manhattan ...
1    a7b20877  Rashida Tlaib is busy at work during a nationa...
```

```
2    fb721890  Natural News The oldest magazine in the United...
3    ed8a09ac  Ministers are undermining trust in foreign aid...
4    f454e71d  Today the Education Policy Institute's Indepen...
..        ...                                               ...
931  3886ead8  The bombshell claim comes from over 20 hours o...
932  da3319cc  This is a rush transcript from Fox News Sunday...
933  7b9e930d  The use of cocaine in Britain has doubled in s...
934  48026a71  A ndy Murray served up an ace to John Inverdal...
935  31d33510  Though the whole world relies on RT-PCR to “di...

                                                 title   rating  \
0    Hillary Clinton Boards The Climate Crisis Trai...  mixture   
1    Tlaib Files Lawsuit to Ban the American Flag i...    FALSE   
2    Still think 5G is harmless? Scientific America...    FALSE   
3    Ministers are undermining trust in foreign aid...     TRUE   
4    Apocalyptic Sea-Level Rise—Just a Thing of the...     TRUE   
..                                                 ...      ...   
931  Breaking: Breonna Taylor’s boyfriend says SHE ...    FALSE   
932  Pruitt defends decision to withdraw from Paris...  mixture   
933  Britain's cocaine use doubles in last seven ye...    other   
934  Andy Murray aces John Inverdale after BBC pres...  mixture   
935   COVID19 PCR Tests are Scientifically Meaningless    FALSE   

                                            text_title  
0    Hillary Clinton's plane passes over Manhattan ...  
1    Rashida Tlaib is busy at work during a nationa...  
2    Natural News The oldest magazine in the United...  
3    Ministers are undermining trust in foreign aid...  
4    Today the Education Policy Institute's Indepen...  
..                                                 ...  
931  The bombshell claim comes from over 20 hours o...  
932  This is a rush transcript from Fox News Sunday...  
933  The use of cocaine in Britain has doubled in s...  
934  A ndy Murray served up an ace to John Inverdal...  
935  Though the whole world relies on RT-PCR to “di...  

[936 rows x 5 columns]
```



40 semble une bonne valeur pour le nombre de topics

- On entraîne notre modèle LDA avec ce nombre de topics, et puis on affiche les topcis avec les mots associés + leurs poids dans chaque topic
- On affiche par la suite la cohérence et la perplexité (qui est censée être petite)
- On applique la méthode format_topics_sentences sur le modèle lda entraîné et notre colonne de text+titre pour avoir un tableau de topic dominant + son pourcentage de contribution et les mots-clés de chaque topic

```
num_words=20 # nombre de mots par topics


num_topic_best=40

lda_model = gensim.models.ldamulticore.LdaMulticore(
                           corpus=corpus,
                           num_topics=num_topic_best,
                           id2word=dictionary,
                           chunksize=100,
                           workers=7,
                           passes=10,
                           random_state=100,
                           eval_every = 1,
                           per_word_topics=True)

print ("Affichage des ",num_topic_best," différents topics pour le
corpus TF-IDF :")

for idx, topic in lda_model.print_topics(-1,num_words):
    print('Topic : {} Words : {}'.format(idx, topic))
```

```
coherence_model_lda = CoherenceModel(model=lda_model,
texts=bigram_token, dictionary=dictionary,
                                     coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('Cohérence  : ', coherence_lda)

print('Perplexité : ', lda_model.log_perplexity(corpus))
```

```
df_topic_sents_keywords = format_topics_sentences(ldamodel=lda_model,
corpus=corpus, texts=dftrain_txt_ttl)
```

```
display(df_topic_sents_keywords)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during
thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

```
Affichage des  40  différents topics pour le corpus TF-IDF :
Topic : 0 Words : 0.010*"say" + 0.007*"state" + 0.007*"open carry" +
0.004*"work" + 0.003*"year old" + 0.002*"year" + 0.002*"make" +
0.002*"state require" + 0.002*"income" + 0.002*"law" + 0.002*"claim" +
0.002*"smoker lung" + 0.002*"people cease" + 0.002*"black lung" +
0.002*"election" + 0.002*"way" + 0.002*"get" + 0.002*"need" +
0.002*"fire" + 0.002*"water"
Topic : 1 Words : 0.011*"say" + 0.005*"people" + 0.004*"year" +
0.004*"country" + 0.003*"need" + 0.003*"chick pizza" + 0.003*"report"
+ 0.003*"party work" + 0.002*"also" + 0.002*"public" + 0.002*"pizza
party" + 0.002*"state" + 0.002*"government" + 0.002*"well" +
0.002*"many" + 0.002*"business" + 0.002*"wedding" +
0.002*"information" + 0.002*"see" + 0.002*"receipt attach"
Topic : 2 Words : 0.012*"say" + 0.004*"people" + 0.004*"year" +
0.004*"report" + 0.003*"make" + 0.003*"go" + 0.003*"sea level" +
0.003*"get" + 0.003*"time" + 0.003*"work" + 0.003*"vote" +
0.003*"child" + 0.003*"week" + 0.002*"woman" + 0.002*"change" +
0.002*"know" + 0.002*"government" + 0.002*"include" + 0.002*"country"
+ 0.002*"think"
Topic : 3 Words : 0.006*"illegal alien" + 0.006*"alien" +
0.006*"criminal alien" + 0.006*"state" + 0.005*"percent" +
0.004*"offense" + 0.004*"percent percent" + 0.004*"prison" +
0.003*"sanctuary policy" + 0.003*"assault" + 0.003*"time" +
0.003*"report" + 0.003*"government" + 0.002*"murder" + 0.002*"jail" +
0.002*"commit" + 0.002*"spanish" + 0.002*"que" + 0.002*"percent
arrest" + 0.002*"sexual assault"
Topic : 4 Words : 0.008*"say" + 0.005*"make" + 0.005*"navy" +
0.004*"see" + 0.004*"cut" + 0.003*"police" + 0.003*"take" +
0.003*"also" + 0.003*"year" + 0.003*"new" + 0.003*"slave" +
0.003*"officer" + 0.003*"sailor" + 0.002*"event" + 0.002*"heat wave" +
0.002*"ship" + 0.002*"get" + 0.002*"service" + 0.002*"climate change"
+ 0.002*"time"
Topic : 5 Words : 0.019*"getty image" + 0.006*"gold medal" +
0.005*"say" + 0.005*"child" + 0.005*"day getty" + 0.005*"image bronze"
+ 0.003*"image gold" + 0.003*"image win" + 0.003*"final day" +
0.003*"celebrate win" + 0.003*"win medal" + 0.002*"image" + 0.002*"day
clive" + 0.002*"win bronze" + 0.002*"win gold" + 0.002*"final image" +
0.002*"rose getty" + 0.002*"time" + 0.002*"hide caption" +
0.002*"gender neutral"
Topic : 6 Words : 0.011*"mental health" + 0.010*"say" + 0.005*"lung
cancer" + 0.005*"ex cnrp" + 0.004*"year" + 0.004*"patient" +
0.003*"people" + 0.003*"service" + 0.003*"health" + 0.003*"make" +
0.003*"nhs" + 0.003*"public health" + 0.003*"also" + 0.003*"find" +
0.002*"health care" + 0.002*"covid patient" + 0.002*"last year" +
0.002*"global warming" + 0.002*"day" + 0.002*"diagnose"
Topic : 7 Words : 0.004*"witness" + 0.004*"trial" + 0.003*"make" +
0.002*"show" + 0.002*"crown court" + 0.002*"find" + 0.002*"cause" +
0.002*"lie coverup" + 0.002*"say certainty" + 0.002*"increase
atmospheric" + 0.002*"arrest" + 0.002*"case" + 0.002*"report say" +
0.002*"crack" + 0.002*"watchdog say" + 0.002*"magistrate court" +
0.002*"inspectorate" + 0.002*"defendant acquit" + 0.002*"come" +
0.002*"say"
Topic : 8 Words : 0.020*"say" + 0.004*"food stamp" + 0.003*"work" +
0.003*"service" + 0.003*"child" + 0.003*"tell" + 0.003*"state" +
0.002*"hear say" + 0.002*"year" + 0.002*"fick" + 0.002*"service
industry" + 0.002*"lottery" + 0.002*"work permit" + 0.002*"go" +
0.002*"system" + 0.002*"many people" + 0.002*"call" + 0.002*"leave" +
0.002*"come" + 0.002*"last week"
Topic : 9 Words : 0.007*"saharan heatwave" + 0.007*"scorch saharan" +
0.007*"relief scorch" + 0.007*"seek relief" + 0.007*"heatwave picture"
+ 0.005*"say" + 0.004*"climate change" + 0.002*"gun" + 0.002*"law" +
0.002*"winter sport" + 0.002*"process" + 0.002*"temperature" +
0.002*"year" + 0.002*"next year" + 0.002*"olympic winter" +
0.002*"know" + 0.002*"also" + 0.002*"last week" + 0.002*"man" +
0.001*"due process"
Topic : 10 Words : 0.005*"hurricane" + 0.004*"storm" + 0.003*"tell" +
0.003*"people" + 0.002*"show" + 0.002*"also" + 0.002*"increase" +
0.002*"see" + 0.002*"actually" + 0.002*"say" + 0.002*"fact" +
0.002*"day" + 0.002*"report" + 0.002*"climate" + 0.002*"damage" +
0.002*"poison" + 0.002*"expect" + 0.002*"year" + 0.002*"come" +
0.002*"use"
Topic : 11 Words : 0.009*"short seller" + 0.006*"people" +
0.006*"hedge fund" + 0.005*"junior doctor" + 0.005*"gamestop stock" +
0.004*"stock price" + 0.004*"go" + 0.003*"doctor" + 0.003*"asuu" +
0.003*"say" + 0.003*"stock market" + 0.003*"price go" + 0.003*"amount
money" + 0.002*"make" + 0.002*"strike" + 0.002*"hundred thousand" +
0.002*"stock go" + 0.002*"bet company" + 0.002*"hold stock" +
0.002*"buy gamestop"
```

```
Topic : 12 Words : 0.007*"say" + 0.003*"child" + 0.002*"use" +
0.002*"email address" + 0.002*"cent" + 0.002*"teacher shortage" +
0.002*"maybe" + 0.002*"marry adopt" + 0.002*"man woman" +
0.002*"leave" + 0.002*"fact" + 0.002*"year old" + 0.002*"catastrophe"
+ 0.002*"enter valid" + 0.002*"school" + 0.002*"message verifyerror" +
0.002*"rise degree" + 0.001*"put pressure" + 0.001*"staff train" +
0.001*"take"
Topic : 13 Words : 0.007*"say" + 0.006*"climate change" + 0.004*"use"
+ 0.004*"get" + 0.003*"time" + 0.002*"look" + 0.002*"video" +
0.002*"vaccine" + 0.002*"elisa" + 0.002*"time fast" + 0.002*"world" +
0.002*"year" + 0.002*"make" + 0.002*"think" + 0.002*"right" +
0.002*"win" + 0.002*"case" + 0.002*"professional agitator" +
0.001*"back" + 0.001*"live"
Topic : 14 Words : 0.006*"say" + 0.004*"minimum unit" + 0.004*"contain
unit" + 0.004*"say drunk" + 0.004*"alcohol previous" + 0.004*"unit
pricing" + 0.004*"say drink" + 0.004*"cost least" + 0.004*"people
drink" + 0.003*"come" + 0.002*"public school" + 0.002*"counsel say" +
0.002*"principal sinclair" + 0.002*"school district" + 0.002*"candy
cane" + 0.002*"elkhorn public" + 0.002*"government" + 0.002*"report" +
0.002*"people" + 0.002*"socioeconomic status"
Topic : 15 Words : 0.003*"say" + 0.003*"guillotine" + 0.003*"bill" +
0.003*"pcr test" + 0.003*"people" + 0.003*"also" + 0.003*"government"
+ 0.003*"year" + 0.003*"school" + 0.003*"vote" + 0.003*"company" +
0.002*"new" + 0.002*"change" + 0.002*"cut" + 0.002*"need" +
0.002*"question" + 0.002*"process" + 0.002*"program" + 0.002*"bank" +
0.002*"covid"
Topic : 16 Words : 0.009*"co" + 0.008*"say" + 0.006*"email address" +
0.004*"arrhenius" + 0.004*"water vapour" + 0.004*"enter valid" +
0.004*"message verifyerror" + 0.004*"assumption" + 0.002*"increase
global" + 0.002*"lorius" + 0.002*"report say" + 0.002*"newsletter
message" + 0.002*"verifyerror message" + 0.002*"event update" +
0.002*"thank sign" + 0.002*"update independent" + 0.002*"offer event"
+ 0.002*"independent read" + 0.002*"verifyerror like" + 0.002*"read
privacy"
Topic : 17 Words : 0.016*"say" + 0.009*"state" + 0.008*"care facility"
+ 0.006*"long term" + 0.006*"facility" + 0.005*"death" + 0.004*"job
plan" + 0.004*"nursing home" + 0.003*"case" + 0.003*"plan" +
0.003*"least" + 0.003*"center" + 0.003*"case death" + 0.003*"people" +
0.003*"year" + 0.003*"report" + 0.003*"include" + 0.003*"school board"
+ 0.003*"datum" + 0.003*"number"
Topic : 18 Words : 0.007*"say" + 0.004*"type diabetes" +
0.004*"people" + 0.004*"work" + 0.003*"attack" + 0.003*"go" +
0.003*"isis" + 0.003*"much" + 0.003*"also" + 0.003*"take" +
0.002*"vote" + 0.002*"state" + 0.002*"increase" + 0.002*"government" +
0.002*"year" + 0.002*"even" + 0.002*"last year" + 0.002*"happen" +
0.002*"country" + 0.002*"world"
Topic : 19 Words : 0.008*"say" + 0.004*"state" + 0.003*"even" +
0.003*"muslim woman" + 0.002*"people" + 0.002*"work" + 0.002*"woman" +
0.002*"election" + 0.002*"year" + 0.002*"trump supporter" +
0.002*"never concede" + 0.002*"stop count" + 0.002*"vote counting" +
0.002*"medium" + 0.002*"number" + 0.002*"far" + 0.002*"point" +
0.002*"ct" + 0.002*"district" + 0.001*"muslim"
Topic : 20 Words : 0.009*"say" + 0.008*"vaccine" + 0.006*"study" +
0.005*"year" + 0.005*"people" + 0.004*"go" + 0.004*"make" +
0.004*"global warming" + 0.003*"change" + 0.003*"worker" +
0.003*"look" + 0.003*"year old" + 0.002*"cervical cancer" +
0.002*"group" + 0.002*"report" + 0.002*"show" + 0.002*"cancer" +
0.002*"give" + 0.002*"adverse event" + 0.002*"see"
Topic : 21 Words : 0.015*"say" + 0.006*"think" + 0.005*"police" +
0.004*"get" + 0.004*"people" + 0.004*"see" + 0.004*"go" +
0.003*"process" + 0.003*"district" + 0.003*"also" + 0.003*"map" +
0.003*"change" + 0.003*"take" + 0.003*"time" + 0.003*"point" +
0.003*"hear" + 0.003*"way" + 0.003*"know" + 0.003*"even" +
0.002*"work"
Topic : 22 Words : 0.009*"say" + 0.005*"polio" + 0.005*"vaccine" +
0.004*"people" + 0.004*"year" + 0.003*"make" + 0.003*"also" +
0.003*"time" + 0.003*"morad say" + 0.002*"cancer cell" +
0.002*"disease" + 0.002*"new" + 0.002*"university" + 0.002*"use" +
0.002*"outbreak" + 0.002*"virus" + 0.002*"cell" + 0.002*"student" +
0.002*"cancer" + 0.002*"tell"
Topic : 23 Words : 0.006*"say" + 0.005*"police" + 0.005*"human right"
+ 0.004*"time" + 0.004*"child" + 0.004*"officer" + 0.004*"report" +
0.003*"family" + 0.003*"take" + 0.003*"government" + 0.003*"state" +
0.003*"also" + 0.003*"follow" + 0.003*"people" + 0.003*"day" +
0.003*"work" + 0.002*"school" + 0.002*"right" + 0.002*"country" +
0.002*"shoot"
Topic : 24 Words : 0.008*"say" + 0.006*"wage" + 0.006*"minimum wage" +
0.004*"increase minimum" + 0.004*"worker" + 0.003*"state" +
0.003*"work" + 0.003*"child" + 0.003*"raise minimum" +
0.003*"increase" + 0.003*"people" + 0.003*"sea level" + 0.002*"accord"
+ 0.002*"support" + 0.002*"hunger" + 0.002*"wage hour" + 0.002*"find"
+ 0.002*"starvation" + 0.002*"also" + 0.002*"job"
Topic : 25 Words : 0.014*"virus" + 0.008*"case" + 0.007*"country" +
0.006*"vaccine" + 0.006*"people" + 0.005*"test" + 0.005*"also" +
0.004*"rate" + 0.004*"make" + 0.004*"time" + 0.004*"coronavirus" +
0.004*"work" + 0.003*"result" + 0.003*"go" + 0.003*"infection" +
0.003*"say" + 0.003*"see" + 0.003*"need" + 0.003*"find" +
0.003*"covid"
Topic : 26 Words : 0.007*"say" + 0.003*"vacuum cleaner" + 0.002*"also"
+ 0.002*"use" + 0.002*"time" + 0.002*"climate change" + 0.002*"carbon
dioxide" + 0.002*"atmospheric co" + 0.002*"control knob" + 0.002*"take
great" + 0.002*"truth sentencing" + 0.002*"chemical mouthwash" +
0.002*"make" + 0.001*"virus cause" + 0.001*"law" + 0.001*"long
history" + 0.001*"ensure" + 0.001*"join court" + 0.001*"hear case" +
0.001*"pro health"
Topic : 27 Words : 0.011*"say" + 0.008*"year" + 0.005*"health visitor"
+ 0.003*"child" + 0.003*"cost" + 0.003*"police" + 0.003*"first time" +
0.003*"people" + 0.003*"number" + 0.003*"think" + 0.002*"show" +
0.002*"health visit" + 0.002*"health secretary" + 0.002*"today pledge"
+ 0.002*"child health" + 0.002*"last year" + 0.002*"investigation" +
0.002*"witch hunt" + 0.002*"mueller investigation" + 0.002*"prison
sentence"
Topic : 28 Words : 0.005*"also" + 0.004*"migrant" + 0.004*"year" +
0.003*"say" + 0.003*"state" + 0.003*"email" + 0.003*"government" +
0.003*"make" + 0.003*"teacher" + 0.003*"malware" + 0.003*"official
say" + 0.003*"time" + 0.003*"country" + 0.002*"scam" + 0.002*"border"
+ 0.002*"take" + 0.002*"see" + 0.002*"border patrol" + 0.002*"number"
+ 0.002*"vaccine"
Topic : 29 Words : 0.003*"zafar" + 0.003*"say" + 0.003*"allegation" +
0.003*"happen" + 0.003*"industry" + 0.002*"harassment" + 0.002*"work"
+ 0.002*"false" + 0.002*"child sacrifice" + 0.002*"woman" +
0.002*"year" + 0.002*"global elite" + 0.002*"email protect" +
0.002*"also" + 0.002*"child" + 0.002*"party" + 0.002*"time" +
0.002*"face" + 0.002*"case" + 0.002*"nature"
Topic : 30 Words : 0.010*"say" + 0.004*"people" + 0.004*"year" +
0.004*"abortion" + 0.004*"vote" + 0.003*"woman" + 0.003*"last year" +
0.003*"bill" + 0.003*"amazon rainforest" + 0.003*"problem" +
0.002*"also" + 0.002*"people die" + 0.002*"help" + 0.002*"state" +
0.002*"happen" + 0.002*"rise" + 0.002*"think" + 0.002*"country" +
0.002*"termination" + 0.002*"climate"
Topic : 31 Words : 0.007*"campaign" + 0.006*"russian government" +
0.005*"trump campaign" + 0.005*"election" + 0.005*"investigation" +
0.005*"office" + 0.005*"special counsel" + 0.003*"report" +
0.003*"government" + 0.003*"candidate trump" + 0.003*"evidence" +
0.003*"information" + 0.003*"document" + 0.003*"time" +
0.003*"intelligence" + 0.003*"year" + 0.003*"product contain" +
0.003*"investigation establish" + 0.003*"attorney work" +
0.003*"material protect"
Topic : 32 Words : 0.004*"say" + 0.003*"need" + 0.003*"long" +
0.003*"also" + 0.003*"year" + 0.003*"people" + 0.003*"nhs" +
0.003*"crime" + 0.003*"go" + 0.003*"increase co" + 0.002*"take" +
0.002*"police officer" + 0.002*"service" + 0.002*"street" +
0.002*"independence referendum" + 0.002*"image getty" +
0.002*"percent" + 0.002*"first" + 0.002*"come" + 0.002*"health
insurance"
Topic : 33 Words : 0.006*"year" + 0.006*"say" + 0.006*"change" +
0.005*"climate change" + 0.004*"sea level" + 0.004*"happen" +
0.003*"last year" + 0.003*"time" + 0.003*"water" + 0.003*"go" +
0.002*"see" + 0.002*"ocean" + 0.002*"rise" + 0.002*"storm" +
0.002*"carbon dioxide" + 0.002*"know" + 0.002*"come" + 0.002*"warm
water" + 0.002*"climate scientist" + 0.002*"public charge"
Topic : 34 Words : 0.010*"low pay" + 0.008*"year" + 0.006*"pay" +
0.004*"make" + 0.004*"work" + 0.004*"last year" + 0.003*"phillip" +
0.003*"say" + 0.003*"student" + 0.002*"tweet" + 0.002*"worker low" +
0.002*"subsequent decade" + 0.002*"blunder" + 0.002*"build wall" +
0.002*"temperature increase" + 0.002*"well" + 0.002*"first" +
0.002*"record" + 0.002*"great" + 0.002*"year decade"
Topic : 35 Words : 0.006*"absentee ballot" + 0.006*"rejection rate" +
0.006*"high education" + 0.006*"say" + 0.004*"excellence framework" +
0.003*"student" + 0.003*"reject ballot" + 0.002*"record number" +
0.002*"year" + 0.002*"state law" + 0.002*"people" +
0.002*"presidential election" + 0.002*"state" + 0.002*"education" +
0.002*"election" + 0.002*"increase" + 0.002*"work" + 0.002*"poll
worker" + 0.002*"correctly fill" + 0.002*"ballot rejection"
Topic : 36 Words : 0.019*"say" + 0.008*"year" + 0.005*"also" +
0.004*"people" + 0.004*"make" + 0.004*"go" + 0.003*"even" +
0.003*"climate change" + 0.003*"state" + 0.003*"need" + 0.003*"time" +
0.003*"come" + 0.003*"include" + 0.003*"change" + 0.003*"see" +
0.003*"get" + 0.003*"world" + 0.002*"work" + 0.002*"many" +
0.002*"country"
Topic : 37 Words : 0.006*"lung cancer" + 0.004*"glide vehicle" +
0.004*"missile defense" + 0.004*"control room" + 0.003*"cancer" +
0.003*"government" + 0.003*"use" + 0.002*"say" + 0.002*"radioactive
particle" + 0.002*"nuclear weapon" + 0.002*"time" + 0.002*"cause lung"
+ 0.002*"new" + 0.002*"future" + 0.002*"next year" + 0.002*"young
people" + 0.002*"present" + 0.002*"brief avangard" + 0.002*"mean
potential" + 0.002*"early bird"
Topic : 38 Words : 0.005*"great barrier" + 0.004*"say" + 0.003*"year"
+ 0.003*"coral reef" + 0.003*"coral die" + 0.002*"people" +
0.002*"change" + 0.002*"life" + 0.002*"live" + 0.002*"really" +
0.002*"occur" + 0.002*"bear" + 0.002*"american citizen" +
0.002*"recently become" + 0.002*"journalist tell" + 0.002*"vote
decriminalise" + 0.002*"tape special" + 0.002*"citizen married" +
0.002*"receive lovely" + 0.002*"feature politician"
Topic : 39 Words : 0.010*"say" + 0.006*"people" + 0.005*"year" +
0.003*"woman" + 0.002*"time" + 0.002*"report" + 0.002*"wallace" +
0.002*"know" + 0.002*"year old" + 0.002*"give" + 0.002*"junior doctor"
+ 0.002*"use" + 0.002*"also" + 0.002*"go" + 0.002*"virus" +
0.002*"country" + 0.002*"make" + 0.002*"need" + 0.002*"study" +
0.002*"call"
Cohérence  :  0.3513085752969611
Perplexité :  -10.321911214476186

     topic_dominant  pourcentage_contrib  \
0                18               0.6456
1                36               0.9920
2                21               0.6917
3                31               0.5922
4                 6               0.9959
..              ...                  ...
931              23               0.9964
932              36               0.5905
933              31               0.9914
934              34               0.9927
935              15               0.4005

                                        topic_keywords  \
0    say, type diabetes, people, work, attack, go, ...
1    say, year, also, people, make, go, even, clima...
2    say, think, police, get, people, see, go, proc...
```

```
3    campaign, russian government, trump campaign, ...
4    mental health, say, lung cancer, ex cnrp, year...
..                                                 ...
931  say, police, human right, time, child, officer...
932  say, year, also, people, make, go, even, clima...
933  campaign, russian government, trump campaign, ...
934  low pay, year, pay, make, work, last year, phi...
935  say, guillotine, bill, pcr test, people, also,...

                                            text_title
0    Hillary Clinton's plane passes over Manhattan ...
1    Rashida Tlaib is busy at work during a nationa...
2    Natural News The oldest magazine in the United...
3    Ministers are undermining trust in foreign aid...
4    Today the Education Policy Institute's Indepen...
..                                                 ...
931  The bombshell claim comes from over 20 hours o...
932  This is a rush transcript from Fox News Sunday...
933  The use of cocaine in Britain has doubled in s...
934  A ndy Murray served up an ace to John Inverdal...
935  Though the whole world relies on RT-PCR to “di...

[936 rows x 4 columns]
```

On rajoute les mots-clés à notre DataFrame de départ pour pouvoir faire la classification

- On a essayé la classification sur les keywords uniquement mais l'accuracy était très basse donc on va essayer de rajouter les mots-clés à notre text, titre, text+titre respectivement

```
# modification du dataframe pour intégrer les mots associés au topic
dominant à chaque document
dftrain['keywords']=df_topic_sents_keywords['topic_keywords']
display(dftrain)
```

```
# selection des données
X=pd.concat([dftrain.iloc[:,1:3], dftrain.iloc[:,4:6]],
axis=1).reset_index(drop=True)

y=dftrain.rating

# Création d'un jeu d'apprentissage et de test
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2,random_state=8)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during
```

```
thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)

           id                                               text  \
0    c9a710dc  Hillary Clinton's plane passes over Manhattan ...
1    a7b20877  Rashida Tlaib is busy at work during a nationa...
2    fb721890  Natural News The oldest magazine in the United...
3    ed8a09ac  Ministers are undermining trust in foreign aid...
4    f454e71d  Today the Education Policy Institute's Indepen...
..        ...                                                ...
931  3886ead8  The bombshell claim comes from over 20 hours o...
932  da3319cc  This is a rush transcript from Fox News Sunday...
933  7b9e930d  The use of cocaine in Britain has doubled in s...
934  48026a71  A ndy Murray served up an ace to John Inverdal...
935  31d33510  Though the whole world relies on RT-PCR to “di...

                                                 title   rating  \
0    Hillary Clinton Boards The Climate Crisis Trai...  mixture
1    Tlaib Files Lawsuit to Ban the American Flag i...    FALSE
2    Still think 5G is harmless? Scientific America...    FALSE
3    Ministers are undermining trust in foreign aid...     TRUE
4    Apocalyptic Sea-Level Rise—Just a Thing of the...     TRUE
..                                                 ...      ...
931  Breaking: Breonna Taylor’s boyfriend says SHE ...    FALSE
932  Pruitt defends decision to withdraw from Paris...  mixture
933  Britain's cocaine use doubles in last seven ye...    other
934  Andy Murray aces John Inverdale after BBC pres...  mixture
935   COVID19 PCR Tests are Scientifically Meaningless    FALSE

                                            text_title  \
0    Hillary Clinton's plane passes over Manhattan ...
1    Rashida Tlaib is busy at work during a nationa...
2    Natural News The oldest magazine in the United...
3    Ministers are undermining trust in foreign aid...
4    Today the Education Policy Institute's Indepen...
..                                                 ...
931  The bombshell claim comes from over 20 hours o...
932  This is a rush transcript from Fox News Sunday...
933  The use of cocaine in Britain has doubled in s...
934  A ndy Murray served up an ace to John Inverdal...
935  Though the whole world relies on RT-PCR to “di...

                                              keywords
0    say, type diabetes, people, work, attack, go, ...
1    say, year, also, people, make, go, even, clima...
2    say, think, police, get, people, see, go, proc...
3    campaign, russian government, trump campaign, ...
4    mental health, say, lung cancer, ex cnrp, year...
..                                                 ...
931  say, police, human right, time, child, officer...
```

```
932  say, year, also, people, make, go, even, clima...
933  campaign, russian government, trump campaign, ...
934  low pay, year, pay, make, work, last year, phi...
935  say, guillotine, bill, pcr test, people, also,...

[936 rows x 6 columns]
```

Vu qu'on va travailler sur text+keywords puis sur titre+keywords après sur la colonne de concaténation de titre et text+keywords, Donc on va d'abord concaténér :

- Texte et keywords
- Titre et keywords
- Titre+texte et keywords

et on va séléctionner ces dernières depuis le X_train et X_test pour apprendre et tester après

```
train_text_keywords = X_train.apply(lambda x : '{}
{}'.format(x['text'],x['keywords']),axis=1)
test_text_keywords = X_test.apply(lambda x : '{}
{}'.format(x['text'],x['keywords']),axis=1)

X_train['text_keywords'] = train_text_keywords
X_train_text_keywords = X_train['text_keywords']
X_train_text_keywords.reset_index(drop = True, inplace = True)

X_test['text_keywords'] = test_text_keywords
X_test_text_keywords = X_test['text_keywords']
X_test_text_keywords.reset_index(drop = True, inplace = True)

train_title_keywords = X_train.apply(lambda x : '{}
{}'.format(x['title'],x['keywords']),axis=1)
test_title_keywords = X_test.apply(lambda x : '{}
{}'.format(x['title'],x['keywords']),axis=1)


X_train['title_keywords'] = train_title_keywords
X_train_title_keywords = X_train['title_keywords']
X_train_title_keywords.reset_index(drop = True, inplace = True)

X_test['title_keywords'] = test_title_keywords
X_test_title_keywords = X_test['title_keywords']
X_test_title_keywords.reset_index(drop = True, inplace = True)

train_text_title_keywords = X_train.apply(lambda x : '{}
{}'.format(x['text_title'],x['keywords']),axis=1)
test_text_title_keywords = X_test.apply(lambda x : '{}
{}'.format(x['text_title'],x['keywords']),axis=1)
```

```
X_train['text_title_keywords'] = train_text_title_keywords
X_train_text_title_keywords = X_train['text_title_keywords']
X_train_text_title_keywords.reset_index(drop = True, inplace = True)

X_test['text_title_keywords'] = test_text_title_keywords
X_test_text_title_keywords = X_test['text_title_keywords']
X_test_text_title_keywords.reset_index(drop = True, inplace = True)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during
thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

Etape 2 : Classification selon la colonne TEXT et KEYWORDS (concaténés) :

Ici, c'est une étape importante, on va tester différents classifieurs, pour chacun des classifieurs, on va appliquer le prétraitement + Vectorisation TfIdf, et on applique une cross_val_score avec un Kfold de 10 fois, par la suite on stocke dans une liste all_results la moyenne des accuracy + l'écart type et on la trie par ordre décroissant de moyenne d'accuracy et d'écart type. on remarque que les 2 meilleurs sont SVM et RF qu'on va séléctionner pour leur appliquer le GridSearch sur les paramètres des prétraitements + leurs hyperparamètres pour pouvoir choisir le meilleur.

```
# Utilisez la méthode ravel() pour transformer y_train en un tableau
unidimensionnel
y_train = np.ravel(y_train)

np.random.seed(42)  # Set the random seed for NumPy

score = 'accuracy'
seed = 7
allresults = []
results = []
names = []


# Liste des modèles à tester
models = [
    ('MultinomialNB', MultinomialNB()),
    ('LogisticRegression', LogisticRegression(random_state=42))
]


#models.append(('LR', LogisticRegression(solver='lbfgs')))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier(random_state=42)))
models.append(('RF', RandomForestClassifier(random_state=42)))
models.append(('SVM', SVC(random_state=42)))

# Création d'un pipeline pour chaque modèle
pipelines = []
for name,model in models:
    pipeline = Pipeline([
        ('normalize', TextNormalizer()),
        ('tfidf', TfidfVectorizer()),
        (name,model)
    ])
    pipelines.append((name,pipeline))
all_results=[]
scores=[]
for p in pipelines:
    print(p[1])
    # cross validation en 10 fois
    kfold = KFold(n_splits=10,random_state=seed,shuffle=True)

    start_time = time.time()
    # application de la classification
    cv_results = cross_val_score(p[1],X_train_text_keywords,y_train,
cv=kfold, scoring=score)
    scores.append(cv_results)

    all_results.append((p[0],cv_results.mean(),cv_results.std()))
    end_time = time.time()


all_results = sorted(all_results, key=lambda x: (-x[1], -x[2]))
print("all resultats", all_results)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during
thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)

Pipeline(steps=[('normalize', TextNormalizer()), ('tfidf',
TfidfVectorizer()),
                ('MultinomialNB', MultinomialNB())])
Pipeline(steps=[('normalize', TextNormalizer()), ('tfidf',
TfidfVectorizer()),
                ('LogisticRegression',
LogisticRegression(random_state=42))])
Pipeline(steps=[('normalize', TextNormalizer()), ('tfidf',
TfidfVectorizer()),
                ('KNN', KNeighborsClassifier())])
```

```
Exception ignored on calling ctypes callback function: <function
ThreadpoolController._find_libraries_with_dl_iterate_phdr.<locals>.mat
ch_library_callback at 0x7f8174d16440>
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/threadpoolctl.py",
line 584, in match_library_callback
    self._make_controller_from_path(filepath)
  File "/usr/local/lib/python3.10/dist-packages/threadpoolctl.py",
line 725, in _make_controller_from_path
    lib_controller = lib_controller_class(
  File "/usr/local/lib/python3.10/dist-packages/threadpoolctl.py",
line 842, in __init__
    super().__init__(**kwargs)
  File "/usr/local/lib/python3.10/dist-packages/threadpoolctl.py",
line 810, in __init__
    self._dynlib = ctypes.CDLL(filepath, mode=_RTLD_NOLOAD)
  File "/usr/lib/python3.10/ctypes/__init__.py", line 374, in __init__
    self._handle = _dlopen(self._name, mode)
OSError:
/usr/local/lib/python3.10/dist-packages/numpy.libs/libopenblas64_p-r0-
2f7c42d4.3.18.so: cannot open shared object file: No such file or
directory

Pipeline(steps=[('normalize', TextNormalizer()), ('tfidf',
TfidfVectorizer()),
                ('CART', DecisionTreeClassifier(random_state=42))])
Pipeline(steps=[('normalize', TextNormalizer()), ('tfidf',
TfidfVectorizer()),
                ('RF', RandomForestClassifier(random_state=42))])
Pipeline(steps=[('normalize', TextNormalizer()), ('tfidf',
TfidfVectorizer()),
                ('SVM', SVC(random_state=42))])
all resultats [('SVM', 0.6965045045045045, 0.057608943384907636),
('RF', 0.687081081081081, 0.044142603392858075),
('LogisticRegression', 0.675099099099099, 0.05613998064799174),
('CART', 0.6550810810810811, 0.04452193917534138), ('MultinomialNB',
0.5976576576576578, 0.07196433388726892), ('KNN', 0.4319099099099099,
0.061277817052785476)]
```

On a un pipeline pour chaque prétraitement différent, on essaye pas mal (miniscule, lemmatisation, miniscule + lemmatisation..) et on stocke le fit_transorm de nos X_train, X_test sur les pipelines dans des listes qui vont contenir tous les fit_transform des pipelines pour chaque classifieur, par la suite on parcourt ces listes là, on itère dessus, et chaque élement de la liste (train) va passer par le GridSearch et puis on predict sur son corresapondant dans liste (test).

```
np.random.seed(42)  # Set the random seed for NumPy
```

```
# pipeline de l'utilisation de TfidfVectorizer avec differents pre-
traitements
TFIDF_brut = Pipeline ([('cleaner', TextNormalizer()),
                    ('tfidf_vectorizer',
TfidfVectorizer(lowercase=False))])

TFIDF_lowcase = Pipeline([('cleaner',
TextNormalizer(removestopwords=False,lowercase=True,

getstemmer=False,removedigit=False)),
                    ('tfidf_vectorizer',
TfidfVectorizer(lowercase=False))])
TFIDF_lowStop = Pipeline([('cleaner',
TextNormalizer(removestopwords=True,lowercase=True,

getstemmer=False,removedigit=False)),
                    ('tfidf_vectorizer',
TfidfVectorizer(lowercase=False))])

TFIDF_lowStopstem = Pipeline([('cleaner',
TextNormalizer(removestopwords=True,lowercase=True,

getstemmer=True,removedigit=False)),
                    ('tfidf_vectorizer',
TfidfVectorizer(lowercase=False))])


# Liste de tous les modeles à tester
all_models = [
    ("TFIDF_lowcase", TFIDF_lowcase),
    ("TFIDF_lowStop", TFIDF_lowStop),
    ("TFIDF_lowStopstem",TFIDF_lowStopstem),
    ("TFIDF_brut", TFIDF_brut)
]

X_train_text_keywords_SVC = []
X_test_text_keywords_SVC = []

X_train_text_keywords_RandomForestClassifier = []
X_test_text_keywords_RandomForestClassifier = []


for name, pipeline in all_models :

X_train_text_keywords_SVC.append(pipeline.fit_transform(X_train_text_k
eywords).toarray())

X_test_text_keywords_SVC.append(pipeline.transform(X_test_text_keyword
s).toarray())
```

```
X_train_text_keywords_RandomForestClassifier.append(pipeline.fit_trans
form(X_train_text_keywords).toarray())

X_test_text_keywords_RandomForestClassifier.append(pipeline.transform(
X_test_text_keywords).toarray())
```

```
models = {
    'SVC': SVC(random_state=42),
    'RandomForestClassifier': RandomForestClassifier(random_state=42)
}


params = {'SVC': [{'C': [0.01, 0.1, 1,2]},
             {'gamma': [0.001, 0.01, 0.1,1]},
             {'kernel': ['linear', 'rbf']}],
    'RandomForestClassifier': [{'n_estimators': [10, 50, 100, 200,
300]},
                              {'max_features': ['auto', 'sqrt',
'log2']}],
}

for model_name, model in models.items():
    score='accuracy'
    X_train_text_keywords = eval('X_train_text_keywords_' +
model_name)
    X_test_text_keywords = eval('X_test_text_keywords_' + model_name)
    for i in range (len(X_train_text_keywords)):
      grid_search = GridSearchCV(model, params[model_name], n_jobs=-1,
verbose=1,scoring=score)
      print("grid search fait")
      grid_search.fit(X_train_text_keywords[i],y_train)
      print ('meilleur score %0.3f'%(grid_search.best_score_),'\n')
      print ('meilleur estimateur',grid_search.best_estimator_,'\n')
      y_pred = grid_search.predict(X_test_text_keywords[i])
      MyshowAllScores(y_test,y_pred)

      print("Ensemble des meilleurs paramètres :")
      best_parameters = grid_search.best_estimator_.get_params()
      for param_dict in params[model_name]:
        for param_name, param_value in param_dict.items():
            print("\t%s: %r" % (param_name,
best_parameters[param_name]))
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to
```

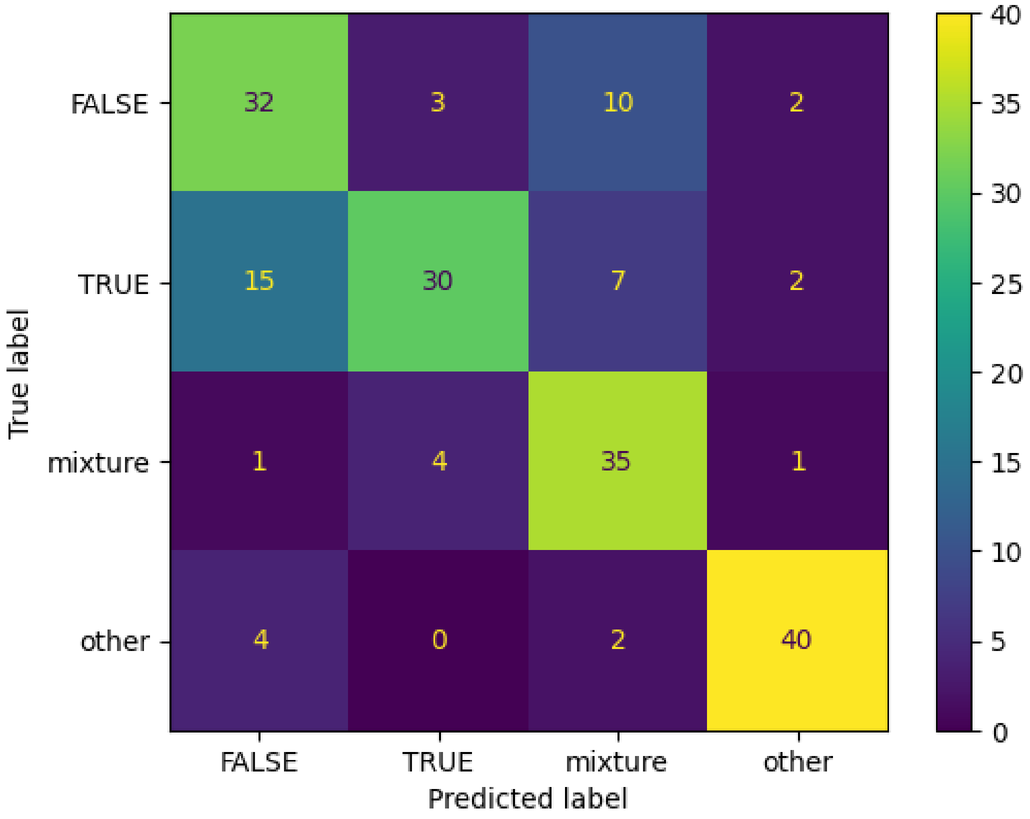
```
`transformed_cell` argument and any exception that happen during
thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

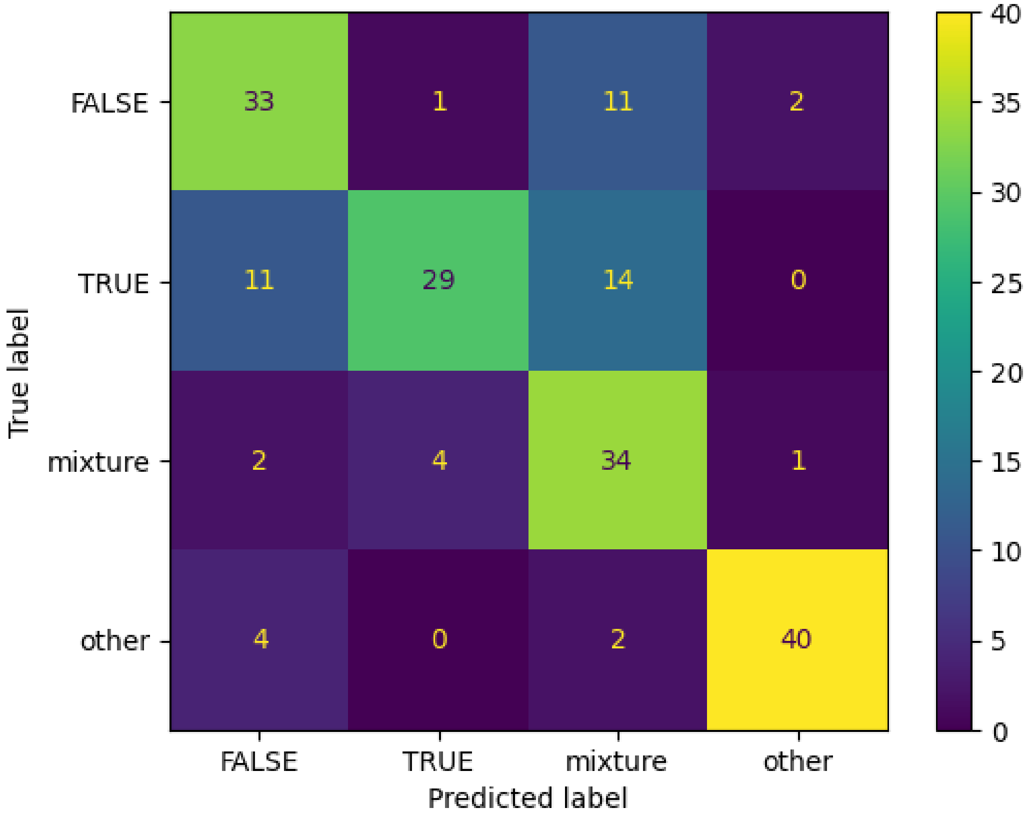
```
grid search fait
Fitting 5 folds for each of 10 candidates, totalling 50 fits
meilleur score 0.695

meilleur estimateur SVC(C=2, random_state=42)

Accuracy : 0.729
Classification Report
              precision    recall  f1-score   support

       FALSE    0.61538   0.68085   0.64646        47
        TRUE    0.81081   0.55556   0.65934        54
     mixture    0.64815   0.85366   0.73684        41
       other    0.88889   0.86957   0.87912        46

    accuracy                        0.72872       188
   macro avg    0.74081   0.73991   0.73044       188
weighted avg    0.74558   0.72872   0.72680       188

Ensemble des meilleurs paramètres :
     C: 2
     gamma: 'scale'
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 10 candidates, totalling 50 fits
meilleur score 0.701

meilleur estimateur SVC(C=1, random_state=42)

Accuracy : 0.723
Classification Report
              precision    recall  f1-score   support

       FALSE    0.66000   0.70213   0.68041        47
        TRUE    0.85294   0.53704   0.65909        54
     mixture    0.55738   0.82927   0.66667        41
       other    0.93023   0.86957   0.89888        46

    accuracy                        0.72340       188
   macro avg    0.75014   0.73450   0.72626       188
weighted avg    0.75916   0.72340   0.72474       188

Ensemble des meilleurs paramètres :
     C: 1
     gamma: 'scale'
```

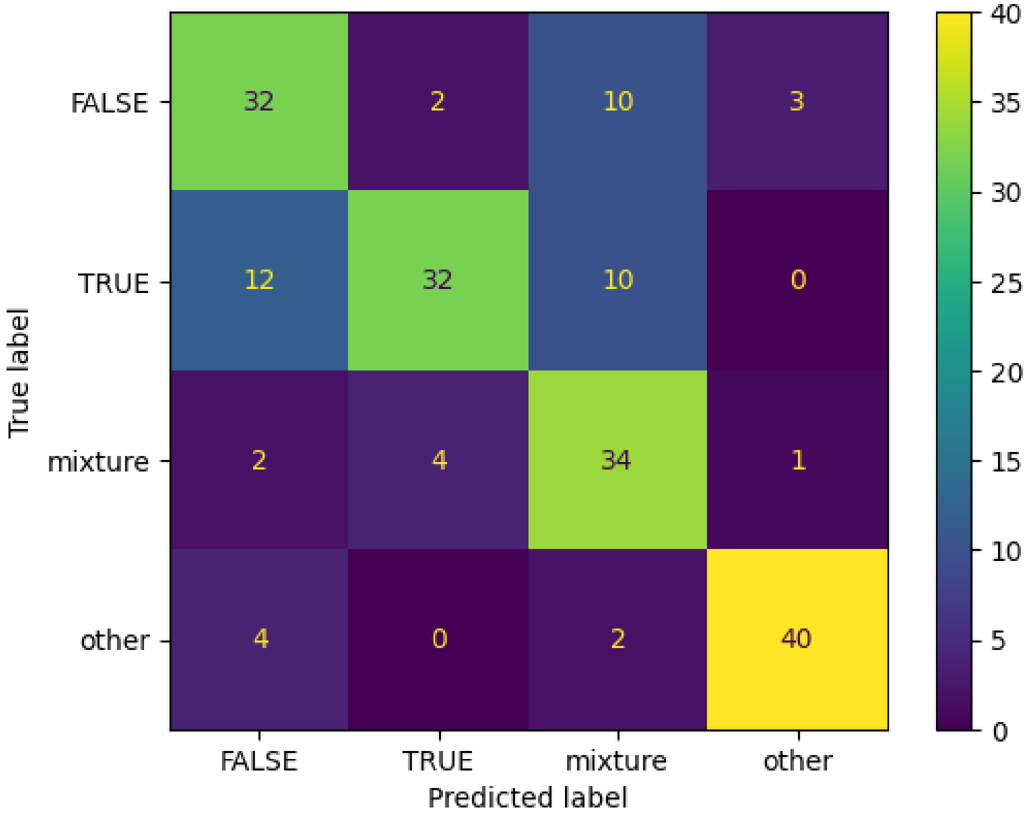
```
      kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 10 candidates, totalling 50 fits
meilleur score 0.691

meilleur estimateur SVC(C=2, random_state=42)

Accuracy : 0.734
Classification Report
              precision    recall  f1-score   support

       FALSE    0.64000   0.68085   0.65979        47
        TRUE    0.84211   0.59259   0.69565        54
     mixture    0.60714   0.82927   0.70103        41
       other    0.90909   0.86957   0.88889        46

    accuracy                        0.73404       188
   macro avg    0.74958   0.74307   0.73634       188
weighted avg    0.75673   0.73404   0.73514       188

Ensemble des meilleurs paramètres :
      C: 2
      gamma: 'scale'
      kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 10 candidates, totalling 50 fits
meilleur score 0.690

meilleur estimateur SVC(C=2, random_state=42)

Accuracy : 0.718
Classification Report
              precision    recall  f1-score   support

       FALSE    0.59184   0.61702   0.60417        47
        TRUE    0.80000   0.59259   0.68085        54
     mixture    0.61818   0.82927   0.70833        41
       other    0.90909   0.86957   0.88889        46

    accuracy                        0.71809       188
   macro avg    0.72978   0.72711   0.72056       188
weighted avg    0.73500   0.71809   0.71858       188

Ensemble des meilleurs paramètres :
      C: 2
      gamma: 'scale'
      kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
```

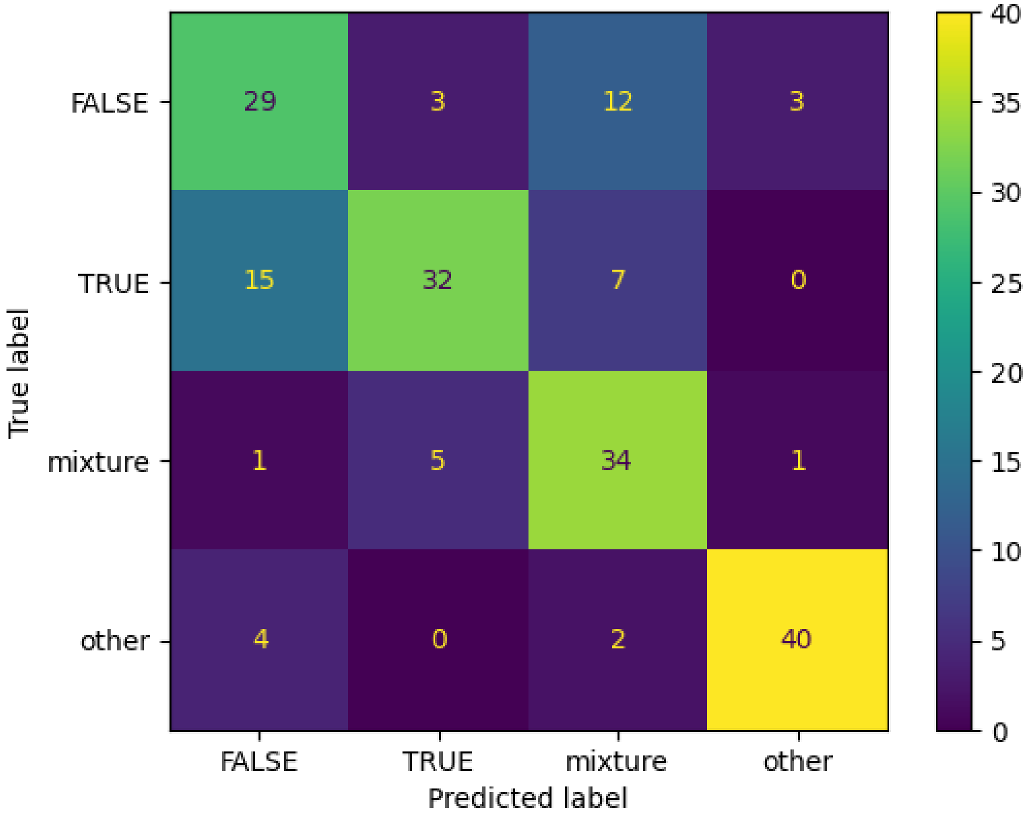
```
meilleur score 0.678

meilleur estimateur RandomForestClassifier(n_estimators=300,
random_state=42)

Accuracy : 0.734
Classification Report
              precision    recall  f1-score   support

       FALSE    0.60714   0.72340   0.66019        47
        TRUE    0.71429   0.64815   0.67961        54
     mixture    0.78378   0.70732   0.74359        41
       other    0.86957   0.86957   0.86957        46

    accuracy                        0.73404       188
   macro avg    0.74369   0.73711   0.73824       188
weighted avg    0.74065   0.73404   0.73519       188

Ensemble des meilleurs paramètres :
     n_estimators: 300
     max_features: 'sqrt'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.687

meilleur estimateur RandomForestClassifier(random_state=42)

Accuracy : 0.707
Classification Report
              precision    recall  f1-score   support

       FALSE    0.53448   0.65957   0.59048        47
        TRUE    0.80488   0.61111   0.69474        54
     mixture    0.65854   0.65854   0.65854        41
       other    0.87500   0.91304   0.89362        46

    accuracy                        0.70745       188
   macro avg    0.71822   0.71057   0.70934       188
weighted avg    0.72252   0.70745   0.70944       188

Ensemble des meilleurs paramètres :
     n_estimators: 100
     max_features: 'sqrt'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.670

meilleur estimateur RandomForestClassifier(n_estimators=300,
random_state=42)
```

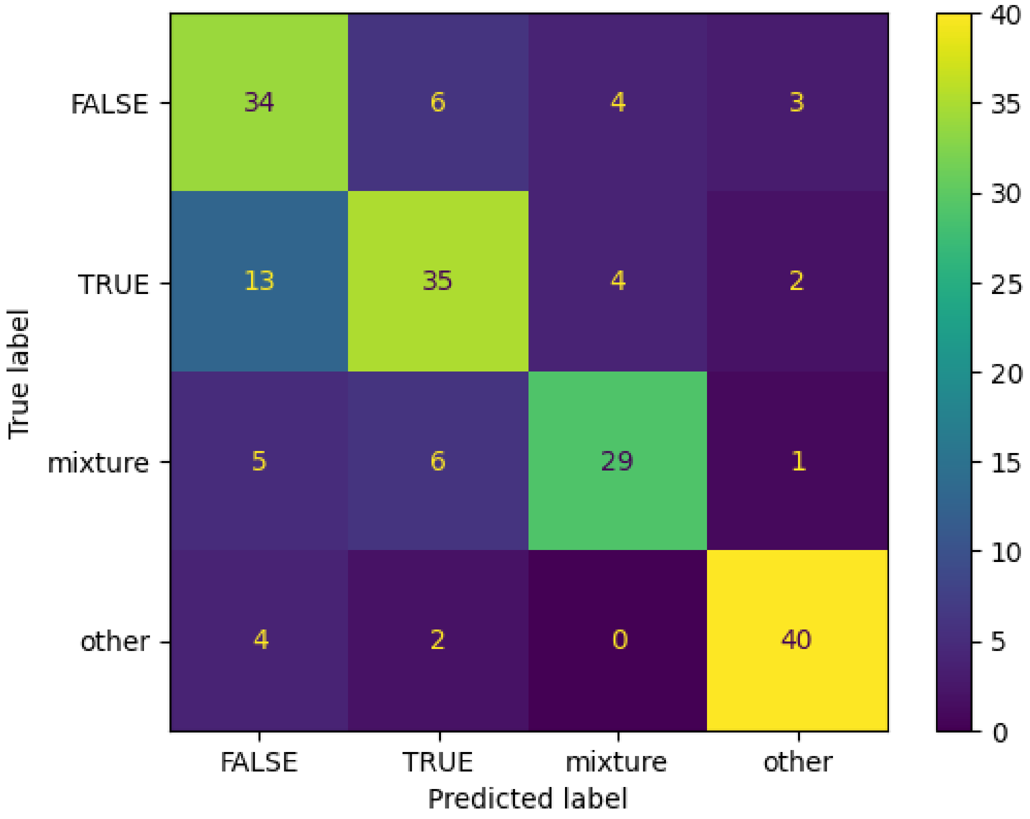
```
Accuracy : 0.702
Classification Report
              precision    recall  f1-score   support

       FALSE    0.51852   0.59574   0.55446        47
        TRUE    0.77778   0.64815   0.70707        54
     mixture    0.65909   0.70732   0.68235        41
       other    0.88889   0.86957   0.87912        46

    accuracy                        0.70213       188
   macro avg    0.71107   0.70519   0.70575       188
weighted avg    0.71427   0.70213   0.70562       188

Ensemble des meilleurs paramètres :
     n_estimators: 300
     max_features: 'sqrt'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.690

meilleur estimateur RandomForestClassifier(n_estimators=300,
random_state=42)

Accuracy : 0.702
Classification Report
              precision    recall  f1-score   support

       FALSE    0.55000   0.70213   0.61682        47
        TRUE    0.73333   0.61111   0.66667        54
     mixture    0.66667   0.63415   0.65000        41
       other    0.90909   0.86957   0.88889        46

    accuracy                        0.70213       188
   macro avg    0.71477   0.70424   0.70559       188
weighted avg    0.71597   0.70213   0.70494       188

Ensemble des meilleurs paramètres :
     n_estimators: 300
     max_features: 'sqrt'
```

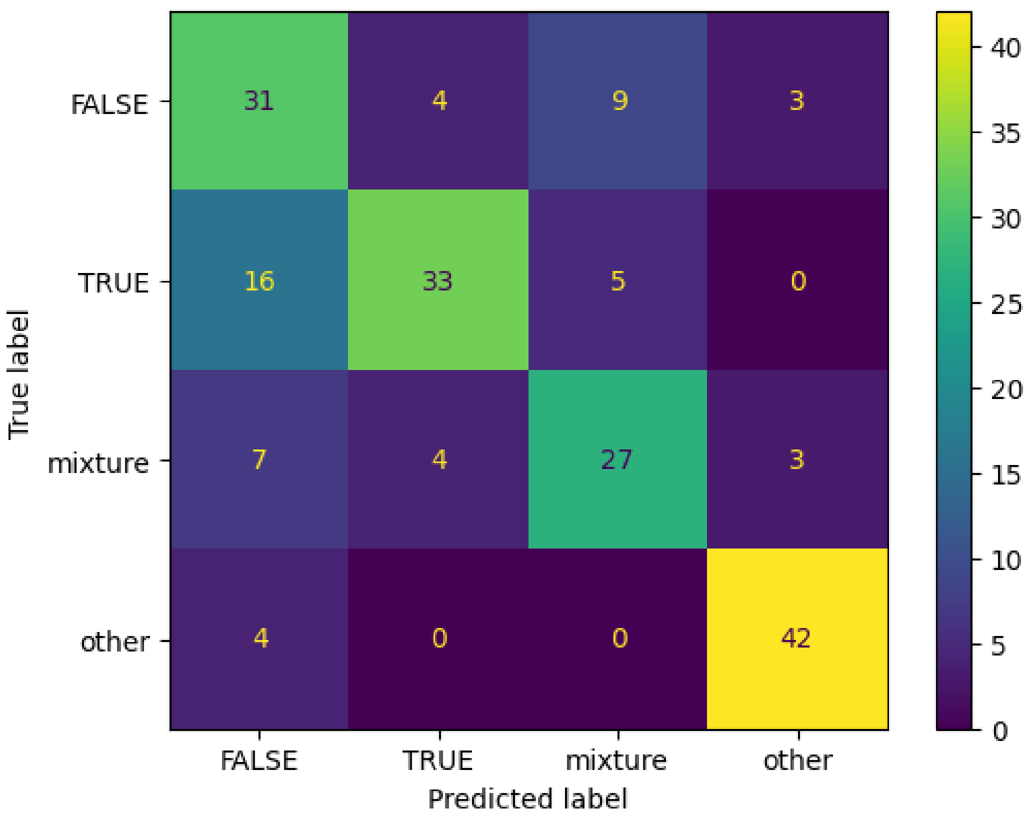

FALSE
TRUE
mixture
other
True label
32
3
10
2
15
30
7
2
1
4
35
1
4
0
2
40
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0

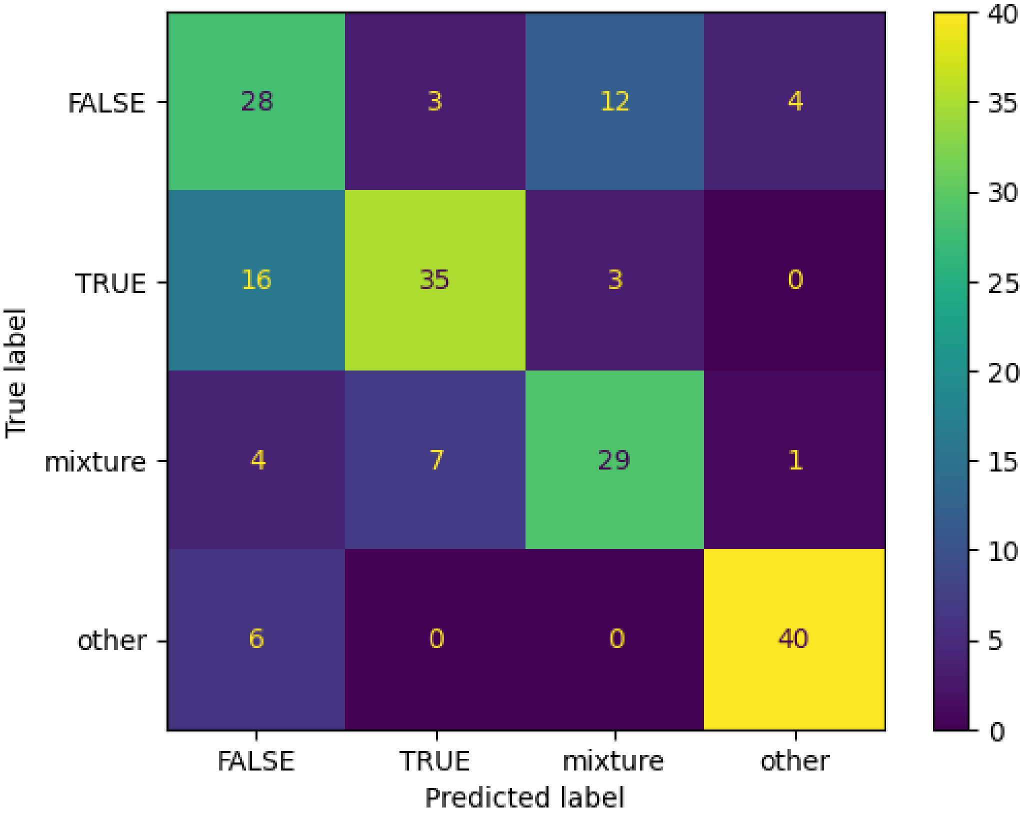

FALSE
TRUE
mixture
other
True label
33
1
11
2
11
29
14
0
2
4
34
1
4
0
2
40
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0

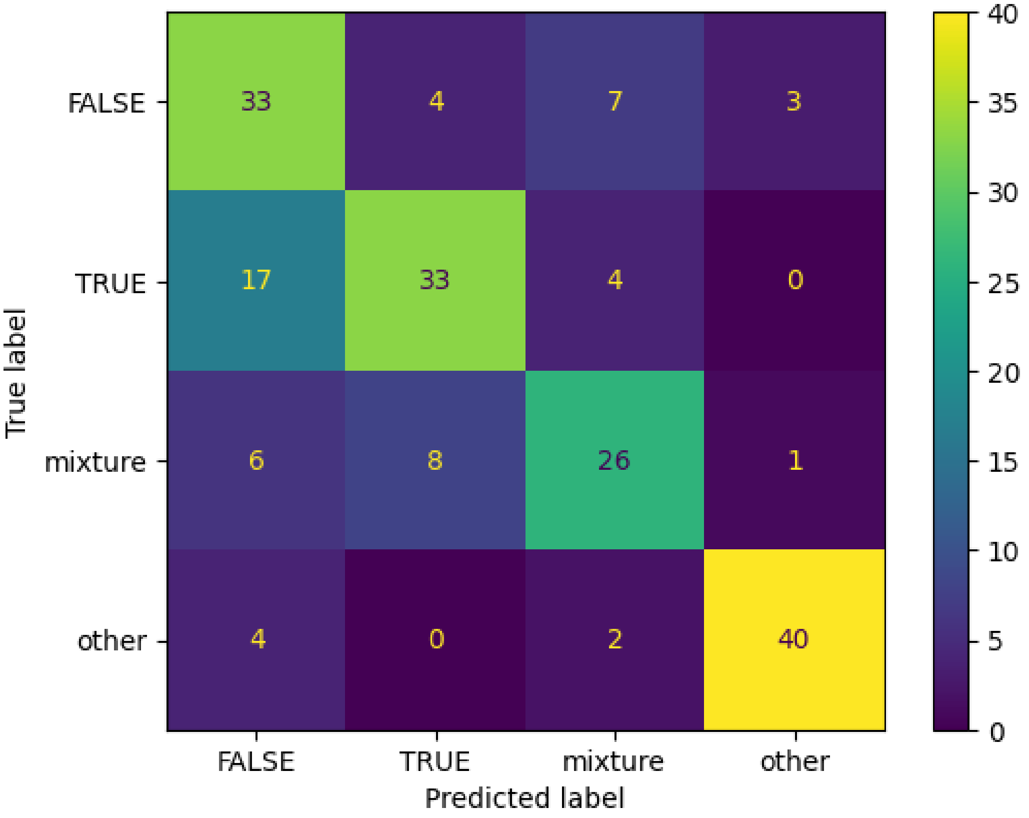

True label
FALSE
TRUE
mixture
other
32
2
10
3
12
32
10
0
2
4
34
1
4
0
2
40
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0


True label
FALSE
TRUE
mixture
other
29
3
12
3
15
32
7
0
1
5
34
1
4
0
2
40
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0


FALSE
TRUE
mixture
other
True label
34
6
4
3
13
35
4
2
5
6
29
1
4
2
0
40
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0


FALSE
TRUE
mixture
other
True label
31
4
9
3
16
33
5
0
7
4
27
3
4
0
0
42
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0


True label
Predicted label
FALSE
TRUE
mixture
other
FALSE 28 3 12 4
TRUE 16 35 3 0
mixture 4 7 29 1
other 6 0 0 40
FALSE
TRUE
mixture
other
0
5
10
15
20
25
30
35
40


True label
Predicted label
FALSE
TRUE
mixture
other
FALSE 33 4 7 3
TRUE 17 33 4 0
mixture 6 8 26 1
other 4 0 2 40
FALSE
TRUE
mixture
other
0
5
10
15
20
25
30
35
40

Etape 3 : Classification selon la colonne TITRE et KEYWORDS (concaténés):

Ici, c'est une étape importante, on va tester différents classifieurs, pour chacun des classifieurs, on va appliquer le prétraitement + Vectorisation TfIdf, et on applique une cross_val_score avec un Kfold de 10 fois, par la suite on stocke dans une liste all_results la moyenne des accuracy + l'écart type et on la trie par ordre décroissant de moyenne d'accuracy et d'écart type. on remarque que les 2 meilleurs sont SVM et RF qu'on va séléctionner pour leur appliquer le GridSearch sur les paramètres des prétraitements + leurs hyperparamètres pour pouvoir choisir le meilleur.

```
# Utilisez la méthode ravel() pour transformer y_train en un tableau unidimensionnel
y_train = np.ravel(y_train)

np.random.seed(42)  # Set the random seed for NumPy

score = 'accuracy'
seed = 7
allresults = []
results = []
names = []


# Liste des modèles à tester
models = [
    ('MultinomialNB', MultinomialNB()),
    ('LogisticRegression', LogisticRegression(random_state=42))
]


#models.append(('LR', LogisticRegression(solver='lbfgs')))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier(random_state=42)))
models.append(('RF', RandomForestClassifier(random_state=42)))
models.append(('SVM', SVC(random_state=42)))

# Création d'un pipeline pour chaque modèle
pipelines = []
for name,model in models:
    pipeline = Pipeline([
        ('normalize', TextNormalizer()),
        ('tfidf', TfidfVectorizer()),
        (name,model)
    ])
    pipelines.append((name,pipeline))
    #pipeline.fit(X_train_text,y_train)
all_results=[]
scores=[]
for p in pipelines:
    print(p[1])
    # cross validation en 10 fois
    kfold = KFold(n_splits=10,random_state=seed,shuffle=True)

#    print ("Evaluation de ",p)
    start_time = time.time()
    # application de la classification
    cv_results = cross_val_score(p[1],X_train_title_keywords,y_train,
cv=kfold, scoring=score)
    #print("Pour le classifieur",p[0],"on a un score
de",cv_results.mean(),"et un écart type de",cv_results.std())
    scores.append(cv_results)

    all_results.append((p[0],cv_results.mean(),cv_results.std()))
    end_time = time.time()


all_results = sorted(all_results, key=lambda x: (-x[1], -x[2]))
print("all resultats", all_results)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during
thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

```
Pipeline(steps=[('normalize', TextNormalizer()), ('tfidf',
TfidfVectorizer()),
                ('MultinomialNB', MultinomialNB())])
Pipeline(steps=[('normalize', TextNormalizer()), ('tfidf',
TfidfVectorizer()),
                ('LogisticRegression',
LogisticRegression(random_state=42))])
Pipeline(steps=[('normalize', TextNormalizer()), ('tfidf',
TfidfVectorizer()),
                ('KNN', KNeighborsClassifier())])
Pipeline(steps=[('normalize', TextNormalizer()), ('tfidf',
TfidfVectorizer()),
                ('CART', DecisionTreeClassifier(random_state=42))])
Pipeline(steps=[('normalize', TextNormalizer()), ('tfidf',
TfidfVectorizer()),
                ('RF', RandomForestClassifier(random_state=42))])
Pipeline(steps=[('normalize', TextNormalizer()), ('tfidf',
TfidfVectorizer()),
                ('SVM', SVC(random_state=42))])
all resultats [('SVM', 0.6256936936936937, 0.0378895571029324),
('CART', 0.6055855855855856, 0.048888896267274086), ('RF',
0.5975855855855856, 0.041202066197067774), ('LogisticRegression',
0.5668828828828828, 0.040074340463373624), ('MultinomialNB',
0.5026846846846846, 0.03345266684072579), ('KNN', 0.3528828828828829, 
0.04894001834558235)]
```

On a un pipeline pour chaque prétraitement différent, on essaye pas mal (miniscule, lemmatisation, miniscule + lemmatisation..) et on stocke le fit_transorm de nos X_train, X_test sur les pipelines dans des listes qui vont contenir tous les fit_transform des pipelines pour chaque classifieur, par la suite on parcourt ces listes là, on itère dessus, et chaque élement de la liste (train) va passer par le GridSearch et puis on predict sur son corresapondant dans liste (test).

```
np.random.seed(42)  # Set the random seed for NumPy

# le plus simple est de faire un test sur differents pipelines.
# pipeline de l'utilisation de TfidfVectorizer avec differents pre-
traitements
TFIDF_brut = Pipeline ([('cleaner', TextNormalizer()),
                    ('tfidf_vectorizer', 
TfidfVectorizer(lowercase=False))])

TFIDF_lowcase = Pipeline([('cleaner', 
TextNormalizer(removestopwords=False,lowercase=True,

getstemmer=False,removedigit=False)),
                    ('tfidf_vectorizer', 
TfidfVectorizer(lowercase=False))])
TFIDF_lowStop = Pipeline([('cleaner', 
TextNormalizer(removestopwords=True,lowercase=True,

getstemmer=False,removedigit=False)),
                    ('tfidf_vectorizer', 
TfidfVectorizer(lowercase=False))])

TFIDF_lowStopstem = Pipeline([('cleaner', 
TextNormalizer(removestopwords=True,lowercase=True,

getstemmer=True,removedigit=False)),
                    ('tfidf_vectorizer', 
TfidfVectorizer(lowercase=False))])


# Liste de tous les modeles à tester
all_models = [
    ("TFIDF_lowcase", TFIDF_lowcase),
    ("TFIDF_lowStop", TFIDF_lowStop),
    ("TFIDF_lowStopstem",TFIDF_lowStopstem),
    ("TFIDF_brut", TFIDF_brut)
]

X_train_title_keywords_SVC = []
```

```
X_test_title_keywords_SVC = []

X_train_title_keywords_RandomForestClassifier = []
X_test_title_keywords_RandomForestClassifier = []


for name, pipeline in all_models :

X_train_title_keywords_SVC.append(pipeline.fit_transform(X_train_title
_keywords).toarray())

X_test_title_keywords_SVC.append(pipeline.transform(X_test_title_keywo
rds).toarray())

X_train_title_keywords_RandomForestClassifier.append(pipeline.fit_tran
sform(X_train_title_keywords).toarray())

X_test_title_keywords_RandomForestClassifier.append(pipeline.transform
(X_test_title_keywords).toarray())




models = {
    'SVC': SVC(random_state=42),
    'RandomForestClassifier': RandomForestClassifier(random_state=42)
}


params = {'SVC': [{'C': [0.001, 0.01, 0.1, 1,2]},
             {'gamma': [0.001, 0.01, 0.1,0.2]},
             {'kernel': ['linear', 'rbf']}],
    'RandomForestClassifier': [{'n_estimators': [10, 50, 100, 200,
300]},
                              {'max_features': ['auto', 'sqrt',
'log2']}],
}

for model_name, model in models.items():
    score='accuracy'
    X_train_title_keywords = eval('X_train_title_keywords_' +
model_name)
    X_test_title_keywords = eval('X_test_title_keywords_' +
model_name)
    for i in range (len(X_train_title_keywords)):
      grid_search = GridSearchCV(model, params[model_name], n_jobs=-1,
verbose=1,scoring=score)
      print("grid search fait")
      grid_search.fit(X_train_title_keywords[i],y_train)
      print ('meilleur score %0.3f'%(grid_search.best_score_),'\n')
      print ('meilleur estimateur',grid_search.best_estimator_,'\n')
      y_pred = grid_search.predict(X_test_title_keywords[i])
      MyshowAllScores(y_test,y_pred)

      print("Ensemble des meilleurs paramètres :")
      best_parameters = grid_search.best_estimator_.get_params()
      for param_dict in params[model_name]:
        for param_name, param_value in param_dict.items():
            print("\t%s: %r" % (param_name,
best_parameters[param_name]))
```

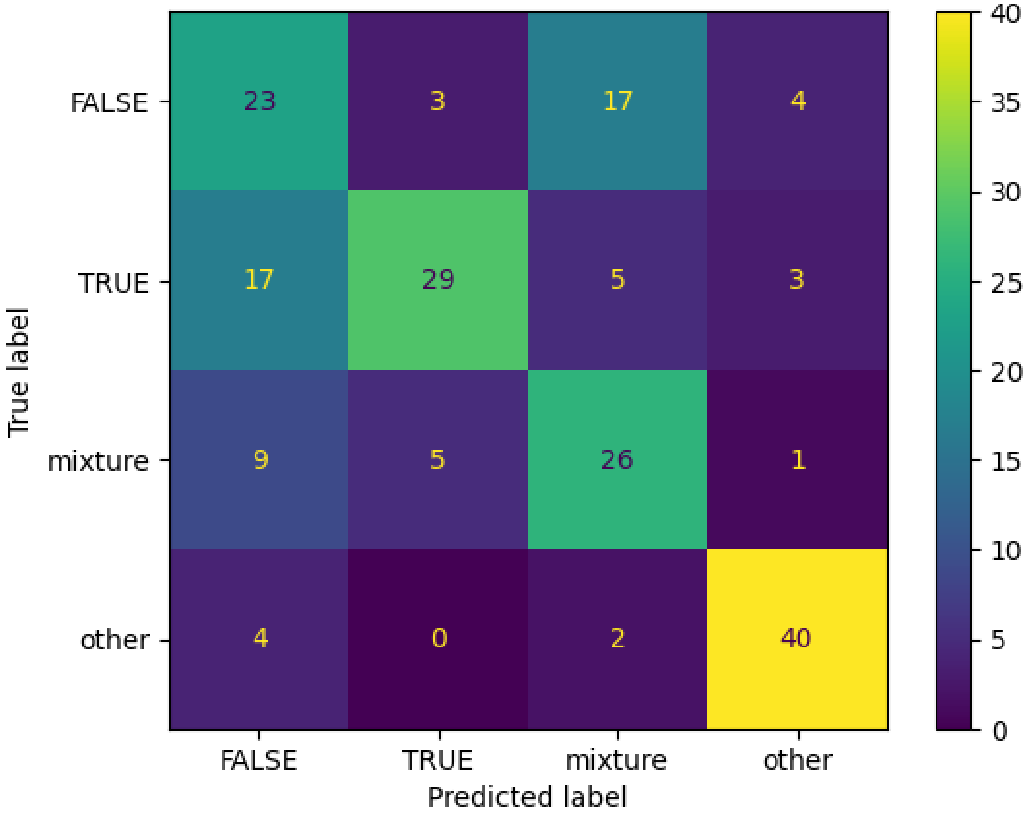
```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during
thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)

grid search fait
Fitting 5 folds for each of 11 candidates, totalling 55 fits
meilleur score 0.627

meilleur estimateur SVC(C=2, random_state=42)

Accuracy : 0.628
Classification Report
              precision    recall  f1-score   support

       FALSE    0.43396   0.48936   0.46000        47
        TRUE    0.78378   0.53704   0.63736        54
     mixture    0.52000   0.63415   0.57143        41
       other    0.83333   0.86957   0.85106        46

    accuracy                        0.62766       188
   macro avg    0.64277   0.63253   0.62996       188
weighted avg    0.65092   0.62766   0.63093       188

Ensemble des meilleurs paramètres :
     C: 2
     gamma: 'scale'
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 11 candidates, totalling 55 fits
meilleur score 0.631

meilleur estimateur SVC(C=2, random_state=42)

Accuracy : 0.606
Classification Report
```

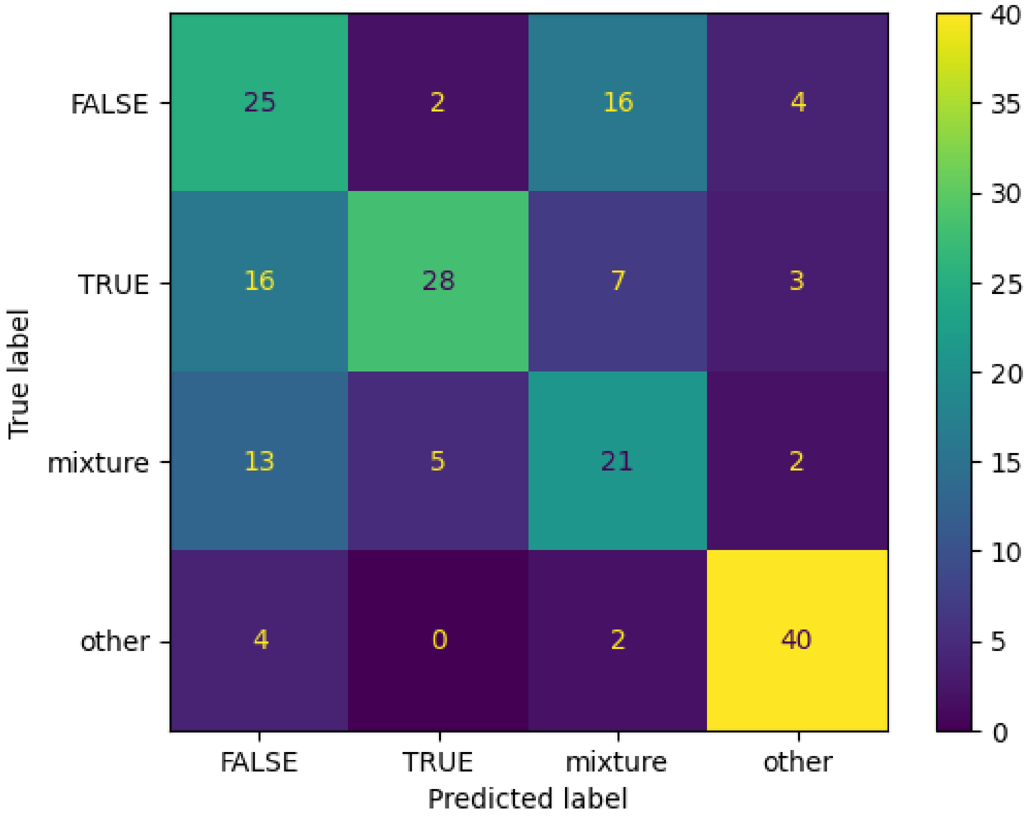
```
              precision    recall  f1-score   support

       FALSE    0.43103   0.53191   0.47619        47
        TRUE    0.80000   0.51852   0.62921        54
     mixture    0.45652   0.51220   0.48276        41
       other    0.81633   0.86957   0.84211        46

    accuracy                        0.60638       188
   macro avg    0.62597   0.60805   0.60757       188
weighted avg    0.63685   0.60638   0.61111       188

Ensemble des meilleurs paramètres :
     C: 2
     gamma: 'scale'
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 11 candidates, totalling 55 fits
meilleur score 0.619

meilleur estimateur SVC(C=2, random_state=42)

Accuracy : 0.622
Classification Report
              precision    recall  f1-score   support

       FALSE    0.43636   0.51064   0.47059        47
        TRUE    0.77778   0.51852   0.62222        54
     mixture    0.52083   0.60976   0.56180        41
       other    0.81633   0.86957   0.84211        46

    accuracy                        0.62234       188
   macro avg    0.63783   0.62712   0.62418       188
weighted avg    0.64582   0.62234   0.62494       188

Ensemble des meilleurs paramètres :
     C: 2
     gamma: 'scale'
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 11 candidates, totalling 55 fits
meilleur score 0.636

meilleur estimateur SVC(C=1, random_state=42)

Accuracy : 0.622
Classification Report
              precision    recall  f1-score   support

       FALSE    0.50000   0.44681   0.47191        47
```

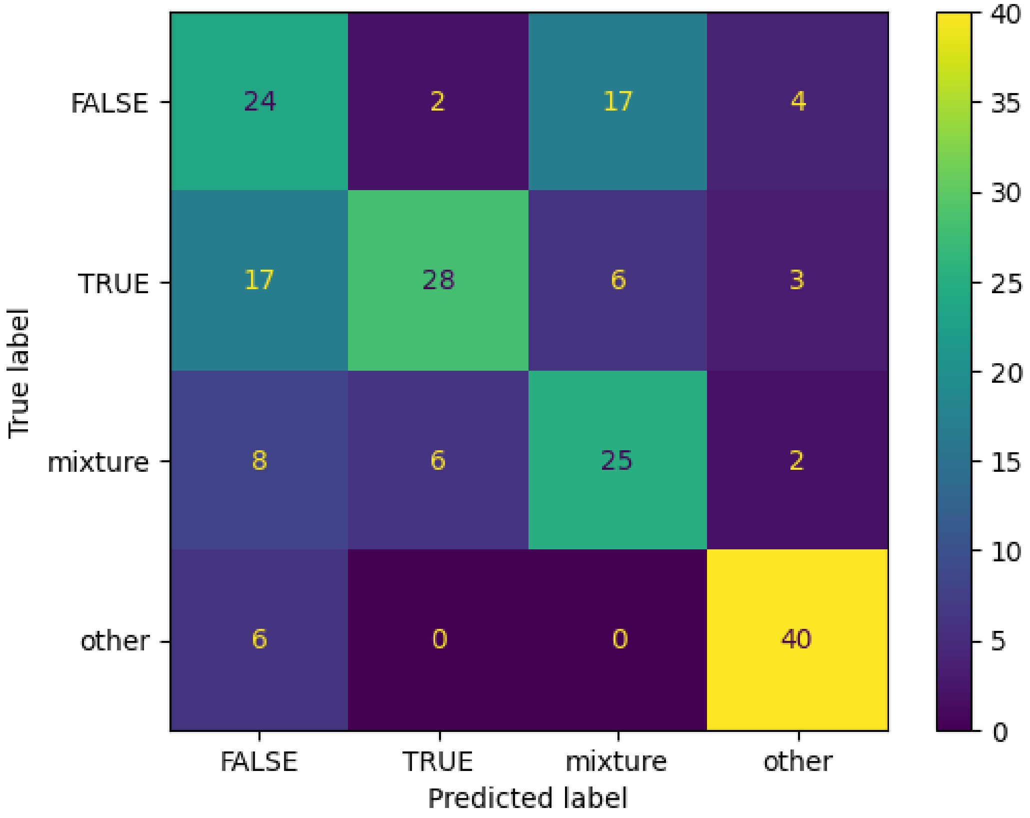
```
        TRUE    0.92593   0.46296   0.61728        54
     mixture    0.45714   0.78049   0.57658        41
       other    0.79592   0.84783   0.82105        46

    accuracy                        0.62234       188
   macro avg    0.66975   0.63452   0.62171       188
weighted avg    0.68540   0.62234   0.62192       188

Ensemble des meilleurs paramètres :
     C: 1
     gamma: 'scale'
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.631

meilleur estimateur RandomForestClassifier(n_estimators=200,
random_state=42)

Accuracy : 0.564
Classification Report
              precision    recall  f1-score   support

       FALSE    0.36735   0.38298   0.37500        47
        TRUE    0.70000   0.51852   0.59574        54
     mixture    0.42553   0.48780   0.45455        41
       other    0.76923   0.86957   0.81633        46

    accuracy                        0.56383       188
   macro avg    0.56553   0.56472   0.56040       188
weighted avg    0.57392   0.56383   0.56374       188

Ensemble des meilleurs paramètres :
     n_estimators: 200
     max_features: 'sqrt'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.632

meilleur estimateur RandomForestClassifier(n_estimators=300,
random_state=42)

Accuracy : 0.590
Classification Report
              precision    recall  f1-score   support

       FALSE    0.45833   0.46809   0.46316        47
        TRUE    0.62791   0.50000   0.55670        54
     mixture    0.44898   0.53659   0.48889        41
```

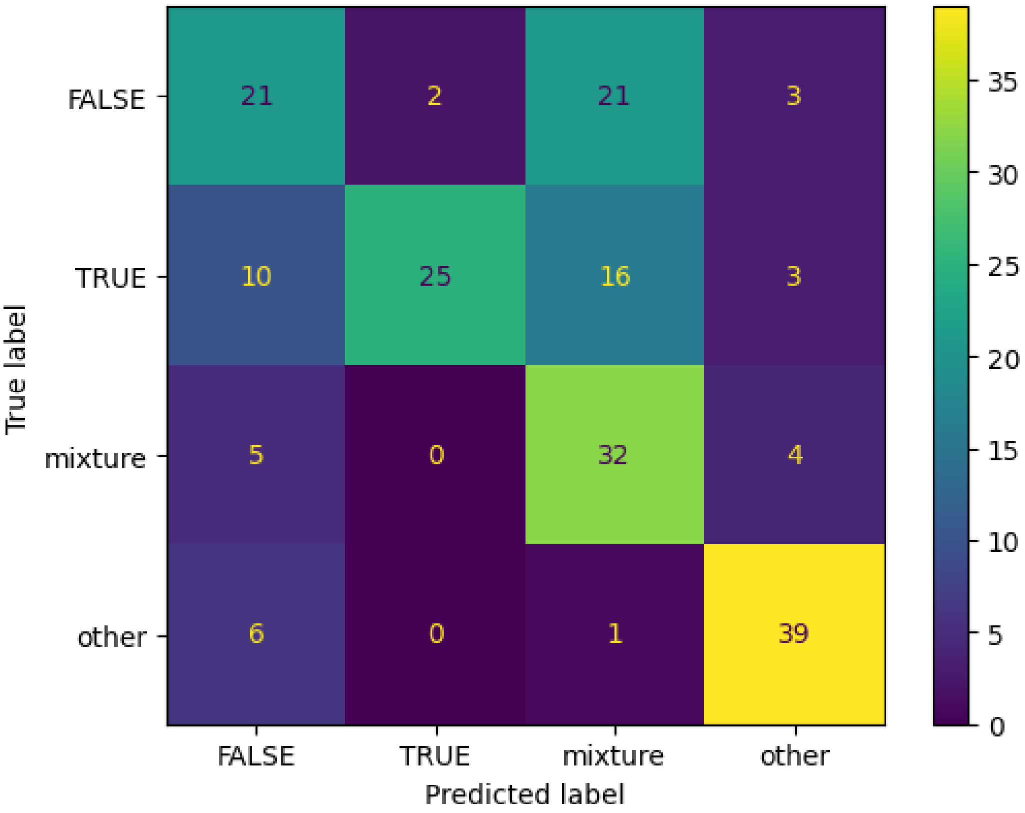
```
       other    0.83333   0.86957   0.85106        46

    accuracy                        0.59043       188
   macro avg    0.59214   0.59356   0.58995       188
weighted avg    0.59676   0.59043   0.59055       188

Ensemble des meilleurs paramètres :
     n_estimators: 300
     max_features: 'sqrt'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.623

meilleur estimateur RandomForestClassifier(random_state=42)

Accuracy : 0.612
Classification Report
              precision    recall  f1-score   support

       FALSE    0.48936   0.48936   0.48936        47
        TRUE    0.67391   0.57407   0.62000        54
     mixture    0.47727   0.51220   0.49412        41
       other    0.78431   0.86957   0.82474        46

    accuracy                        0.61170       188
   macro avg    0.60622   0.61130   0.60706       188
weighted avg    0.61190   0.61170   0.60998       188

Ensemble des meilleurs paramètres :
     n_estimators: 100
     max_features: 'sqrt'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.647

meilleur estimateur RandomForestClassifier(n_estimators=300,
random_state=42)

Accuracy : 0.612
Classification Report
              precision    recall  f1-score   support

       FALSE    0.51163   0.46809   0.48889        47
        TRUE    0.68889   0.57407   0.62626        54
     mixture    0.44898   0.53659   0.48889        41
       other    0.78431   0.86957   0.82474        46

    accuracy                        0.61170       188
   macro avg    0.60845   0.61208   0.60720       188
```

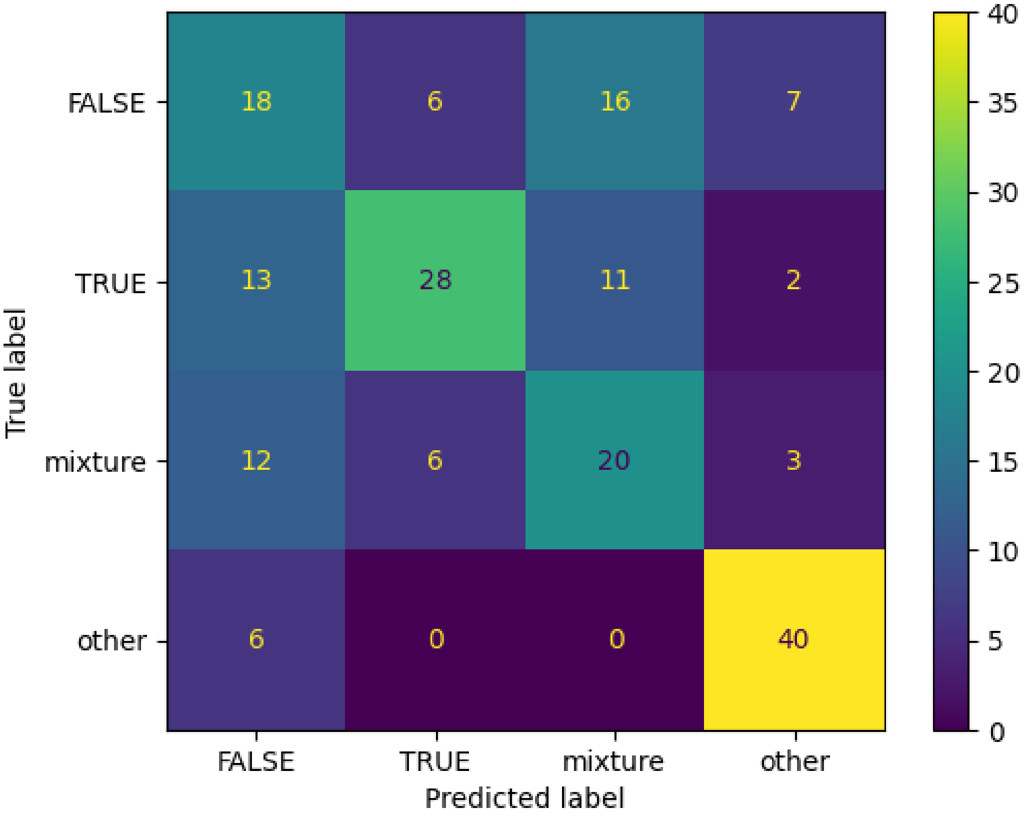
```
weighted avg    0.61560   0.61170   0.61052       188

Ensemble des meilleurs paramètres :
     n_estimators: 300
     max_features: 'sqrt'
```

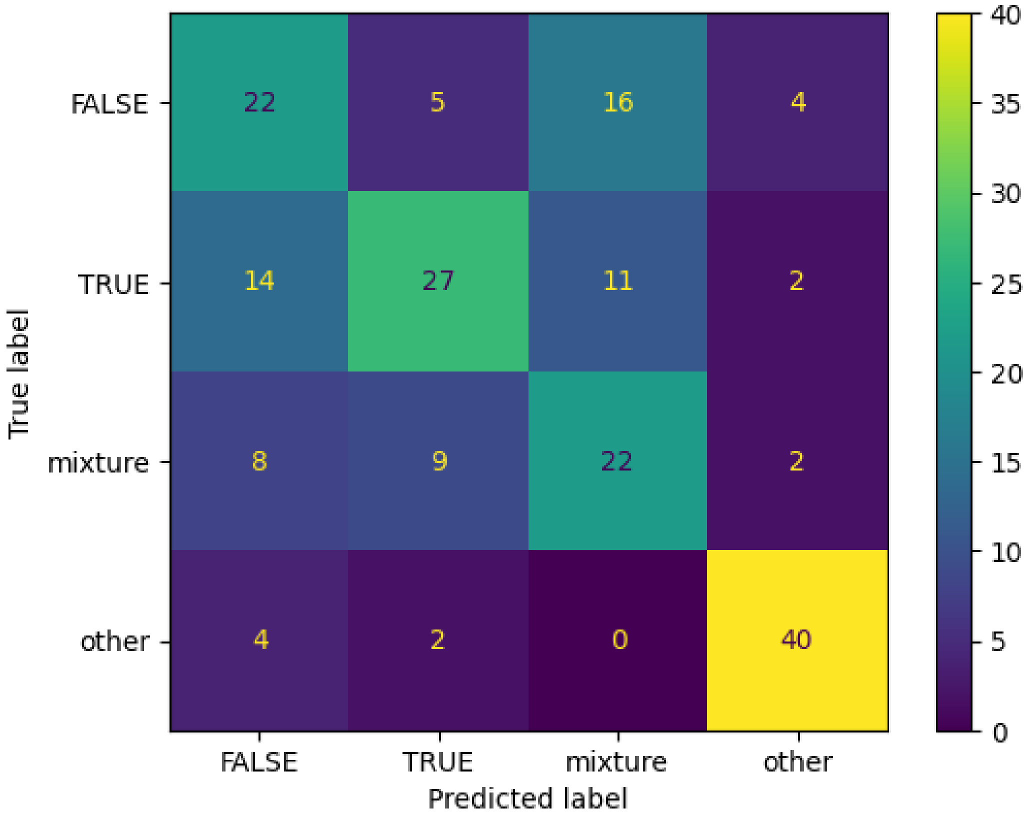



True label
FALSE
TRUE
mixture
other
25
2
16
4
16
28
7
3
13
5
21
2
4
0
2
40
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0

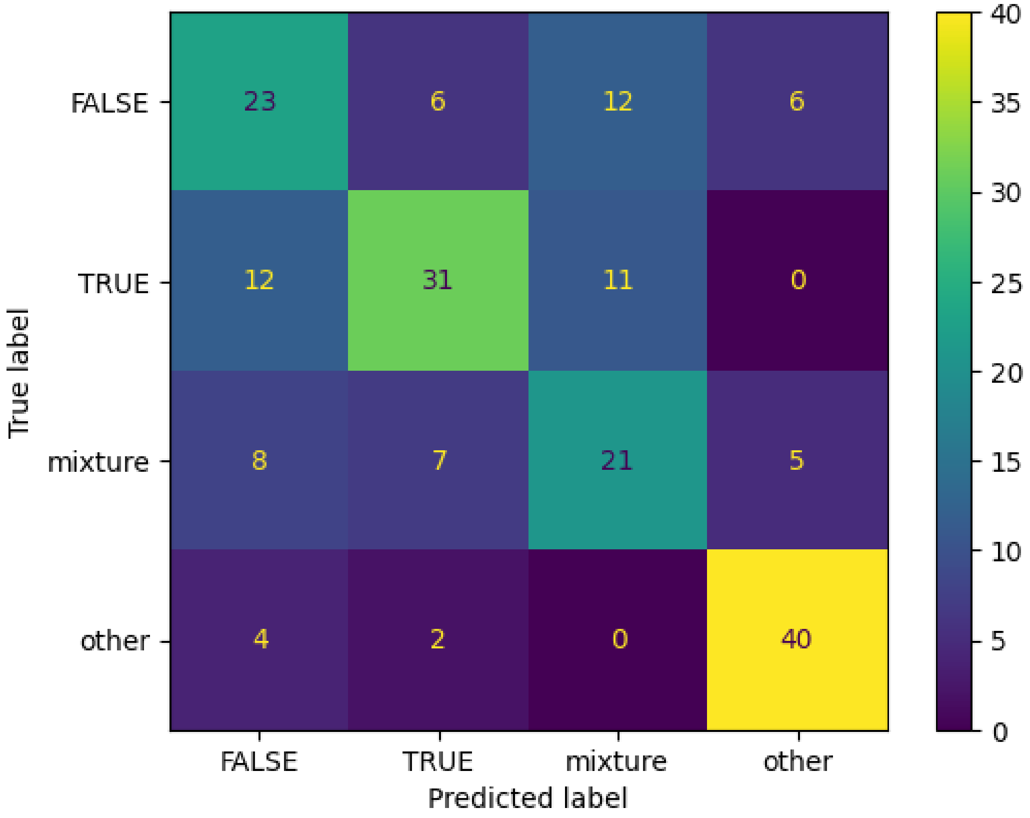

True label
FALSE
TRUE
mixture
other
24
2
17
4
17
28
6
3
8
6
25
2
6
0
0
40
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0


True label
FALSE
TRUE
mixture
other
21
2
21
3
10
25
16
3
5
0
32
4
6
0
1
39
FALSE
TRUE
mixture
other
Predicted label
35
30
25
20
15
10
5
0

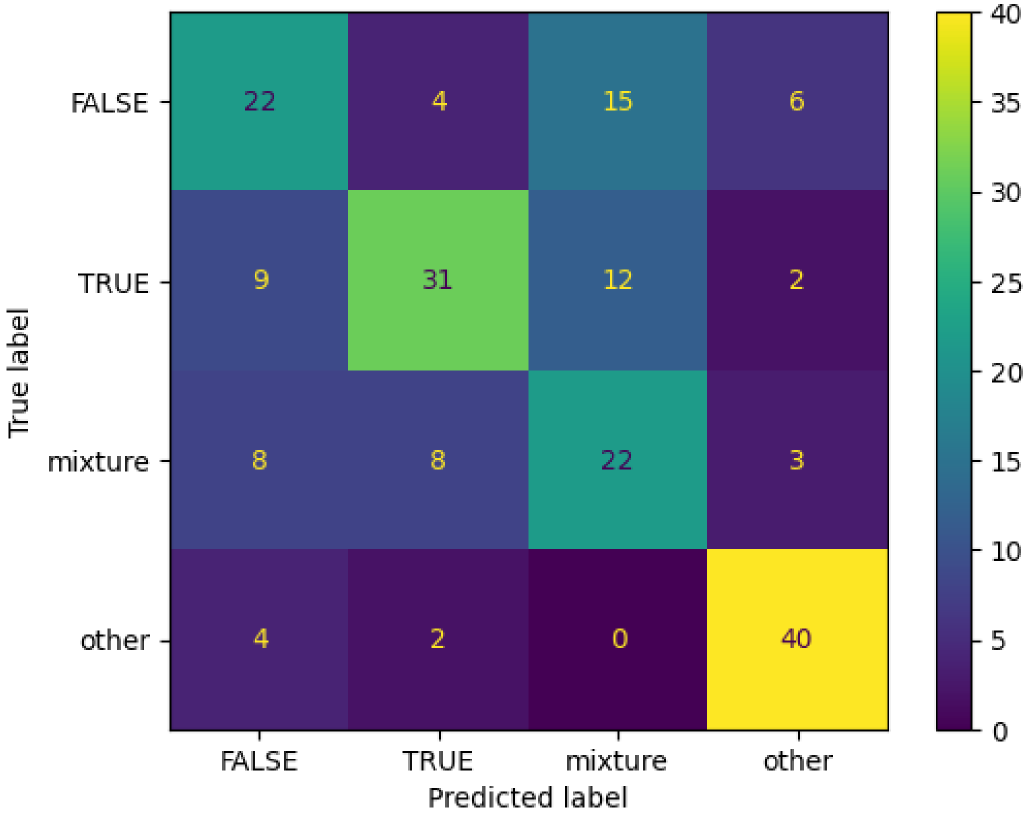

True label
FALSE
TRUE
mixture
other
18
6
16
7
13
28
11
2
12
6
20
3
6
0
0
40
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0



True label
FALSE
TRUE
mixture
other
22
5
16
4
14
27
11
2
8
9
22
2
4
2
0
40
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0



True label
FALSE
TRUE
mixture
other
23
6
12
6
12
31
11
0
8
7
21
5
4
2
0
40
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0



##**Etape 4 : Classification selon la colonne TEXT+TITRE et KEYWORDS (concaténés) :**

Ici, c'est une étape importante, on va tester différents classifieurs, pour chacun des classifieurs, on va appliquer le prétraitement + Vectorisation TfIdf, et on applique une cross_val_score avec un Kfold de 10 fois, par la suite on stocke dans une liste all_results la moyenne des accuracy + l'écart type et on la trie par ordre décroissant de moyenne d'accuracy et d'écart type. on remarque que les 2 meilleurs sont SVM et RF qu'on va séléctionner pour leur appliquer le GridSearch sur les paramètres des prétraitements + leurs hyperparamètres pour pouvoir choisir le meilleur.

```
# Utilisez la méthode ravel() pour transformer y_train en un tableau
unidimensionnel
y_train = np.ravel(y_train)

np.random.seed(42)  # Set the random seed for NumPy

score = 'accuracy'
seed = 7
allresults = []
results = []
names = []


# Liste des modèles à tester
```

```
models = [
    ('MultinomialNB', MultinomialNB()),
    ('LogisticRegression', LogisticRegression(random_state=42))
]


#models.append(('LR', LogisticRegression(solver='lbfgs')))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier(random_state=42)))
models.append(('RF', RandomForestClassifier(random_state=42)))
models.append(('SVM', SVC(random_state=42)))

# Création d'un pipeline pour chaque modèle
pipelines = []
for name,model in models:
    pipeline = Pipeline([
        ('normalize', TextNormalizer()),
        ('tfidf', TfidfVectorizer()),
        (name,model)
    ])
    pipelines.append((name,pipeline))
    #pipeline.fit(X_train_text,y_train)
all_results=[]
scores=[]
for p in pipelines:
    print(p[1])
    # cross validation en 10 fois
    kfold = KFold(n_splits=10,random_state=seed,shuffle=True)

#    print ("Evaluation de ",p)
    start_time = time.time()
    # application de la classification
    cv_results =
cross_val_score(p[1],X_train_text_title_keywords,y_train, cv=kfold,
scoring=score)
    #print("Pour le classifieur",p[0],"on a un score
de",cv_results.mean(),"et un écart type de",cv_results.std())
    scores.append(cv_results)

    all_results.append((p[0],cv_results.mean(),cv_results.std()))
    end_time = time.time()


all_results = sorted(all_results, key=lambda x: (-x[1], -x[2]))
print("all resultats", all_results)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during
```

```
thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)

Pipeline(steps=[('normalize', TextNormalizer()), ('tfidf',
TfidfVectorizer()),
                ('MultinomialNB', MultinomialNB())])
Pipeline(steps=[('normalize', TextNormalizer()), ('tfidf',
TfidfVectorizer()),
                ('LogisticRegression',
LogisticRegression(random_state=42))])
Pipeline(steps=[('normalize', TextNormalizer()), ('tfidf',
TfidfVectorizer()),
                ('KNN', KNeighborsClassifier())])
Pipeline(steps=[('normalize', TextNormalizer()), ('tfidf',
TfidfVectorizer()),
                ('CART', DecisionTreeClassifier(random_state=42))])
Pipeline(steps=[('normalize', TextNormalizer()), ('tfidf',
TfidfVectorizer()),
                ('RF', RandomForestClassifier(random_state=42))])
Pipeline(steps=[('normalize', TextNormalizer()), ('tfidf',
TfidfVectorizer()),
                ('SVM', SVC(random_state=42))])
all resultats [('SVM', 0.6951711711711711, 0.04658233013760278),
('RF', 0.6817837837837838, 0.06062137728405921),
('LogisticRegression', 0.6738378378378378, 0.04823432509826642),
('CART', 0.6202342342342343, 0.03539594124246769), ('MultinomialNB',
0.5936756756756756, 0.07363293823675082), ('KNN', 0.4210810810810811,
0.06143911649190258)]
```

On a un pipeline pour chaque prétraitement différent, on essaye pas mal (miniscule, lemmatisation, miniscule + lemmatisation..) et on stocke le fit_transorm de nos X_train, X_test sur les pipelines dans des listes qui vont contenir tous les fit_transform des pipelines pour chaque classifieur, par la suite on parcourt ces listes là, on itère dessus, et chaque élement de la liste (train) va passer par le GridSearch et puis on predict sur son corresapondant dans liste (test).

```
np.random.seed(42)  # Set the random seed for NumPy

# le plus simple est de faire un test sur differents pipelines.
# pipeline de l'utilisation de CountVectorizer sur le texte avec
differents pre-traitements
CV_brut = Pipeline([('cleaner', TextNormalizer()),
                    ('count_vectorizer',
CountVectorizer(lowercase=False))])
CV_lowcase = Pipeline([('cleaner',
TextNormalizer(removestopwords=False,lowercase=True,

getstemmer=False,removedigit=False)),
                    ('count_vectorizer',
CountVectorizer(lowercase=False))])
```

```
CV_lowStop = Pipeline([('cleaner',
TextNormalizer(removestopwords=True,lowercase=True,

getstemmer=False,removedigit=False)),
                    ('count_vectorizer',
CountVectorizer(lowercase=False))])

CV_lowStopstem = Pipeline([('cleaner',
TextNormalizer(removestopwords=True,lowercase=True,

getstemmer=True,removedigit=False)),
                    ('count_vectorizer',
CountVectorizer(lowercase=False))])

# pipeline de l'utilisation de TfidfVectorizer avec differents pre-
traitements
TFIDF_brut = Pipeline ([('cleaner', TextNormalizer()),
                    ('tfidf_vectorizer',
TfidfVectorizer(lowercase=False))])

TFIDF_lowcase = Pipeline([('cleaner',
TextNormalizer(removestopwords=False,lowercase=True,

getstemmer=False,removedigit=False)),
                    ('tfidf_vectorizer',
TfidfVectorizer(lowercase=False))])
TFIDF_lowStop = Pipeline([('cleaner',
TextNormalizer(removestopwords=True,lowercase=True,

getstemmer=False,removedigit=False)),
                    ('tfidf_vectorizer',
TfidfVectorizer(lowercase=False))])

TFIDF_lowStopstem = Pipeline([('cleaner',
TextNormalizer(removestopwords=True,lowercase=True,

getstemmer=True,removedigit=False)),
                    ('tfidf_vectorizer',
TfidfVectorizer(lowercase=False))])


# Liste de tous les modeles à tester
all_models = [
    ("CV_brut", CV_brut),
    ("CV_lowcase", CV_lowcase),
    ("CV_lowStop", CV_lowStop),
    ("CV_lowStopstem",CV_lowStopstem),
    ("TFIDF_lowcase", TFIDF_lowcase),
    ("TFIDF_lowStop", TFIDF_lowStop),
    ("TFIDF_lowStopstem",TFIDF_lowStopstem),
    ("TFIDF_brut", TFIDF_brut)
]

X_train_text_title_keywords_SVC = []
X_test_text_title_keywords_SVC = []

X_train_text_title_keywords_RandomForestClassifier = []
X_test_text_title_keywords_RandomForestClassifier = []


for name, pipeline in all_models :

X_train_text_title_keywords_SVC.append(pipeline.fit_transform(X_train_
text_title_keywords).toarray())

X_test_text_title_keywords_SVC.append(pipeline.transform(X_test_text_t
itle_keywords).toarray())

X_train_text_title_keywords_RandomForestClassifier.append(pipeline.fit
_transform(X_train_text_title_keywords).toarray())

X_test_text_title_keywords_RandomForestClassifier.append(pipeline.tran
sform(X_test_text_title_keywords).toarray())




models = {
    'SVC': SVC(random_state=42),
    'RandomForestClassifier': RandomForestClassifier(random_state=42)
}


params = {'SVC': [{'C': [0.001, 0.01, 0.1, 1,2]},
             {'gamma': [0.001, 0.01, 0.1,1]},
             {'kernel': ['linear', 'rbf']}],
    'RandomForestClassifier': [{'n_estimators': [10, 50, 100, 200,
300]},
                              {'max_features': ['auto', 'sqrt',
'log2']}],
}

for model_name, model in models.items():
    score='accuracy'
    X_train_text_title_keywords = eval('X_train_text_title_keywords_'
+ model_name)
    X_test_text_title_keywords = eval('X_test_text_title_keywords_' +
model_name)
```

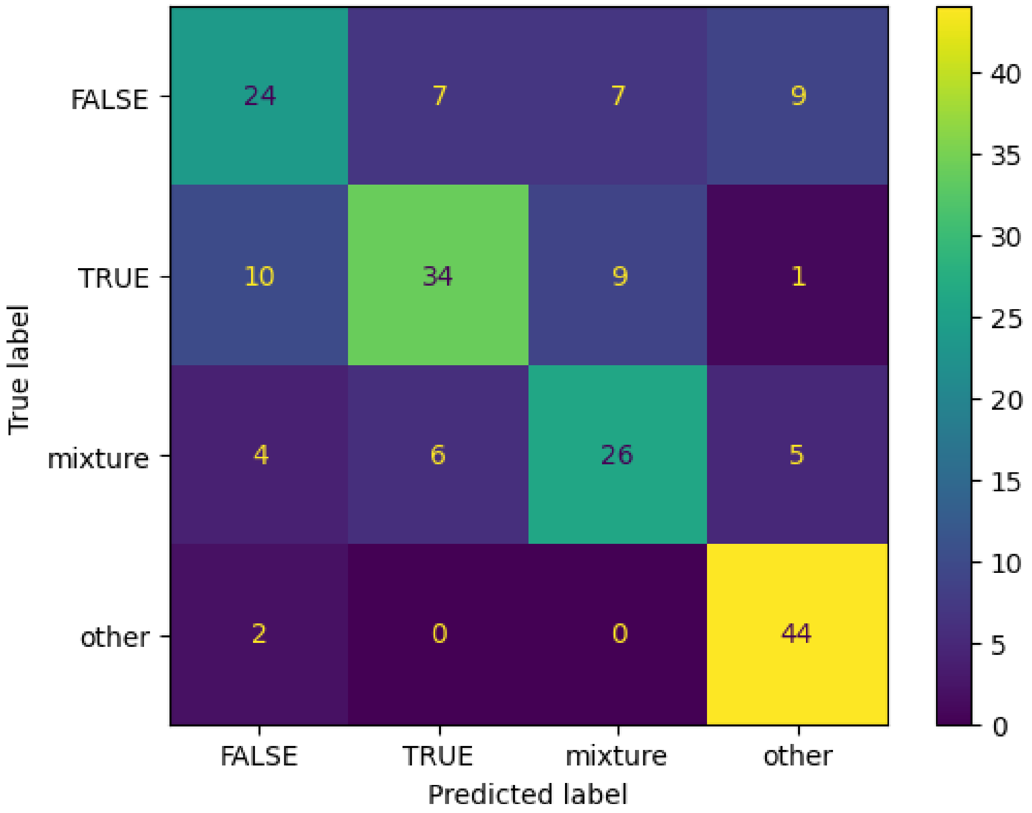
```
    for i in range (len(X_train_text_title_keywords)):
      grid_search = GridSearchCV(model, params[model_name], n_jobs=-1,
verbose=1,scoring=score)
      print("grid search fait")
      grid_search.fit(X_train_text_title_keywords[i],y_train)
      print ('meilleur score %0.3f'%(grid_search.best_score_),'\n')
      print ('meilleur estimateur',grid_search.best_estimator_,'\n')
      y_pred = grid_search.predict(X_test_text_title_keywords[i])
      MyshowAllScores(y_test,y_pred)

      print("Ensemble des meilleurs paramètres :")
      best_parameters = grid_search.best_estimator_.get_params()
      for param_dict in params[model_name]:
        for param_name, param_value in param_dict.items():
            print("\t%s: %r" % (param_name,
best_parameters[param_name]))
```

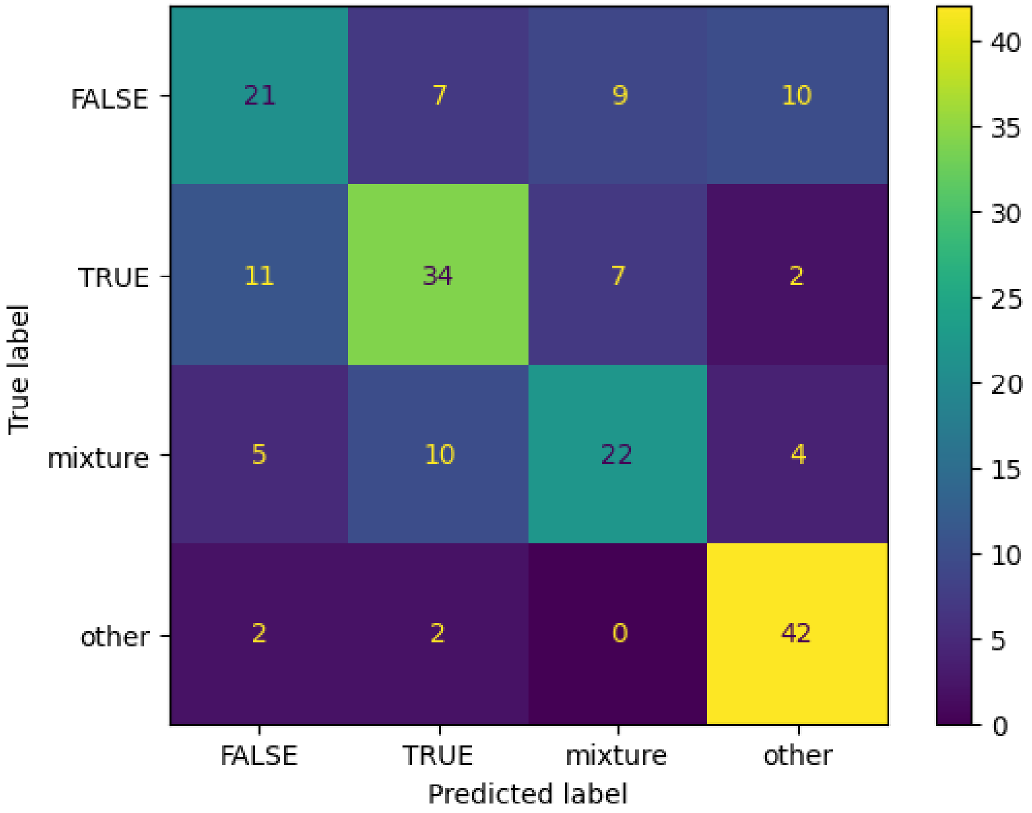
```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during
thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

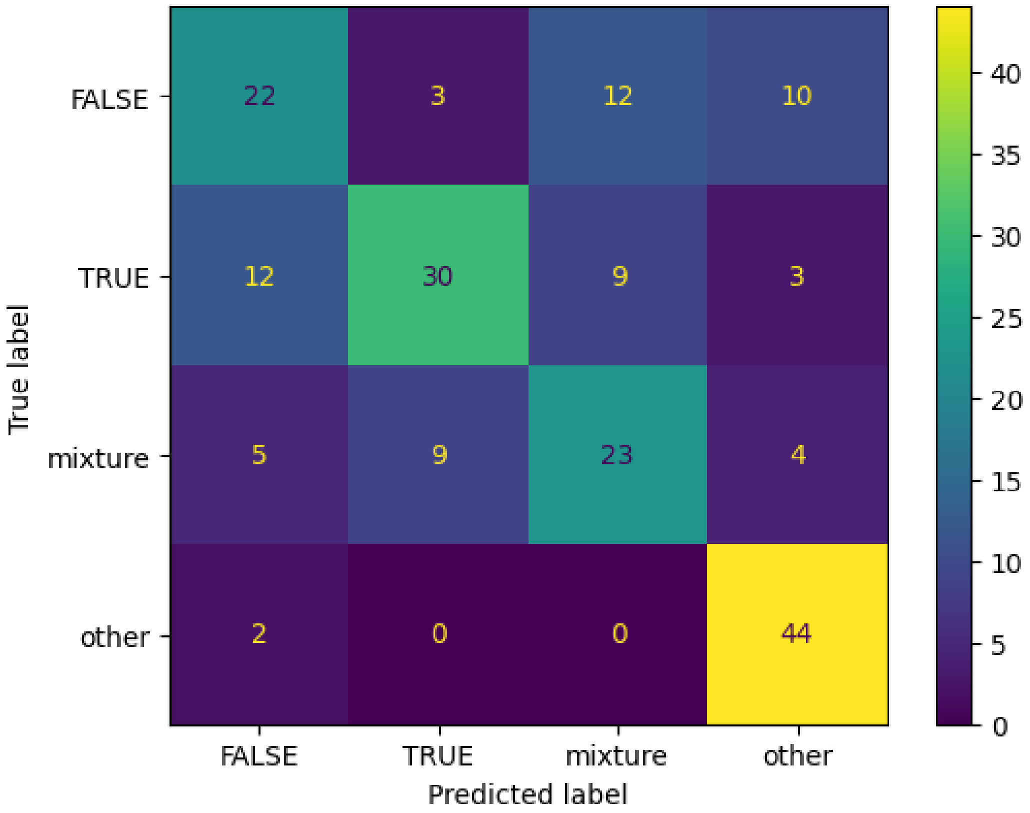
```
grid search fait
Fitting 5 folds for each of 11 candidates, totalling 55 fits
meilleur score 0.662

meilleur estimateur SVC(kernel='linear', random_state=42)

Accuracy : 0.681
Classification Report
              precision    recall  f1-score   support

       FALSE    0.60000   0.51064   0.55172        47
        TRUE    0.72340   0.62963   0.67327        54
     mixture    0.61905   0.63415   0.62651        41
       other    0.74576   0.95652   0.83810        46

    accuracy                        0.68085       188
   macro avg    0.67205   0.68273   0.67240       188
weighted avg    0.67527   0.68085   0.67301       188

Ensemble des meilleurs paramètres :
      C: 1.0
      gamma: 'scale'
      kernel: 'linear'
grid search fait
Fitting 5 folds for each of 11 candidates, totalling 55 fits
meilleur score 0.654
meilleur estimateur SVC(kernel='linear', random_state=42)

Accuracy : 0.633
Classification Report
              precision    recall  f1-score   support

       FALSE    0.53846   0.44681   0.48837        47
        TRUE    0.64151   0.62963   0.63551        54
     mixture    0.57895   0.53659   0.55696        41
       other    0.72414   0.91304   0.80769        46

    accuracy                        0.63298       188
   macro avg    0.62076   0.63152   0.62214       188
weighted avg    0.62232   0.63298   0.62373       188

Ensemble des meilleurs paramètres :
      C: 1.0
      gamma: 'scale'
      kernel: 'linear'
grid search fait
Fitting 5 folds for each of 11 candidates, totalling 55 fits
meilleur score 0.651

meilleur estimateur SVC(kernel='linear', random_state=42)

Accuracy : 0.633
Classification Report
              precision    recall  f1-score   support

       FALSE    0.53659   0.46809   0.50000        47
        TRUE    0.71429   0.55556   0.62500        54
     mixture    0.52273   0.56098   0.54118        41
       other    0.72131   0.95652   0.82243        46

    accuracy                        0.63298       188
   macro avg    0.62373   0.63528   0.62215       188
weighted avg    0.62980   0.63298   0.62378       188

Ensemble des meilleurs paramètres :
      C: 1.0
      gamma: 'scale'
      kernel: 'linear'
grid search fait
Fitting 5 folds for each of 11 candidates, totalling 55 fits
meilleur score 0.639

meilleur estimateur SVC(kernel='linear', random_state=42)
```

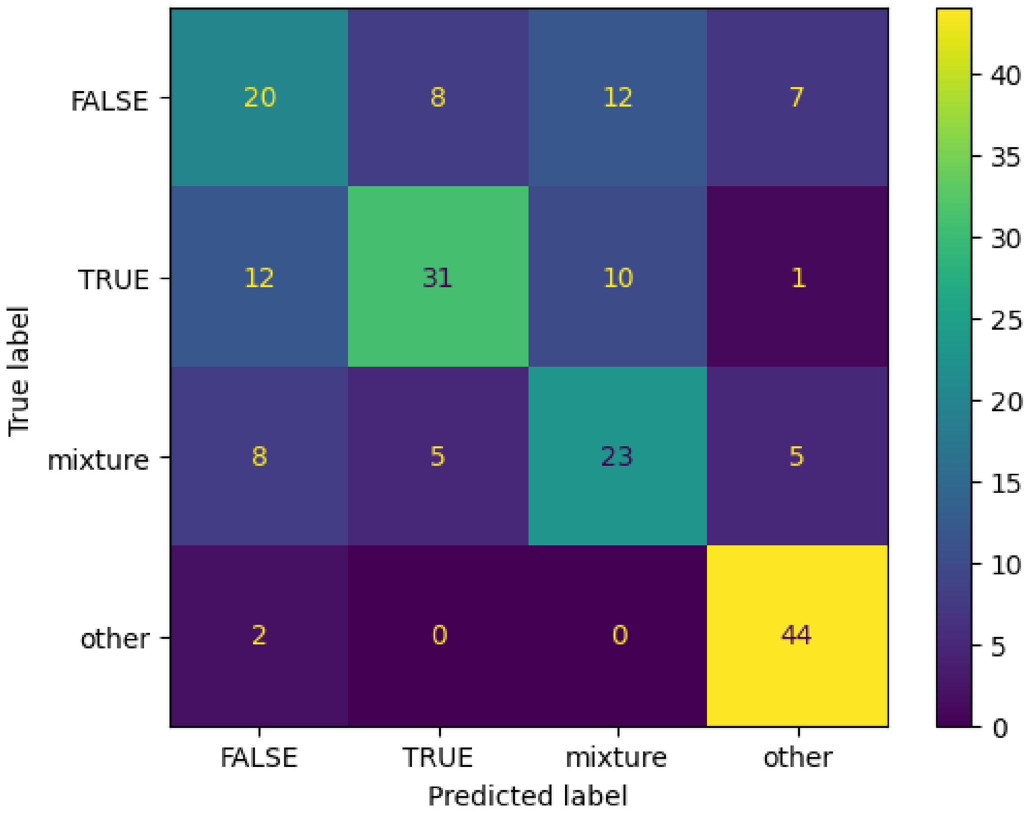
```
Accuracy : 0.628
Classification Report
              precision    recall  f1-score   support

       FALSE    0.47619   0.42553   0.44944        47
        TRUE    0.70455   0.57407   0.63265        54
     mixture    0.51111   0.56098   0.53488        41
       other    0.77193   0.95652   0.85437        46

    accuracy                        0.62766       188
   macro avg    0.61594   0.62928   0.61784       188
weighted avg    0.62176   0.62766   0.61978       188

Ensemble des meilleurs paramètres :
     C: 1.0
     gamma: 'scale'
     kernel: 'linear'
grid search fait
Fitting 5 folds for each of 11 candidates, totalling 55 fits
meilleur score 0.691

meilleur estimateur SVC(C=2, random_state=42)

Accuracy : 0.729
Classification Report
              precision    recall  f1-score   support

       FALSE    0.60784   0.65957   0.63265        47
        TRUE    0.80000   0.59259   0.68085        54
     mixture    0.62963   0.82927   0.71579        41
       other    0.93023   0.86957   0.89888        46

    accuracy                        0.72872       188
   macro avg    0.74193   0.73775   0.73204       188
weighted avg    0.74667   0.72872   0.72977       188

Ensemble des meilleurs paramètres :
     C: 2
     gamma: 'scale'
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 11 candidates, totalling 55 fits
meilleur score 0.697

meilleur estimateur SVC(C=2, random_state=42)

Accuracy : 0.729
Classification Report
              precision    recall  f1-score   support
       FALSE    0.64706   0.70213   0.67347        47
        TRUE    0.83333   0.55556   0.66667        54
     mixture    0.58621   0.82927   0.68687        41
       other    0.93023   0.86957   0.89888        46

    accuracy                        0.72872       188
   macro avg    0.74921   0.73913   0.73147       188
weighted avg    0.75658   0.72872   0.72959       188

Ensemble des meilleurs paramètres :
     C: 2
     gamma: 'scale'
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 11 candidates, totalling 55 fits
meilleur score 0.693

meilleur estimateur SVC(C=1, random_state=42)

Accuracy : 0.718
Classification Report
              precision    recall  f1-score   support

       FALSE    0.62745   0.68085   0.65306        47
        TRUE    0.85294   0.53704   0.65909        54
     mixture    0.56667   0.82927   0.67327        41
       other    0.93023   0.86957   0.89888        46

    accuracy                        0.71809       188
   macro avg    0.74432   0.72918   0.72107       188
weighted avg    0.75305   0.71809   0.71935       188

Ensemble des meilleurs paramètres :
     C: 1
     gamma: 'scale'
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 11 candidates, totalling 55 fits
meilleur score 0.697

meilleur estimateur SVC(C=2, random_state=42)

Accuracy : 0.739
Classification Report
              precision    recall  f1-score   support

       FALSE    0.63265   0.65957   0.64583        47
        TRUE    0.80000   0.59259   0.68085        54
     mixture    0.62963   0.82927   0.71579        41
       other    0.93333   0.91304   0.92308        46

    accuracy                        0.73936       188
   macro avg    0.74890   0.74862   0.74139       188
weighted avg    0.75363   0.73936   0.73898       188

Ensemble des meilleurs paramètres :
     C: 2
     gamma: 'scale'
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.671

meilleur estimateur RandomForestClassifier(max_features='log2',
random_state=42)

Accuracy : 0.691
Classification Report
              precision    recall  f1-score   support

       FALSE    0.56000   0.59574   0.57732        47
        TRUE    0.70588   0.66667   0.68571        54
     mixture    0.60465   0.63415   0.61905        41
       other    0.90909   0.86957   0.88889        46

    accuracy                        0.69149       188
   macro avg    0.69491   0.69153   0.69274       188
weighted avg    0.69706   0.69149   0.69379       188

Ensemble des meilleurs paramètres :
     n_estimators: 100
     max_features: 'log2'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.682

meilleur estimateur RandomForestClassifier(n_estimators=200,
random_state=42)

Accuracy : 0.713
Classification Report
              precision    recall  f1-score   support

       FALSE    0.57407   0.65957   0.61386        47
        TRUE    0.64286   0.66667   0.65455        54
     mixture    0.75758   0.60976   0.67568        41
       other    0.93333   0.91304   0.92308        46
    accuracy                        0.71277       188
   macro avg    0.72696   0.71226   0.71679       188
weighted avg    0.72175   0.71277   0.71469       188

Ensemble des meilleurs paramètres :
     n_estimators: 200
     max_features: 'sqrt'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.679

meilleur estimateur RandomForestClassifier(n_estimators=300,
random_state=42)

Accuracy : 0.686
Classification Report
              precision    recall  f1-score   support

       FALSE    0.53571   0.63830   0.58252        47
        TRUE    0.66667   0.66667   0.66667        54
     mixture    0.67647   0.56098   0.61333        41
       other    0.90909   0.86957   0.88889        46

    accuracy                        0.68617       188
   macro avg    0.69699   0.68388   0.68785       188
weighted avg    0.69538   0.68617   0.68837       188

Ensemble des meilleurs paramètres :
     n_estimators: 300
     max_features: 'sqrt'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.684

meilleur estimateur RandomForestClassifier(n_estimators=300,
random_state=42)

Accuracy : 0.697
Classification Report
              precision    recall  f1-score   support

       FALSE    0.52083   0.53191   0.52632        47
        TRUE    0.70370   0.70370   0.70370        54
     mixture    0.66667   0.68293   0.67470        41
       other    0.90909   0.86957   0.88889        46

    accuracy                        0.69681       188
   macro avg    0.70007   0.69703   0.69840       188
```

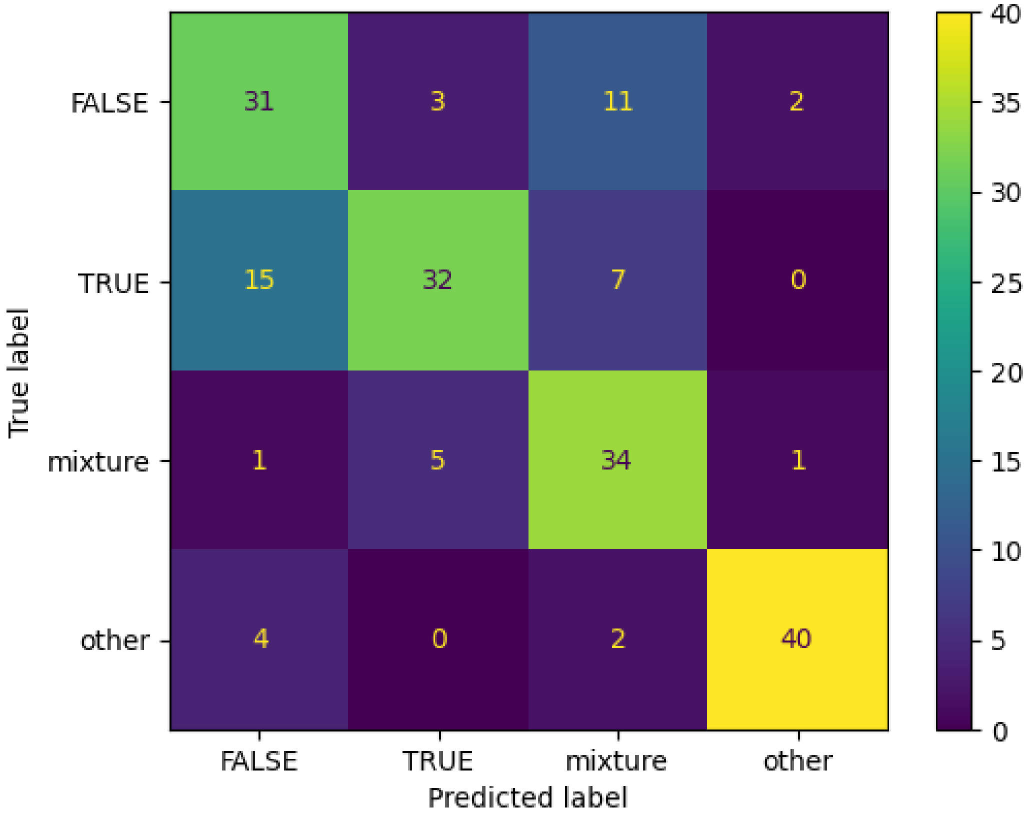
```
weighted avg    0.70016   0.69681   0.69834       188

Ensemble des meilleurs paramètres :
      n_estimators: 300
      max_features: 'sqrt'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.676

meilleur estimateur RandomForestClassifier(n_estimators=200,
random_state=42)

Accuracy : 0.702
Classification Report
              precision    recall  f1-score   support

       FALSE    0.49091   0.57447   0.52941        47
        TRUE    0.80000   0.66667   0.72727        54
     mixture    0.67442   0.70732   0.69048        41
       other    0.88889   0.86957   0.87912        46

    accuracy                        0.70213       188
   macro avg    0.71355   0.70450   0.70657       188
weighted avg    0.71709   0.70213   0.70694       188

Ensemble des meilleurs paramètres :
      n_estimators: 200
      max_features: 'sqrt'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.685

meilleur estimateur RandomForestClassifier(n_estimators=300,
random_state=42)

Accuracy : 0.729
Classification Report
              precision    recall  f1-score   support

       FALSE    0.57143   0.68085   0.62136        47
        TRUE    0.81395   0.64815   0.72165        54
     mixture    0.65116   0.68293   0.66667        41
       other    0.91304   0.91304   0.91304        46

    accuracy                        0.72872       188
   macro avg    0.73740   0.73124   0.73068       188
weighted avg    0.74207   0.72872   0.73142       188

Ensemble des meilleurs paramètres :
     n_estimators: 300
     max_features: 'sqrt'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.683

meilleur estimateur RandomForestClassifier(n_estimators=200,
random_state=42)

Accuracy : 0.681
Classification Report
              precision    recall  f1-score   support

       FALSE    0.54902   0.59574   0.57143        47
        TRUE    0.68750   0.61111   0.64706        54
     mixture    0.58696   0.65854   0.62069        41
       other    0.93023   0.86957   0.89888        46

    accuracy                        0.68085       188
   macro avg    0.68843   0.68374   0.68451       188
weighted avg    0.69034   0.68085   0.68402       188

Ensemble des meilleurs paramètres :
     n_estimators: 200
     max_features: 'sqrt'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.694

meilleur estimateur RandomForestClassifier(n_estimators=200,
random_state=42)

Accuracy : 0.718
Classification Report
              precision    recall  f1-score   support

       FALSE    0.56604   0.63830   0.60000        47
        TRUE    0.71429   0.64815   0.67961        54
     mixture    0.69767   0.73171   0.71429        41
       other    0.93023   0.86957   0.89888        46

    accuracy                        0.71809       188
   macro avg    0.72706   0.72193   0.72319       188
weighted avg    0.72644   0.71809   0.72092       188

Ensemble des meilleurs paramètres :
     n_estimators: 200
     max_features: 'sqrt'
```

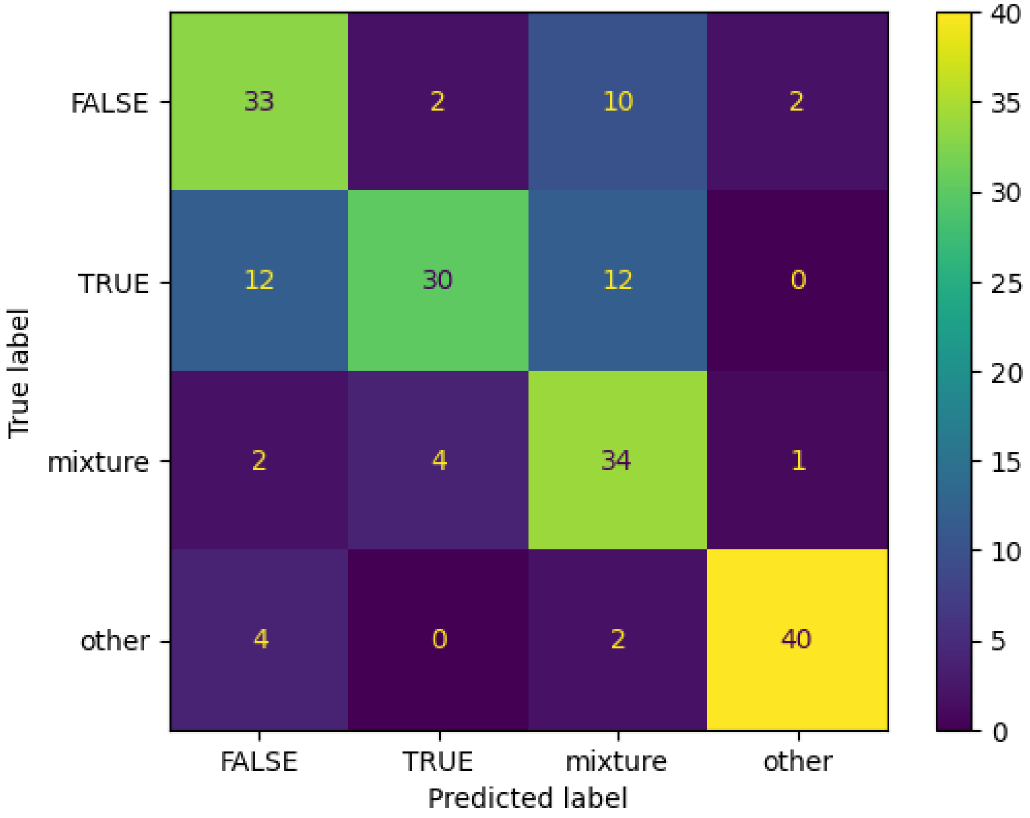

True label
FALSE
TRUE
mixture
other
24 7 7 9
10 34 9 1
4 6 26 5
2 0 0 44
FALSE
TRUE
mixture
other
Predicted label
0
5
10
15
20
25
30
35
40

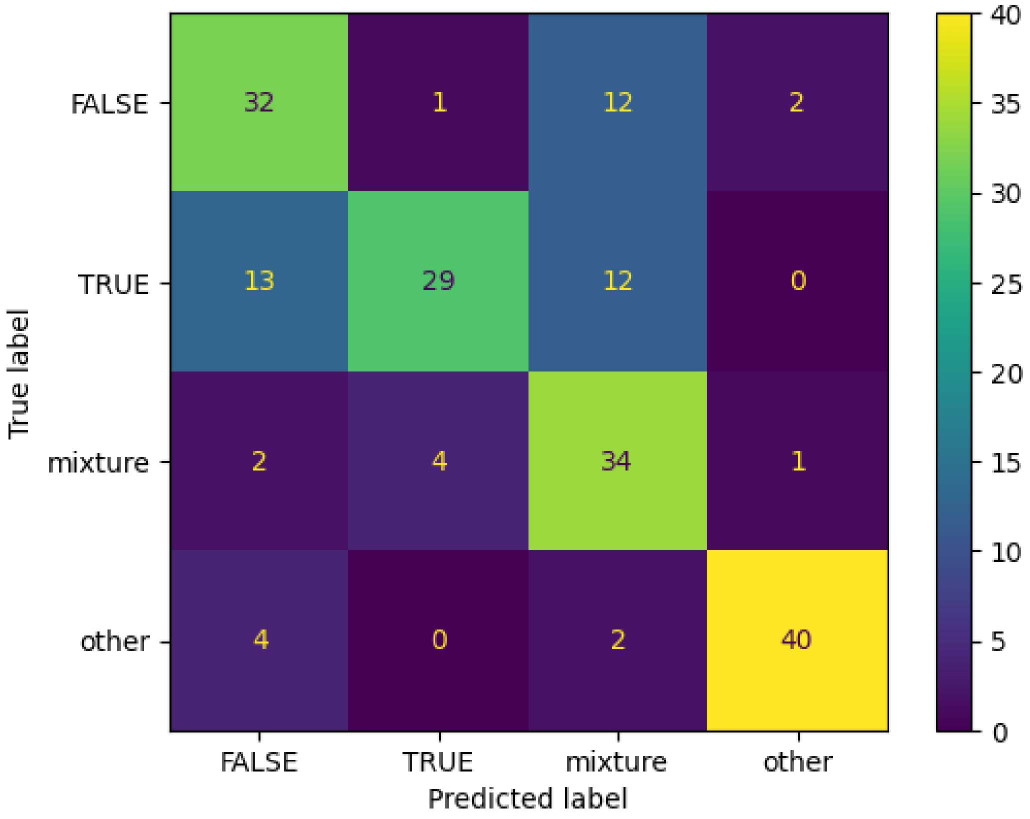

True label
FALSE
TRUE
mixture
other
21 7 9 10
11 34 7 2
5 10 22 4
2 2 0 42
FALSE
TRUE
mixture
other
Predicted label
0
5
10
15
20
25
30
35
40

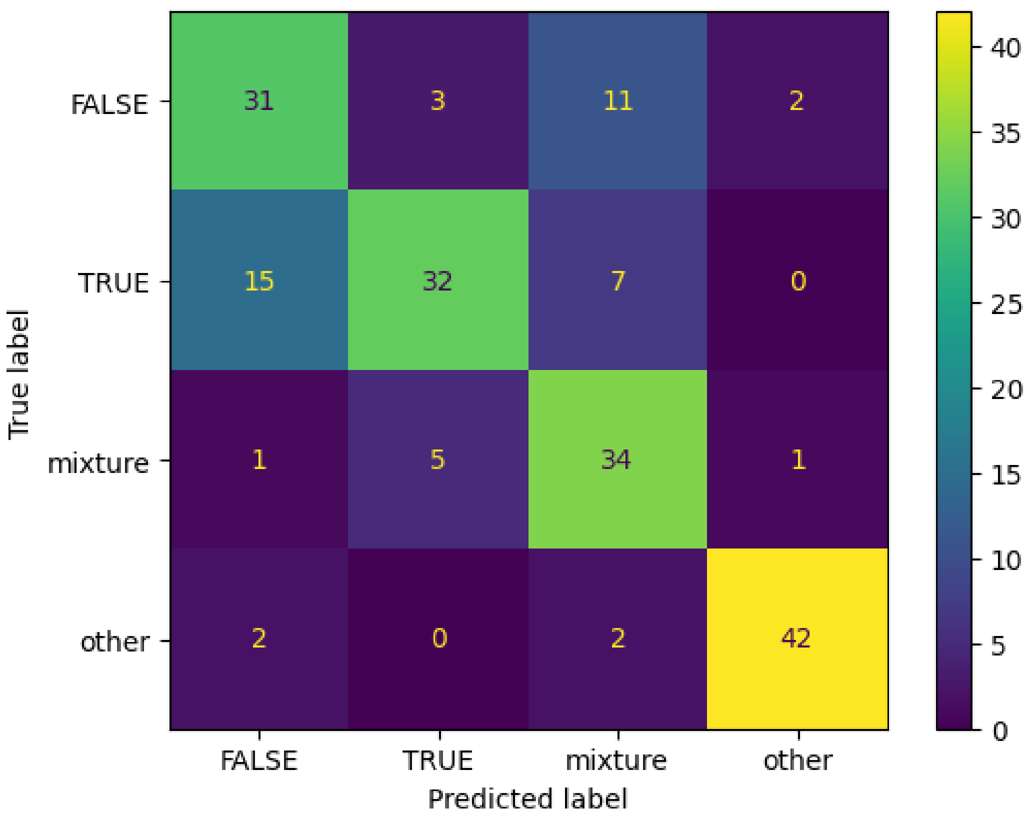

True label
Predicted label
FALSE
TRUE
mixture
other
22 3 12 10
12 30 9 3
5 9 23 4
2 0 0 44
FALSE
TRUE
mixture
other
0
5
10
15
20
25
30
35
40

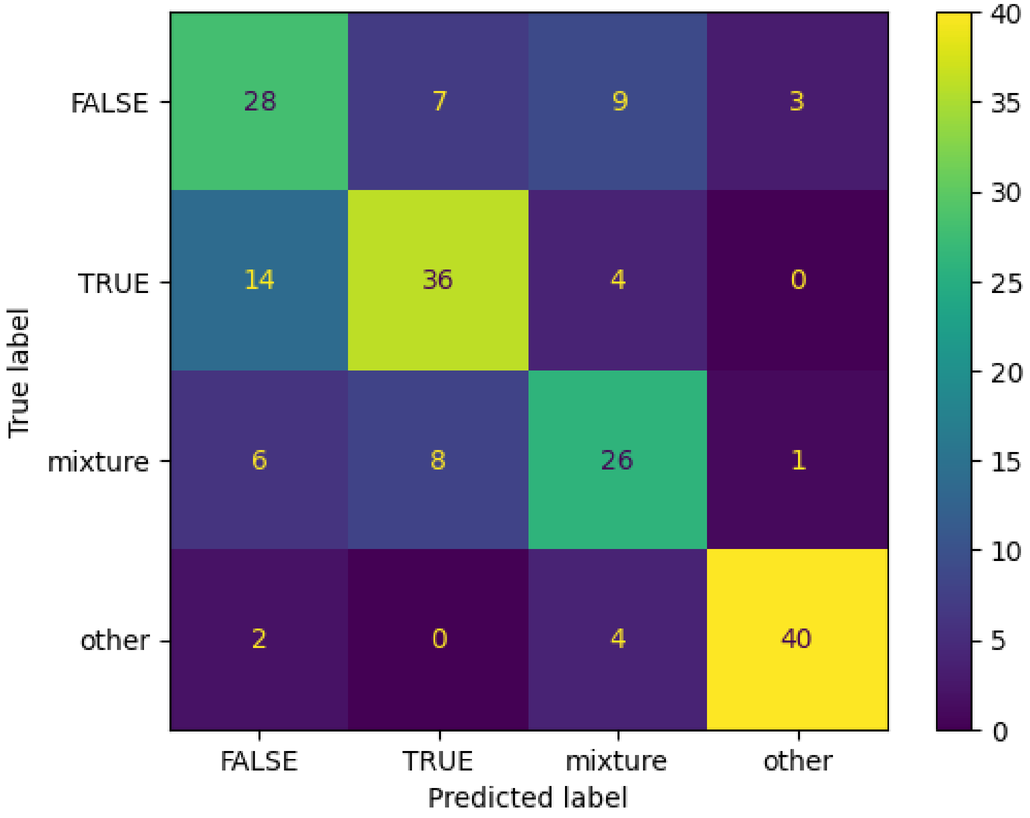

True label
Predicted label
FALSE
TRUE
mixture
other
20 8 12 7
12 31 10 1
8 5 23 5
2 0 0 44
FALSE
TRUE
mixture
other
0
5
10
15
20
25
30
35
40

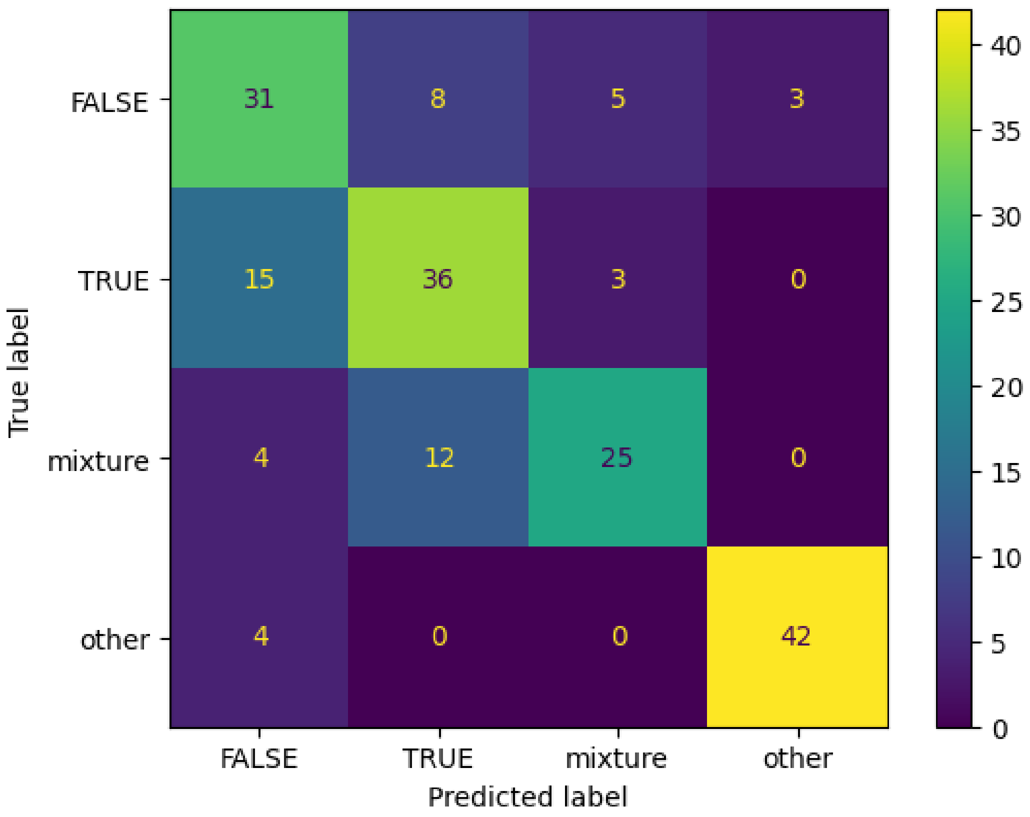

FALSE
TRUE
mixture
other
True label
31
3
11
2
15
32
7
0
1
5
34
1
4
0
2
40
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0

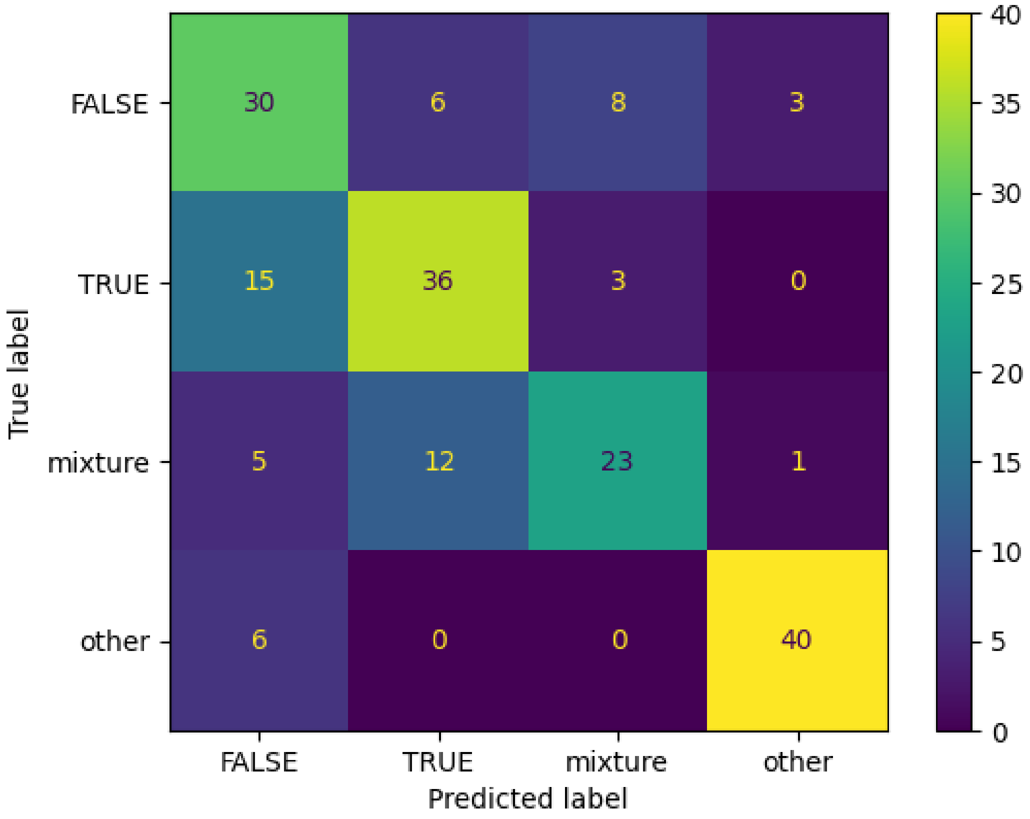

FALSE
TRUE
mixture
other
True label
33
2
10
2
12
30
12
0
2
4
34
1
4
0
2
40
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0

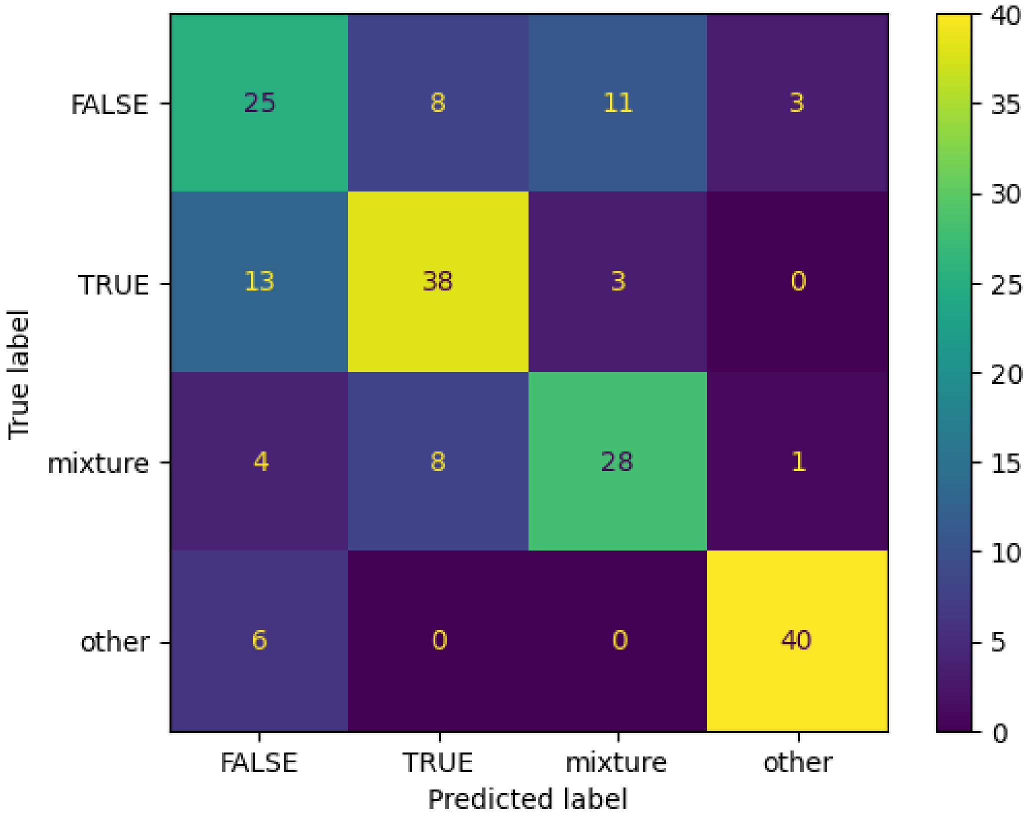

FALSE
TRUE
mixture
other
True label
32
1
12
2
13
29
12
0
2
4
34
1
4
0
2
40
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0

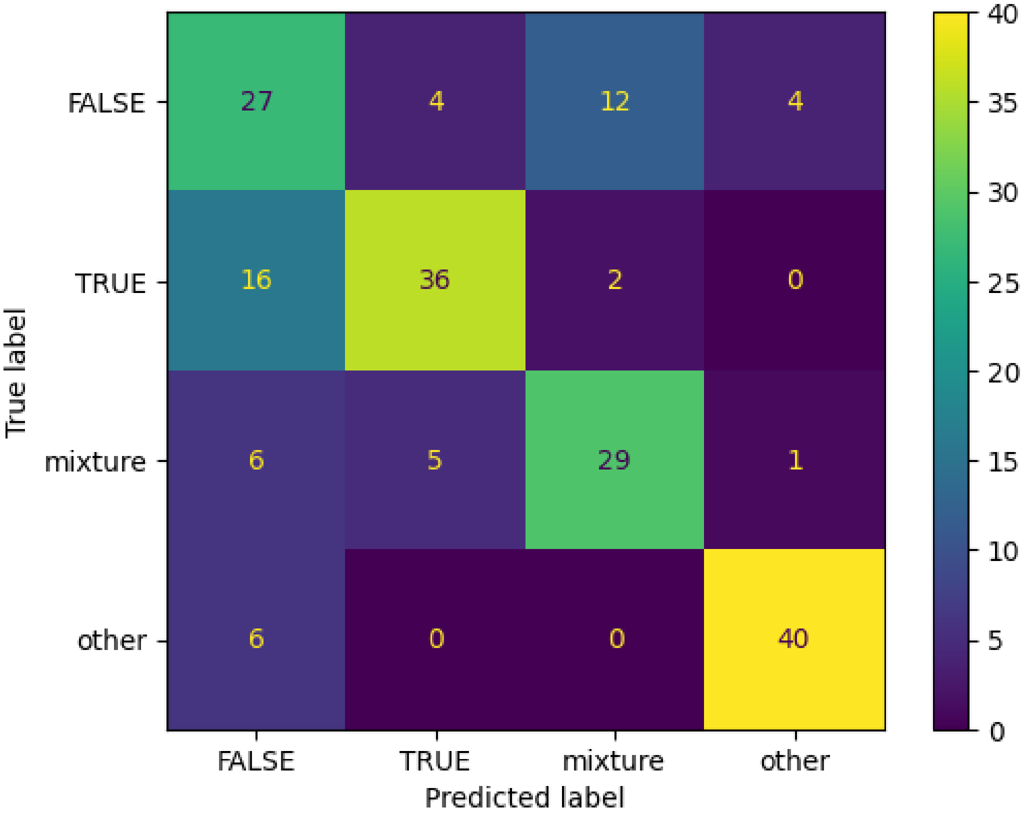

FALSE
TRUE
mixture
other
True label
31
3
11
2
15
32
7
0
1
5
34
1
2
0
2
42
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0

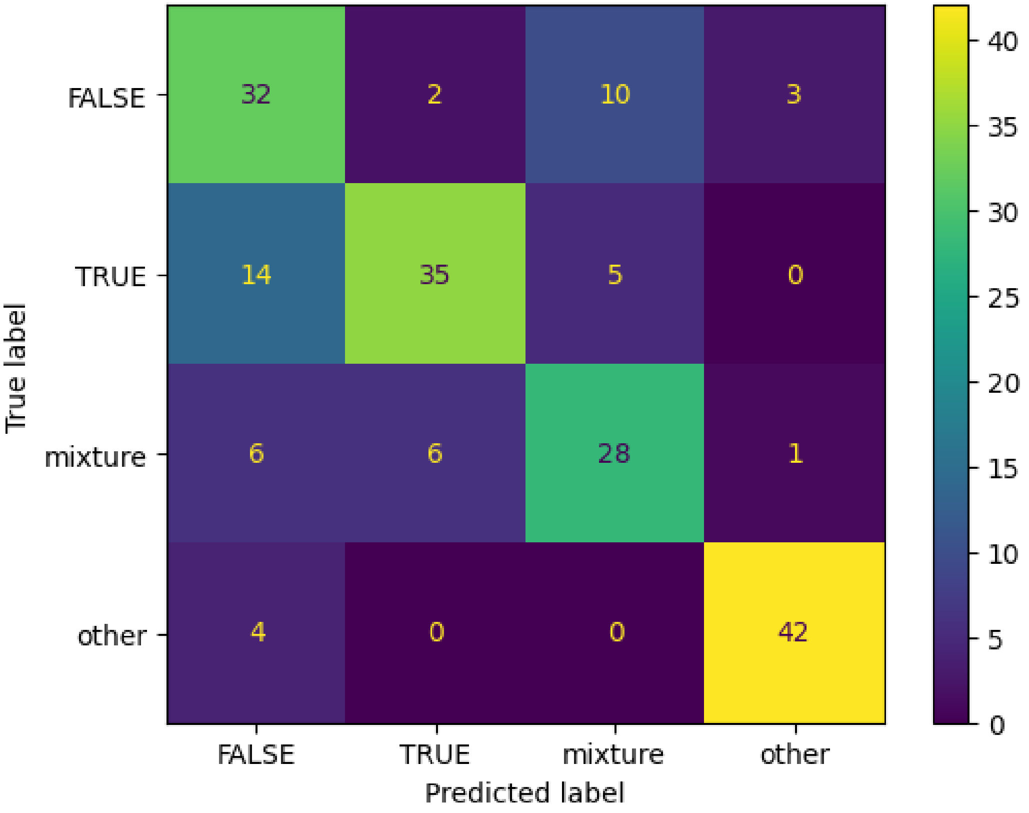

True label
FALSE
TRUE
mixture
other
28
7
9
3
14
36
4
0
6
8
26
1
2
0
4
40
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0

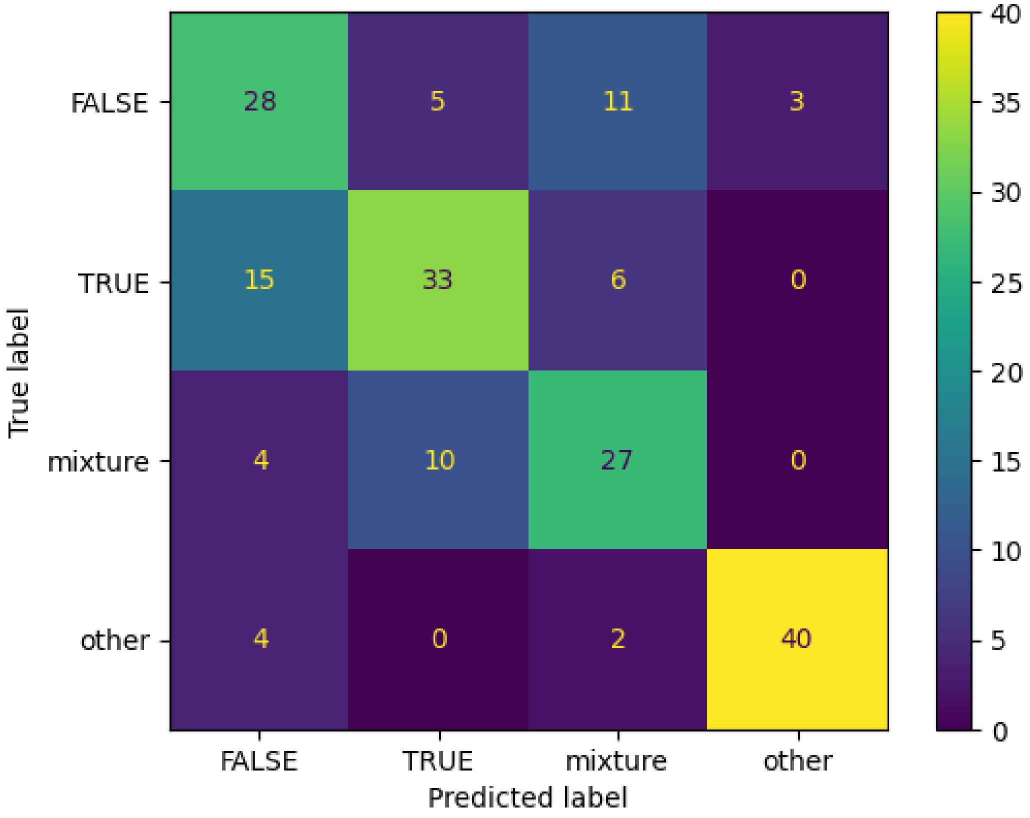

True label
FALSE
TRUE
mixture
other
31
8
5
3
15
36
3
0
4
12
25
0
4
0
0
42
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0

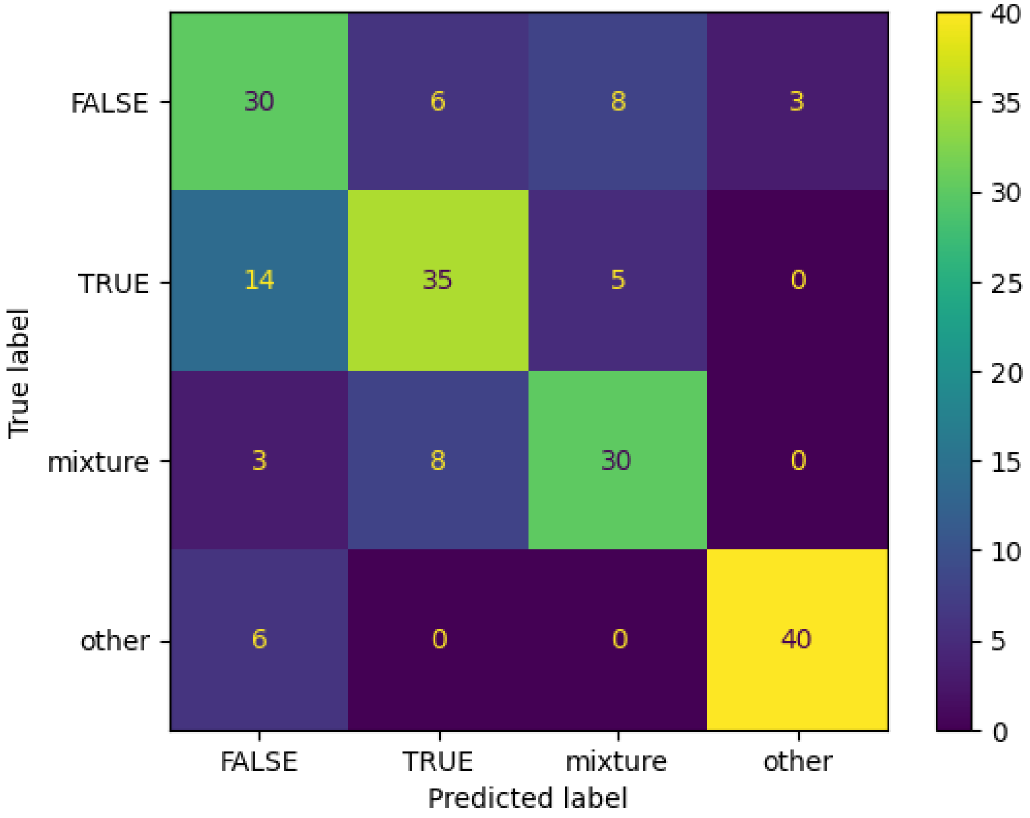

True label
FALSE
TRUE
mixture
other
30
6
8
3
15
36
3
0
5
12
23
1
6
0
0
40
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0


True label
FALSE
TRUE
mixture
other
25
8
11
3
13
38
3
0
4
8
28
1
6
0
0
40
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0


FALSE
TRUE
mixture
other
True label
27 4 12 4
16 36 2 0
6 5 29 1
6 0 0 40
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0


FALSE
TRUE
mixture
other
True label
32 2 10 3
14 35 5 0
6 6 28 1
4 0 0 42
FALSE
TRUE
mixture
other
Predicted label
40
35
30
25
20
15
10
5
0


True label
FALSE
TRUE
mixture
other
28
5
11
3
15
33
6
0
4
10
27
0
4
0
2
40
FALSE
TRUE
mixture
other
Predicted label
0
5
10
15
20
25
30
35
40


True label
FALSE
TRUE
mixture
other
30
6
8
3
14
35
5
0
3
8
30
0
6
0
0
40
FALSE
TRUE
mixture
other
Predicted label
0
5
10
15
20
25
30
35
40