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

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

```
#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
```

```
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
```

```
/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)
[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 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/')
```

```
/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)
```

```
Drive already mounted at /content/gdrive/; to attempt to forcibly
remount, call drive.mount("/content/gdrive/", force_remount=True).
```

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
 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'
 Visualisation_Donnees_2D_3D.ipynb
```

```
/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)
```

```
{"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()
```

```
/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)
[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]   Package punkt is already up-to-date!
```

- 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
```

```
/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 1 : Préparer les données**

- Load et preparer les données à partir des 2 fichiers csv
- Sélectionner que les lignes où on a True, False ou bien Other
- Après en créant une nouvelle colonne "regrouped" si la valeur de la colonne rating est true ou bien false on mettra TRUE/FALSE sinon on laisse OTHER

```
#Ici je cherche à séléctionner que les labels TRUE et FALSE, donc les
LIGNES qui contiennent au rating TRUE et FALSE uniquement, le reste on
enlève

dftrain = pd.read_csv("/content/gdrive/MyDrive/Colab
Notebooks/newsTrain2.csv", names=['id','text','title','rating'],
header=0,sep=',', encoding='utf8')
dftrain.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
dftrain = pd.concat([dftrain, dftrain2], ignore_index=True)
dftrain = dftrain.loc[dftrain['rating'].isin(['TRUE','FALSE',
'other'])]


#On crée une colonne regroupe qui va mettre dans les lignes là où a
true ou bien false la valeur TRUE/FALSE et OTHER ça laisse
dftrain['regrouped'] = dftrain['rating'].apply(lambda x:'TRUE/FALSE'
if x in ['TRUE', 'FALSE'] else 'OTHER')

print("Echantillon de mon dataset \n")
print(dftrain.sample(n=10))
print("\n")
print("Quelques informations importantes \n")
dftrain.info()

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

print("le type de X_test est" ,X_text.columns)
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("voici la dernière case")
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 sont " ,y.value_counts())
```

```
/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)

Echantillon de mon dataset

            id                                                text  \
249   350ee2a0  The National Pulse can reveal the Black Lives ...
513   a8b2aa17  President Obama has taken a great deal of effo...
```

```
159   5f0e6b6d  The evidence for massive voter fraud continues...
1241  e6907337  New Delhi, Feb 20 (IANS) After a Disinfo Lab r...
886   7d021245  The phone number to report Iowa caucus results...
683   c1343280  ✖ Just over a year from now, Doctor Strange in...
1498  dc063e58  A stimulus bill has gone through, but it was a...
672   cbe1ee65                       Wisconsin ACT scores plummet
1329  5fd2bed1  ABC's Isobel Markham reports:  With the threat...
1787  8.00E+25  A coalition of civil society groups has descri...

                                                    title rating
regrouped
249   EXCLUSIVE: Black Lives Matter Website, ‘Defund...  FALSE
TRUE/FALSE
513   "Researchers are looking into the possibility ...   TRUE
TRUE/FALSE
159   US Army seizes Dominion servers in Germany, Tr...  FALSE
TRUE/FALSE
1241  Turkey joins hands with Pakistan to spread unr...  FALSE
TRUE/FALSE
886   Clog the lines': Internet trolls deliberately ...  other
OTHER
683   Ariana Grande No Longer Has a Tattoo That Says...  FALSE
TRUE/FALSE
1498  Trump Will Use Emergency Powers To Remove Pelo...  FALSE
TRUE/FALSE
672   Coronavirus: Ireland is one island with two ve...  FALSE
TRUE/FALSE
1329  Rep. Tom Graves Defends Obamacare De-Funding E...  FALSE
TRUE/FALSE
1787  Tesco food waste rose to equivalent of 119m me...   TRUE
TRUE/FALSE


Quelques informations importantes

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1812 entries, 0 to 2527
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   id         1812 non-null   object
 1   text       1812 non-null   object
 2   title      1784 non-null   object
 3   rating     1812 non-null   object
 4   regrouped  1812 non-null   object
dtypes: object(5)
memory usage: 84.9+ KB
le type de X_test est Index(['text'], dtype='object')
le texte est
```

```
                                                   text
0     Distracted driving causes more deaths in Canad...
3     But things took a turn for the worse when riot...
4     It’s no secret that Epstein and Schiff share a...
5     Nation  UPDATED 8:23 PM - K A B O O M! Governo...
6     November 23, 2019  The U.S. Food and Drug Admi...
...                                                 ...
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...

[1812 rows x 1 columns]
```

le titre est

```
                                                  title
0     You Can Be Fined $1,500 If Your Passenger Is U...
3     Obama’s Daughters Caught on Camera Burning US ...
4     Leaked Visitor Logs Reveal Schiff’s 78 Visits ...
5     K A B O O M! Governor and Secretary of State i...
6     FDA Shocking Study: Cells Used In Vaccines Con...
...                                                 ...
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

[1812 rows x 1 columns]
```

voici la dernière case

```
0       TRUE/FALSE
3       TRUE/FALSE
4       TRUE/FALSE
5            OTHER
6       TRUE/FALSE
           ...
2523    TRUE/FALSE
2524    TRUE/FALSE
2525    TRUE/FALSE
2526    TRUE/FALSE
2527    TRUE/FALSE
Name: regrouped, Length: 1812, dtype: object

la taille de X_text est (1812, 1)
la taille de y_train est  (1812,)
les valeurs de TRUE et FALSE sont  TRUE/FALSE    1578
```

```
OTHER          234
Name: regrouped, 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 de telle sorte que le nombre de lignes de TRUE/FALSE soit = au nbr de lignes de Other. et on mélange le DataFrame.

```
#On applique du sous-échantillonnage (downsampling) : car on a plus de FALSE (578) que des TRUE (211)

# Séparer les classes en deux dataframes
df_false_true = dftrain[dftrain['regrouped'] == 'TRUE/FALSE']
df_other = dftrain [dftrain['regrouped'] == 'OTHER']

# Sous-échantillonner la classe majoritaire (FALSE) pour obtenir un nombre égal d'échantillons pour chaque classe
df_subsampled = df_false_true.sample(n=len(df_other), random_state=42)

# Concaténer les deux dataframes
dftrain = pd.concat([df_subsampled, df_other])

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


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/FALSE et OTHER maintenant sont " ,y.value_counts())
```

```
le texte est

/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)

                                                  text
0     Distracted driving causes more deaths in Canad...
```

```
3     But things took a turn for the worse when riot...
4     It’s no secret that Epstein and Schiff share a...
5     Nation  UPDATED 8:23 PM - K A B O O M! Governo...
6     November 23, 2019  The U.S. Food and Drug Admi...
...                                                 ...
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...

[1812 rows x 1 columns]
```

le titre est

```
                                                  title
0     You Can Be Fined $1,500 If Your Passenger Is U...
3     Obama’s Daughters Caught on Camera Burning US ...
4     Leaked Visitor Logs Reveal Schiff’s 78 Visits ...
5     K A B O O M! Governor and Secretary of State i...
6     FDA Shocking Study: Cells Used In Vaccines Con...
...                                                 ...
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

[1812 rows x 1 columns]
```

le y est

```
947     TRUE/FALSE
2224    TRUE/FALSE
1307    TRUE/FALSE
798          OTHER
320     TRUE/FALSE
           ...
1160    TRUE/FALSE
570          OTHER
1200         OTHER
2190         OTHER
391     TRUE/FALSE
Name: regrouped, Length: 468, dtype: object

la taille de X_text est (1812, 1)
la taille de y_train est  (468,)
les valeurs de TRUE/FALSE et OTHER maintenant sont  TRUE/FALSE    234
OTHER         234
Name: regrouped, 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
```

```
/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)
```

```
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pyLDAvis in
/usr/local/lib/python3.10/dist-packages (3.4.1)
Requirement already satisfied: numpy>=1.24.2 in
/usr/local/lib/python3.10/dist-packages (from pyLDAvis) (1.24.3)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.10/dist-packages (from pyLDAvis) (3.1.2)
Requirement already satisfied: pandas>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from pyLDAvis) (2.0.1)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.10/dist-packages (from pyLDAvis) (67.7.2)
Requirement already satisfied: scikit-learn>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from pyLDAvis) (1.2.2)
Requirement already satisfied: gensim in
/usr/local/lib/python3.10/dist-packages (from pyLDAvis) (4.3.1)
Requirement already satisfied: numexpr in
/usr/local/lib/python3.10/dist-packages (from pyLDAvis) (2.8.4)
Requirement already satisfied: funcy in
/usr/local/lib/python3.10/dist-packages (from pyLDAvis) (2.0)
Requirement already satisfied: scipy in
/usr/local/lib/python3.10/dist-packages (from pyLDAvis) (1.10.1)
Requirement already satisfied: joblib>=1.2.0 in
/usr/local/lib/python3.10/dist-packages (from pyLDAvis) (1.2.0)
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)
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: scipy>=1.7.0 in
/usr/local/lib/python3.10/dist-packages (from gensim) (1.10.1)
Requirement already satisfied: smart-open>=1.8.1 in
/usr/local/lib/python3.10/dist-packages (from gensim) (6.3.0)
Requirement already satisfied: numpy>=1.18.5 in
/usr/local/lib/python3.10/dist-packages (from gensim) (1.24.3)
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)
            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
```

```
/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)
```

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
```

```
/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)
```

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

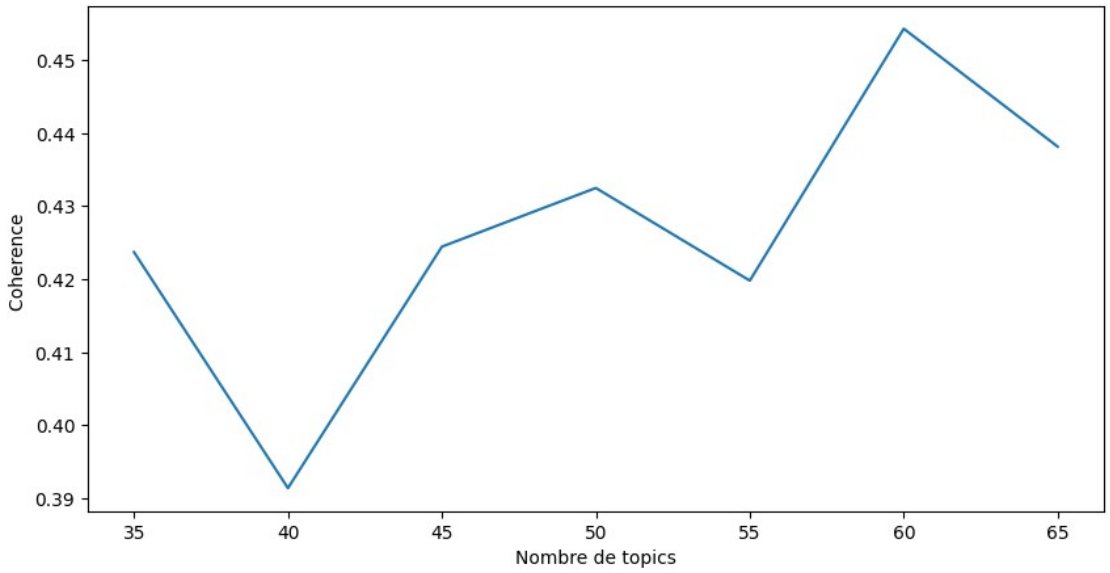
```
import pyLDAvis
import pyLDAvis.gensim_models as gensimvis

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.legend(("Valeurs de cohérencescoherence_values"), loc='best')
plt.show()
```

```
/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    a1334b14  War-torn eastern regions of Ukraine have no la...
1    2855fe5a  TIJUANA, Mexico — It’s the image from the unfo...
2    aacdc4d3  Today, Congresswoman Maxine Waters D-CA, Chair...
3    4e29fefa  Meghan Markle will use the furore over her int...
4    80b443af  Further proof that Democrats are the greatest ...
..        ...                                               ...
463  d49ee9d3  The scale of Antarctica is startling. Miles of...
464  866fa600  Coronavirus may be sexually transmitted and ca...
465  d17185d3  Like what?  Helen Harwatt is a researcher trai...
466  0fd5b80e                    Tumeric kills cancer not patient
467  ad45e0f7  WASHINGTON, DC — The Pentagon has issued an in...

                                                 title rating
regrouped  \
0    Look No Further, The Best Doctor Strange in th...  FALSE
TRUE/FALSE
1    A discussion of ‘smokers’ black lungs’ started...   TRUE
TRUE/FALSE
2    Democratic Lawmaker introduces bill to rename ...  FALSE
TRUE/FALSE
3    Newton Emerson: Swiss model offers food for th...  other
OTHER
4    Democrats Introduce Bill To ‘Euthanize Seniors...  FALSE
TRUE/FALSE
..                                                 ...    ...
...
463               Miles of Ice Collapsing Into the Sea   TRUE
TRUE/FALSE
464  Universal Credit leaves working families worse...  other
OTHER
465              If Everyone Ate Beans Instead of Beef  other
OTHER
466  Vermont state trooper revived with Narcan afte...  other
OTHER
467  Pentagon Confirms Coronavirus Accidently Got I...  FALSE
TRUE/FALSE

                                           text_title  \
0    War-torn eastern regions of Ukraine have no la...
1    TIJUANA, Mexico — It’s the image from the unfo...
2    Today, Congresswoman Maxine Waters D-CA, Chair...
3    Meghan Markle will use the furore over her int...
4    Further proof that Democrats are the greatest ...
..                                                 ...
463  The scale of Antarctica is startling. Miles of...
```

```
464  Coronavirus may be sexually transmitted and ca...
465  Like what?  Helen Harwatt is a researcher trai...
466  Tumeric kills cancer not patient Vermont state...
467  WASHINGTON, DC — The Pentagon has issued an in...

                                                keywords
0    say, health visitor, year, number, service, ca...
1    say, time, report result, know, people, poot, ...
2    say, year, climate change, climate scientist, ...
3    ice shelf, say, shelf, year, collapse, managem...
4    say, year, climate change, climate scientist, ...
..                                                 ...
463  trump, say, mask, vote, state, election, even,...
464  say, see, year, also, people, work, take, new,...
465  short seller, test, gamestop stock, hedge fund...
466  disorder include, include death, kill cancer, ...
467  short seller, test, gamestop stock, hedge fund...

[468 rows x 7 columns]
```




**60 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=60
#alpha_best=df_result['alpha'][0]
#eta_best=df_result['eta'][0]

lda_model = gensim.models.ldamulticore.LdaMulticore(
                           corpus=corpus,
                           num_topics=num_topic_best,
                           #alpha=alpha_best,
                           #eta=eta_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.text_title)


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  60  différents topics pour le corpus TF-IDF :
Topic : 0 Words : 0.005*"junior doctor" + 0.005*"think tank" +
0.005*"nonsense say" + 0.005*"work doctor" + 0.004*"call strike" +
0.003*"bow group" + 0.003*"go ahead" + 0.003*"cost" + 0.003*"good
```

training" + 0.003*"fact close" + 0.003*"operation delay" + 0.003*"left wing" + 0.003*"privileged help" + 0.003*"action month" + 0.003*"school motto" + 0.003*"life lose" + 0.003*"say moet" + 0.003*"warm clime" + 0.003*"friend profession" + 0.003*"union bigwig"
Topic : 1 Words : 0.014*"say" + 0.006*"people" + 0.005*"country" + 0.005*"mental health" + 0.004*"also" + 0.004*"case" + 0.004*"number" + 0.004*"court" + 0.004*"make" + 0.003*"year" + 0.003*"see" + 0.003*"political" + 0.003*"well" + 0.003*"first" + 0.003*"even" + 0.003*"need" + 0.003*"work" + 0.003*"get" + 0.003*"include" + 0.003*"picture"
Topic : 2 Words : 0.010*"day" + 0.009*"rate" + 0.009*"test" + 0.008*"virus" + 0.008*"fire" + 0.008*"positive test" + 0.007*"dhs" + 0.007*"use" + 0.007*"test result" + 0.006*"result" + 0.005*"covid" + 0.005*"test rate" + 0.004*"continue" + 0.004*"state" + 0.004*"make" + 0.004*"however" + 0.004*"people test" + 0.004*"number new" + 0.004*"arson" + 0.004*"question"
Topic : 3 Words : 0.008*"vote" + 0.006*"say" + 0.005*"bad" + 0.005*"warn people" + 0.005*"transmit cause" + 0.005*"coronavirus sexually" + 0.005*"stop eat" + 0.005*"treatment outcome" + 0.005*"health expert" + 0.005*"work family" + 0.005*"male infertility" + 0.005*"credit leave" + 0.005*"study universal" + 0.005*"tilapia nhs" + 0.005*"datum cancer" + 0.005*"fail collect" + 0.004*"trump" + 0.004*"election" + 0.003*"report" + 0.003*"tweet"
Topic : 4 Words : 0.006*"trump" + 0.005*"vote" + 0.005*"state" + 0.004*"election" + 0.004*"point" + 0.004*"find" + 0.003*"child" + 0.003*"scientist" + 0.003*"even" + 0.003*"make" + 0.003*"use" + 0.003*"ice" + 0.003*"know" + 0.003*"win" + 0.003*"well" + 0.003*"year" + 0.003*"number" + 0.003*"people" + 0.002*"recount" + 0.002*"vaccine"
Topic : 5 Words : 0.011*"say" + 0.006*"ice loss" + 0.005*"year" + 0.004*"see" + 0.004*"solar minimum" + 0.003*"year old" + 0.003*"think" + 0.003*"new wave" + 0.003*"huge annual" + 0.003*"even bad" + 0.003*"term trend" + 0.003*"crash" + 0.002*"sun" + 0.002*"climate change" + 0.002*"rate" + 0.002*"social security" + 0.002*"find" + 0.002*"community large" + 0.002*"equal break" + 0.002*"mountain death"
Topic : 6 Words : 0.008*"oa cancel" + 0.004*"year" + 0.004*"make" + 0.004*"narrowly fail" + 0.004*"video" + 0.004*"level thinly" + 0.004*"election last" + 0.004*"voter reject" + 0.004*"hate speech" + 0.004*"year ago" + 0.004*"first female" + 0.004*"trump urge" + 0.004*"president succeed" + 0.004*"veil attack" + 0.004*"year early" + 0.004*"hap basement" + 0.004*"first time" + 0.004*"include cancellation" + 0.004*"cap oa" + 0.004*"day season"
Topic : 7 Words : 0.017*"say" + 0.009*"big business" + 0.009*"scandal picture" + 0.005*"business scandal" + 0.003*"getty big" + 0.003*"know" + 0.003*"cultural fit" + 0.003*"minimum wage" + 0.003*"year" + 0.002*"people" + 0.002*"man" + 0.002*"short term" + 0.002*"bribery fraud" + 0.002*"campaign trail" + 0.002*"climate change" + 0.002*"new" + 0.001*"valid email" + 0.001*"address enter" + 0.001*"message verifyerror" + 0.001*"chemtrail"
Topic : 8 Words : 0.007*"covid" + 0.006*"death" + 0.005*"violence" + 0.005*"city" + 0.005*"black life" + 0.004*"say" + 0.004*"matter" +

0.004*"president" + 0.004*"people die" + 0.004*"happen" +
0.004*"believe" + 0.004*"cause" + 0.004*"burn" + 0.003*"outright" +
0.002*"trump" + 0.002*"see" + 0.002*"infection" + 0.002*"live" +
0.002*"worldwide" + 0.002*"victim virus"
Topic : 9 Words : 0.013*"say" + 0.008*"small business" +
0.008*"government" + 0.007*"state" + 0.006*"business" + 0.005*"job" +
0.005*"people" + 0.005*"get" + 0.004*"community" + 0.004*"call" +
0.004*"take" + 0.004*"also" + 0.004*"picture" + 0.003*"year" +
0.003*"include" + 0.003*"number" + 0.003*"new" + 0.003*"need" +
0.003*"target" + 0.003*"go"
Topic : 10 Words : 0.011*"say" + 0.005*"nhs" + 0.004*"year" +
0.003*"state" + 0.003*"also" + 0.003*"work" + 0.003*"people" +
0.003*"last week" + 0.003*"carbon dioxide" + 0.003*"hear say" +
0.003*"leave campaign" + 0.002*"school" + 0.002*"claim" + 0.002*"last
year" + 0.002*"government" + 0.002*"time" + 0.002*"report" +
0.002*"climate change" + 0.002*"cause" + 0.002*"year old"
Topic : 11 Words : 0.007*"firearm" + 0.004*"bill" + 0.004*"anger
conservative" + 0.004*"tell truth" + 0.004*"say" + 0.003*"liberal
tell" + 0.003*"lie anger" + 0.003*"also" + 0.003*"year" +
0.003*"issue" + 0.003*"senator" + 0.003*"student" + 0.003*"vote" +
0.003*"organization" + 0.003*"name" + 0.003*"anger liberal" +
0.003*"floor" + 0.003*"event" + 0.003*"resemble" + 0.002*"give"
Topic : 12 Words : 0.007*"climate change" + 0.004*"say" +
0.003*"immune system" + 0.003*"mile hour" + 0.003*"primary control" +
0.003*"poll worker" + 0.003*"atmospheric co" + 0.003*"much less" +
0.003*"work" + 0.003*"show" + 0.003*"new regulation" + 0.003*"allow
speed" + 0.002*"entire immune" + 0.002*"call" + 0.002*"blood cell" +
0.002*"yet" + 0.002*"hundred thousand" + 0.002*"rate" +
0.002*"governor motorcycle" + 0.002*"vote"
Topic : 13 Words : 0.018*"mental health" + 0.007*"service" +
0.007*"people" + 0.007*"vaccine" + 0.006*"say" + 0.005*"also" +
0.004*"health" + 0.004*"health care" + 0.004*"hpv" + 0.004*"year" +
0.003*"measle" + 0.003*"increase" + 0.003*"country" + 0.003*"cervical
cancer" + 0.003*"nhs" + 0.003*"lead" + 0.003*"field" + 0.003*"include"
+ 0.003*"child young" + 0.003*"make"
Topic : 14 Words : 0.010*"mask" + 0.006*"say" + 0.004*"use" +
0.004*"study" + 0.004*"white people" + 0.004*"new" + 0.004*"help buy"
+ 0.003*"work" + 0.003*"get" + 0.003*"address" + 0.003*"business
recruitment" + 0.003*"center city" + 0.003*"ballot" + 0.003*"absentee
ballot" + 0.003*"include" + 0.003*"information" + 0.003*"witness
address" + 0.003*"election" + 0.003*"allow" + 0.002*"witness"
Topic : 15 Words : 0.013*"say" + 0.006*"time" + 0.005*"report result"
+ 0.004*"year" + 0.004*"take" + 0.003*"help" + 0.003*"get call" +
0.003*"phone number" + 0.003*"include supporter" + 0.003*"call come" +
0.003*"line night" + 0.003*"official say" + 0.003*"prank call" +
0.003*"phone line" + 0.003*"reason result" + 0.003*"app work" +
0.003*"people" + 0.002*"report say" + 0.002*"child" + 0.002*"call"
Topic : 16 Words : 0.008*"say" + 0.006*"people" + 0.003*"year" +
0.003*"novel coronavirus" + 0.003*"lab handle" + 0.003*"drug third" +
0.003*"exposure" + 0.002*"report" + 0.002*"sar cov" + 0.002*"age" +

0.002*"human" + 0.002*"study coronavirus" + 0.002*"sample wonder" +
0.002*"advanced virus" + 0.002*"disease escape" + 0.002*"protect
people" + 0.002*"image" + 0.002*"enlarge image" + 0.002*"seafood
market" + 0.002*"increase risk"
Topic : 17 Words : 0.007*"covid" + 0.006*"say" + 0.006*"death" +
0.004*"covid death" + 0.003*"go" + 0.003*"year" + 0.003*"get" +
0.003*"certificate" + 0.003*"care home" + 0.003*"call" + 0.003*"nhs
bed" + 0.003*"covid go" + 0.003*"case" + 0.003*"even" + 0.002*"mean" +
0.002*"number" + 0.002*"virus" + 0.002*"percent" + 0.002*"hospital" +
0.002*"make"
Topic : 18 Words : 0.016*"say" + 0.006*"take" + 0.005*"child" +
0.003*"see" + 0.003*"day" + 0.003*"image" + 0.003*"invalid email" +
0.003*"day survey" + 0.003*"health problem" + 0.003*"access child" +
0.003*"know experience" + 0.003*"camp border" + 0.003*"share refugee"
+ 0.003*"take refugee" + 0.003*"country shoulder" + 0.003*"take
syrian" + 0.003*"moral obligation" + 0.002*"people" + 0.002*"make" +
0.002*"work"
Topic : 19 Words : 0.012*"kill cancer" + 0.007*"stop" + 0.006*"poop
paint" + 0.006*"arrest suspect" + 0.006*"matter leader" + 0.006*"black
life" + 0.006*"admit really" + 0.006*"cell vandalize" +
0.006*"government finally" + 0.005*"bill" + 0.005*"add" + 0.003*"read"
+ 0.003*"video" + 0.003*"simple" + 0.003*"pledge allegiance" +
0.003*"aid package" + 0.003*"curb coronavirus" + 0.003*"mouthwash
help" + 0.003*"truth real" + 0.003*"take away"
Topic : 20 Words : 0.008*"year" + 0.006*"say" + 0.005*"hot year" +
0.005*"marry adopt" + 0.005*"surface temperature" + 0.004*"child" +
0.003*"also" + 0.003*"last year" + 0.003*"average surface" +
0.003*"weather pattern" + 0.003*"temperature" + 0.003*"record" +
0.003*"see" + 0.003*"ocean temperature" + 0.002*"right" + 0.002*"year
old" + 0.002*"share" + 0.002*"temperature increase" + 0.002*"onewire"
+ 0.002*"article"
Topic : 21 Words : 0.006*"say" + 0.006*"public school" +
0.004*"election" + 0.004*"poll" + 0.003*"fire" + 0.003*"counsel say" +
0.003*"principal sinclair" + 0.003*"back" + 0.003*"brexit" +
0.003*"tory" + 0.003*"party" + 0.002*"snow people" + 0.002*"make
ornament" + 0.002*"candy cane" + 0.002*"divide national" +
0.002*"national flag" + 0.002*"national civic" + 0.002*"flag
religious" + 0.002*"religious people" + 0.002*"government"
Topic : 22 Words : 0.007*"hodge" + 0.007*"also" + 0.006*"admit" +
0.005*"laugh" + 0.004*"kill" + 0.004*"claim" + 0.004*"die" +
0.004*"assassination" + 0.003*"give" + 0.003*"sleep" +
0.003*"ultimately" + 0.003*"many" + 0.003*"order" + 0.003*"include" +
0.003*"take" + 0.003*"woman" + 0.003*"organization" + 0.003*"security"
+ 0.003*"carry assassination" + 0.003*"hodge say"
Topic : 23 Words : 0.009*"mail ballot" + 0.008*"say" + 0.004*"bump
stock" + 0.004*"make" + 0.004*"governor" + 0.004*"member" +
0.003*"sign replace" + 0.003*"election" + 0.003*"also" +
0.003*"ballot" + 0.003*"stoke central" + 0.003*"nigel farage" +
0.003*"request mail" + 0.002*"chinese" + 0.002*"list" + 0.002*"sign" +
0.002*"time" + 0.002*"give" + 0.002*"never miss" + 0.002*"eighth

circuit"
Topic : 24 Words : 0.004*"noaa chart" + 0.004*"local temperature" + 0.004*"breaking cold" + 0.004*"adjust upwards" + 0.004*"recent temperature" + 0.003*"steal election" + 0.003*"swing state" + 0.003*"alter vote" + 0.002*"voting machine" + 0.002*"drain swamp" + 0.002*"tweet" + 0.002*"say" + 0.002*"temperature record" + 0.002*"vote" + 0.002*"know" + 0.002*"year" + 0.002*"coronavirus lockdown" + 0.002*"report" + 0.002*"atmospheric administration" + 0.002*"national oceanic"
Topic : 25 Words : 0.012*"say" + 0.005*"net migration" + 0.004*"officer" + 0.004*"teacher shortage" + 0.004*"work permit" + 0.004*"valid email" + 0.004*"address enter" + 0.003*"communication" + 0.003*"use" + 0.003*"message verifyerror" + 0.003*"police" + 0.003*"capture" + 0.003*"non specialist" + 0.003*"train subject" + 0.003*"staff train" + 0.003*"cop" + 0.003*"migration watch" + 0.003*"watch advocate" + 0.003*"say migration" + 0.003*"high skilled"
Topic : 26 Words : 0.011*"life matter" + 0.009*"say" + 0.007*"associate professor" + 0.005*"boise police" + 0.005*"activist boise" + 0.005*"find possession" + 0.005*"boise black" + 0.005*"contemporary policing" + 0.002*"report" + 0.002*"black life" + 0.002*"vote" + 0.002*"research" + 0.002*"department" + 0.002*"approach arrest" + 0.002*"track wilson" + 0.002*"police perpetuate" + 0.002*"demonstration council" + 0.002*"blackness guise" + 0.002*"support position" + 0.002*"support black"
Topic : 27 Words : 0.011*"say" + 0.004*"study" + 0.004*"poverty" + 0.003*"vitamin" + 0.003*"climate change" + 0.002*"year old" + 0.002*"coronavirus" + 0.002*"people" + 0.002*"year" + 0.002*"risk" + 0.002*"high risk" + 0.002*"come" + 0.002*"virus" + 0.002*"go" + 0.002*"make" + 0.002*"public health" + 0.002*"ballot" + 0.002*"flu vaccine" + 0.002*"see" + 0.002*"flu shoot"
Topic : 28 Words : 0.015*"self employ" + 0.007*"say" + 0.007*"national insurance" + 0.005*"police" + 0.005*"protestor" + 0.004*"employ people" + 0.003*"say self" + 0.003*"earn less" + 0.003*"state benefit" + 0.003*"budget banner" + 0.003*"people" + 0.003*"also" + 0.003*"get" + 0.003*"trump" + 0.003*"fire" + 0.003*"supply" + 0.003*"prepare" + 0.003*"officer" + 0.003*"former sheriff" + 0.003*"arpaio"
Topic : 29 Words : 0.008*"product" + 0.005*"change" + 0.005*"remove" + 0.005*"recall" + 0.005*"say" + 0.004*"use" + 0.004*"trump" + 0.004*"restaurant" + 0.004*"state department" + 0.004*"chain" + 0.004*"beef" + 0.004*"multiple time" + 0.004*"usda" + 0.003*"page" + 0.003*"lead" + 0.003*"affect" + 0.003*"employee" + 0.003*"biography" + 0.003*"vice" + 0.003*"term end"
Topic : 30 Words : 0.010*"test" + 0.007*"virus" + 0.006*"pcr test" + 0.005*"even" + 0.005*"say" + 0.004*"use" + 0.004*"https pubme" + 0.004*"sar cov" + 0.004*"example" + 0.004*"patient" + 0.004*"health visitor" + 0.003*"case" + 0.003*"pcr" + 0.003*"gold standard" + 0.003*"also" + 0.003*"coronavirus" + 0.003*"positive" + 0.003*"dose vitamin" + 0.003*"paper" + 0.002*"charité"
Topic : 31 Words : 0.009*"vaccine" + 0.008*"citizen" + 0.006*"government" + 0.005*"pay" + 0.005*"note" + 0.005*"state" +

0.005*"know" + 0.005*"birth certificate" + 0.005*"status" +
0.005*"legal" + 0.005*"also" + 0.005*"drug" + 0.005*"bond" +
0.005*"note company" + 0.004*"country" + 0.004*"system" +
0.004*"commodity" + 0.004*"contract" + 0.004*"take" + 0.004*"federal"
Topic : 32 Words : 0.008*"say" + 0.007*"go" + 0.006*"time" +
0.005*"climate change" + 0.004*"judge" + 0.004*"know" + 0.004*"tell" +
0.004*"think" + 0.004*"people" + 0.004*"wackos" + 0.004*"cause" +
0.003*"change" + 0.003*"get" + 0.003*"fossil fuel" + 0.003*"state" +
0.003*"stand" + 0.003*"even" + 0.003*"well" + 0.003*"want" +
0.003*"take"
Topic : 33 Words : 0.010*"vaccine" + 0.006*"say" + 0.005*"study" +
0.004*"people" + 0.004*"make" + 0.003*"year" + 0.003*"cervical cancer"
+ 0.003*"know" + 0.003*"tell" + 0.003*"year old" + 0.003*"give" +
0.003*"go" + 0.002*"many" + 0.002*"child" + 0.002*"death" +
0.002*"bedroom tax" + 0.002*"report" + 0.002*"food stamp" +
0.002*"autoimmune disease" + 0.002*"poot"
Topic : 34 Words : 0.011*"say" + 0.006*"report" + 0.005*"year" +
0.005*"show" + 0.004*"also" + 0.004*"result" + 0.004*"trump" +
0.003*"see" + 0.003*"take" + 0.003*"use" + 0.003*"child" +
0.003*"people" + 0.003*"know" + 0.003*"death" + 0.003*"add" +
0.003*"die" + 0.003*"vote" + 0.003*"lead" + 0.002*"call" +
0.002*"candidate"
Topic : 35 Words : 0.009*"virus" + 0.006*"insertion" +
0.005*"sequence" + 0.005*"ncov" + 0.004*"think" + 0.004*"gag" +
0.004*"amino acid" + 0.004*"point" + 0.004*"genome" + 0.003*"find" +
0.003*"align" + 0.003*"gene" + 0.003*"use" + 0.003*"sander" +
0.003*"say" + 0.003*"coronavirus" + 0.003*"gun control" +
0.003*"claim" + 0.003*"host cell" + 0.003*"table"
Topic : 36 Words : 0.005*"use cocaine" + 0.005*"glide vehicle" +
0.005*"control room" + 0.005*"missile defense" + 0.004*"present say" +
0.003*"century average" + 0.003*"people" + 0.003*"trump legal" +
0.003*"last year" + 0.003*"leader say" + 0.003*"day" + 0.003*"new" +
0.003*"trump supporter" + 0.003*"use" + 0.002*"find" +
0.002*"birthday" + 0.002*"even" + 0.002*"young people" + 0.002*"vote
counting" + 0.002*"count poll"
Topic : 37 Words : 0.006*"say" + 0.005*"new runway" + 0.005*"year old"
+ 0.004*"cannabis" + 0.004*"think" + 0.004*"cause substantial" +
0.004*"question advance" + 0.004*"give debate" + 0.004*"cost day" +
0.003*"even" + 0.002*"deal" + 0.002*"new" + 0.002*"use" +
0.002*"report" + 0.002*"arrest" + 0.002*"show" + 0.002*"sell" +
0.002*"caution" + 0.002*"go" + 0.002*"look"
Topic : 38 Words : 0.007*"say" + 0.006*"police" + 0.005*"people" +
0.004*"report" + 0.004*"state" + 0.004*"datum" + 0.004*"ask secretary"
+ 0.004*"commonwealth affair" + 0.004*"state foreign" + 0.003*"cent" +
0.003*"officer" + 0.003*"side effect" + 0.003*"global warming" +
0.003*"street" + 0.003*"record high" + 0.003*"occur prior" +
0.003*"temperature set" + 0.003*"boko haram" + 0.003*"survey" +
0.003*"allegation"
Topic : 39 Words : 0.000*"people" + 0.000*"say" + 0.000*"time" +
0.000*"also" + 0.000*"year" + 0.000*"case" + 0.000*"find" + 0.000*"go"

+ 0.000*"virus" + 0.000*"report" + 0.000*"make" + 0.000*"use" +
0.000*"know" + 0.000*"work" + 0.000*"country" + 0.000*"cell" +
0.000*"get" + 0.000*"well" + 0.000*"see" + 0.000*"study"
Topic : 40 Words : 0.009*"say" + 0.004*"make" + 0.003*"state" +
0.003*"country" + 0.003*"public health" + 0.003*"boko haram" +
0.003*"day" + 0.003*"go" + 0.003*"covid patient" + 0.003*"president" +
0.003*"serve former" + 0.003*"situation country" + 0.002*"show" +
0.002*"last week" + 0.002*"homeless" + 0.002*"find" + 0.002*"people" +
0.002*"use" + 0.002*"know" + 0.002*"help"
Topic : 41 Words : 0.009*"say" + 0.007*"mask" + 0.004*"contribution
emergency" + 0.004*"british budget" + 0.003*"work" + 0.003*"wear mask"
+ 0.003*"level" + 0.003*"fine" + 0.003*"mask respirator" +
0.003*"satisfaction" + 0.002*"country include" + 0.002*"people" +
0.002*"public" + 0.002*"time" + 0.002*"report" + 0.002*"long" +
0.002*"survey" + 0.002*"low level" + 0.002*"laboratory confirm" +
0.002*"win"
Topic : 42 Words : 0.009*"unvaccinated child" + 0.006*"pay student" +
0.006*"loan burden" + 0.006*"federal student" + 0.005*"say" +
0.004*"outcome vaccinated" + 0.004*"phillip" + 0.004*"reporter ask" +
0.004*"plainly state" + 0.004*"debt borrower" + 0.004*"report
contrary" + 0.004*"reality far" + 0.003*"study" + 0.003*"child" +
0.003*"vaccine" + 0.003*"issue" + 0.003*"year age" + 0.003*"study
clarify" + 0.003*"child receive" + 0.003*"health outcome"
Topic : 43 Words : 0.009*"say" + 0.008*"mayor" + 0.006*"increase" +
0.006*"mendoza" + 0.005*"first" + 0.004*"plan" + 0.004*"sticker" +
0.003*"change" + 0.003*"elector cast" + 0.003*"elector" + 0.003*"legal
challenge" + 0.003*"meet today" + 0.003*"president vice" +
0.003*"procedural vote" + 0.003*"new" + 0.003*"mile km" + 0.003*"long
increase" + 0.003*"glacier go" + 0.003*"emanuel" + 0.003*"sticker fee"
Topic : 44 Words : 0.009*"say" + 0.006*"child sacrifice" + 0.005*"jet
stream" + 0.004*"email protect" + 0.003*"human hunting" +
0.003*"extreme weather" + 0.003*"people" + 0.002*"study link" +
0.002*"global elite" + 0.002*"circle satanic" + 0.002*"year old" +
0.002*"weather pattern" + 0.002*"ice" + 0.002*"extremist" +
0.002*"programme" + 0.002*"referral" + 0.002*"right" +
0.002*"terrorism" + 0.002*"climate scientist" + 0.002*"year"
Topic : 45 Words : 0.005*"people" + 0.005*"say" + 0.004*"key activate"
+ 0.004*"bat soup" + 0.004*"press escape" + 0.004*"close button" +
0.004*"modal window" + 0.004*"modal close" + 0.004*"islamist" +
0.003*"leap" + 0.003*"dictator" + 0.002*"country" + 0.002*"analysis" +
0.002*"center" + 0.002*"infectious disease" + 0.002*"virus" +
0.002*"rule" + 0.002*"regime" + 0.002*"ponder" + 0.002*"islamic"
Topic : 46 Words : 0.007*"say" + 0.004*"year" + 0.004*"climate change"
+ 0.004*"climate scientist" + 0.003*"romney" + 0.003*"smart
guillotine" + 0.003*"sea level" + 0.003*"state" + 0.002*"last year" +
0.002*"go" + 0.002*"new" + 0.002*"warm water" + 0.002*"time" +
0.002*"change" + 0.002*"use" + 0.002*"see" + 0.002*"make" +
0.002*"report" + 0.002*"ocean temperature" + 0.002*"election result"
Topic : 47 Words : 0.013*"say" + 0.005*"child" + 0.004*"type diabetes"
+ 0.004*"work" + 0.003*"people" + 0.003*"year" + 0.003*"last year" +

0.003*"child live" + 0.003*"single people" + 0.003*"week year" +
0.003*"diagnosis diabetes" + 0.003*"diabetes type" + 0.003*"covid" +
0.003*"consultant" + 0.003*"report" + 0.002*"work pension" +
0.002*"time" + 0.002*"even" + 0.002*"incentive move" +
0.002*"coronavirus adult"
Topic : 48 Words : 0.006*"sun" + 0.005*"combustible cigarette" +
0.005*"say" + 0.005*"week" + 0.005*"sea ice" + 0.005*"ventilator" +
0.004*"climate change" + 0.004*"shaviv" + 0.004*"solar activity" +
0.004*"arctic" + 0.004*"smoke" + 0.004*"sale" + 0.004*"smoker" +
0.004*"earth" + 0.003*"cloud" + 0.003*"use" + 0.003*"see" +
0.003*"tell" + 0.003*"plan" + 0.003*"level"
Topic : 49 Words : 0.009*"say" + 0.007*"cell" + 0.005*"cancer" +
0.003*"treatment" + 0.003*"judge say" + 0.003*"abu qatada" +
0.003*"case win" + 0.003*"win right" + 0.003*"human right" +
0.003*"european court" + 0.003*"peptide" + 0.003*"morad say" +
0.003*"mutato" + 0.003*"applicant" + 0.003*"target" + 0.003*"drug" +
0.003*"management" + 0.003*"many" + 0.003*"time" + 0.002*"cancer cell"
Topic : 50 Words : 0.007*"say" + 0.007*"covid" + 0.005*"virus" +
0.005*"pathologist" + 0.004*"novel coronavirus" + 0.004*"go" +
0.004*"report" + 0.004*"patient" + 0.004*"people" + 0.004*"make" +
0.004*"coronavirus" + 0.003*"pandemic" + 0.003*"case" + 0.003*"body" +
0.003*"find" + 0.003*"work" + 0.003*"follow" + 0.003*"prove" +
0.003*"know" + 0.003*"neighborhood"
Topic : 51 Words : 0.007*"social security" + 0.007*"email address" +
0.006*"year old" + 0.005*"message verifyerror" + 0.005*"leader say" +
0.004*"patient die" + 0.004*"enter valid" + 0.003*"new" +
0.003*"school" + 0.003*"ask" + 0.003*"advertisement" + 0.003*"sheriff"
+ 0.003*"add work" + 0.003*"new update" + 0.003*"people" +
0.003*"email offer" + 0.003*"offer event" + 0.003*"like email" +
0.003*"thank sign" + 0.003*"verifyerror message"
Topic : 52 Words : 0.009*"sea level" + 0.008*"say" + 0.008*"rise" +
0.007*"ice shelf" + 0.006*"see" + 0.006*"hcq" + 0.006*"disorder
include" + 0.005*"include death" + 0.005*"know" + 0.004*"collapse" +
0.004*"time" + 0.004*"global warming" + 0.004*"shelf" +
0.004*"vaccine" + 0.004*"fauci" + 0.004*"covid vaccine" +
0.003*"climate" + 0.003*"change" + 0.003*"find" + 0.003*"region"
Topic : 53 Words : 0.006*"island" + 0.005*"visit island" +
0.005*"blackmail" + 0.005*"appear" + 0.005*"say" + 0.004*"year" +
0.004*"flight log" + 0.004*"plastic straw" + 0.004*"plastic" +
0.003*"think" + 0.003*"cotton bud" + 0.003*"number patent" +
0.003*"teacher leave" + 0.003*"use" + 0.003*"leave" + 0.003*"covid" +
0.003*"see" + 0.003*"view" + 0.002*"image" + 0.002*"state"
Topic : 54 Words : 0.009*"ballot" + 0.008*"absentee ballot" +
0.007*"rejection rate" + 0.006*"cast" + 0.005*"private school" +
0.005*"presidential election" + 0.005*"suddenly" + 0.005*"state" +
0.004*"voter" + 0.004*"election" + 0.004*"taxpayer" + 0.004*"total
reject" + 0.004*"fee pay" + 0.004*"ballot reject" + 0.004*"violate
state" + 0.004*"ballot rejection" + 0.004*"explain" + 0.004*"rate" +
0.003*"increase" + 0.003*"fee"
Topic : 55 Words : 0.007*"year" + 0.006*"say" + 0.004*"people" +

0.003*"barnardo" + 0.003*"child" + 0.003*"living standard" +
0.003*"health service" + 0.003*"term plan" + 0.003*"medal" +
0.002*"nhs long" + 0.002*"right" + 0.002*"many" + 0.002*"make" +
0.002*"also" + 0.002*"people drink" + 0.002*"previous week" +
0.002*"week ask" + 0.002*"drunk alcohol" + 0.002*"adult say" +
0.002*"minimum unit"
Topic : 56 Words : 0.012*"say" + 0.007*"mask" + 0.007*"student" +
0.006*"university" + 0.005*"high education" + 0.005*"time" +
0.004*"high" + 0.004*"face mask" + 0.004*"also" + 0.004*"graduate" +
0.004*"sector" + 0.004*"people" + 0.003*"need" + 0.003*"employer" +
0.003*"course" + 0.003*"teaching" + 0.003*"want" + 0.003*"degree" +
0.003*"include" + 0.003*"year"
Topic : 57 Words : 0.007*"say" + 0.004*"people" + 0.004*"take
turmeric" + 0.004*"blood cancer" + 0.004*"empty stomach" +
0.004*"common type" + 0.002*"year" + 0.002*"also" + 0.002*"start" +
0.002*"report" + 0.002*"accord report" + 0.002*"report say" +
0.002*"cell" + 0.002*"day" + 0.002*"use help" + 0.002*"good quality" +
0.002*"people medical" + 0.002*"expert handle" + 0.002*"explanation
recovery" + 0.002*"treat problem"
Topic : 58 Words : 0.006*"work" + 0.006*"come" + 0.005*"say" +
0.004*"year" + 0.004*"opposed wish" + 0.004*"kyari chief" +
0.004*"former minister" + 0.004*"simply disagree" + 0.004*"rumour
vicious" + 0.004*"private facility" + 0.004*"politically unacceptable"
+ 0.004*"open border" + 0.004*"heartless remain" + 0.004*"proclaim
dead" + 0.004*"foreign labour" + 0.004*"staff dead" + 0.004*"term
uncontrolle" + 0.004*"portrayal long" + 0.003*"covid complication" +
0.002*"number"
Topic : 59 Words : 0.014*"short seller" + 0.009*"gamestop stock" +
0.008*"hedge fund" + 0.006*"stock price" + 0.005*"people" +
0.004*"amount money" + 0.004*"stock market" + 0.003*"money short" +
0.003*"big hedge" + 0.003*"bet company" + 0.003*"close position" +
0.003*"cover short" + 0.003*"gamestop short" + 0.003*"hundred
thousand" + 0.002*"make" + 0.002*"use" + 0.002*"important thing" +
0.002*"seller lose" + 0.002*"price go" + 0.002*"good bad"
Cohérence  :  0.36263216952914296
Perplexité :  -9.831830288211334

| | topic_dominant | pourcentage_contrib | \ |
|---|---|---|---|
| 0 | 41 | 0.9833 | |
| 1 | 27 | 0.9981 | |
| 2 | 11 | 0.9853 | |
| 3 | 6 | 0.9920 | |
| 4 | 5 | 0.9849 | |
| .. | ... | ... | |
| 463 | 4 | 0.7730 | |
| 464 | 3 | 0.8907 | |
| 465 | 59 | 0.9924 | |
| 466 | 19 | 0.6722 | |
| 467 | 7 | 0.9827 | |

```
                                          topic_keywords  \
0    say, mask, contribution emergency, british bud...
1    say, study, poverty, vitamin, climate change, ...
2    firearm, bill, anger conservative, tell truth,...
3    oa cancel, year, make, narrowly fail, video, l...
4    say, ice loss, year, see, solar minimum, year ...
..                                                 ...
463  trump, vote, state, election, point, find, chi...
464  vote, say, bad, warn people, transmit cause, c...
465  short seller, gamestop stock, hedge fund, stoc...
466  kill cancer, stop, poop paint, arrest suspect,...
467  say, big business, scandal picture, business s...

                                            text_title
0    War-torn eastern regions of Ukraine have no la...
1    TIJUANA, Mexico – It’s the image from the unfo...
2    Today, Congresswoman Maxine Waters D-CA, Chair...
3    Meghan Markle will use the furore over her int...
4    Further proof that Democrats are the greatest ...
..                                                 ...
463  The scale of Antarctica is startling. Miles of...
464  Coronavirus may be sexually transmitted and ca...
465  Like what?  Helen Harwatt is a researcher trai...
466  Tumeric kills cancer not patient Vermont state...
467  WASHINGTON, DC – The Pentagon has issued an in...

[468 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[:,5:7]],
axis=1).reset_index(drop=True)
#X = dftrain.keywords
y=dftrain.regrouped

# 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    a1334b14  War-torn eastern regions of Ukraine have no la...
1    2855fe5a  TIJUANA, Mexico — It’s the image from the unfo...
2    aacdc4d3  Today, Congresswoman Maxine Waters D-CA, Chair...
3    4e29fefa  Meghan Markle will use the furore over her int...
4    80b443af  Further proof that Democrats are the greatest ...
..        ...                                               ...
463  d49ee9d3  The scale of Antarctica is startling. Miles of...
464  866fa600  Coronavirus may be sexually transmitted and ca...
465  d17185d3  Like what?  Helen Harwatt is a researcher trai...
466  0fd5b80e                    Tumeric kills cancer not patient
467  ad45e0f7  WASHINGTON, DC — The Pentagon has issued an in...

                                                 title rating
regrouped  \
0    Look No Further, The Best Doctor Strange in th...  FALSE
TRUE/FALSE
1    A discussion of ‘smokers’ black lungs’ started...   TRUE
TRUE/FALSE
2    Democratic Lawmaker introduces bill to rename ...  FALSE
TRUE/FALSE
3    Newton Emerson: Swiss model offers food for th...  other
OTHER
4    Democrats Introduce Bill To ‘Euthanize Seniors...  FALSE
TRUE/FALSE
..                                                 ...    ...
...
463               Miles of Ice Collapsing Into the Sea   TRUE
TRUE/FALSE
464  Universal Credit leaves working families worse...  other
OTHER
465              If Everyone Ate Beans Instead of Beef  other
OTHER
466  Vermont state trooper revived with Narcan afte...  other
OTHER
467  Pentagon Confirms Coronavirus Accidently Got I...  FALSE
TRUE/FALSE

                                           text_title  \
0    War-torn eastern regions of Ukraine have no la...
1    TIJUANA, Mexico — It’s the image from the unfo...
2    Today, Congresswoman Maxine Waters D-CA, Chair...
3    Meghan Markle will use the furore over her int...
```

```
4    Further proof that Democrats are the greatest ...
..                                                 ...
463  The scale of Antarctica is startling. Miles of...
464  Coronavirus may be sexually transmitted and ca...
465  Like what?  Helen Harwatt is a researcher trai...
466  Tumeric kills cancer not patient Vermont state...
467  WASHINGTON, DC — The Pentagon has issued an in...

                                              keywords
0    say, mask, contribution emergency, british bud...
1    say, study, poverty, vitamin, climate change, ...
2    firearm, bill, anger conservative, tell truth,...
3    oa cancel, year, make, narrowly fail, video, l...
4    say, ice loss, year, see, solar minimum, year ...
..                                                 ...
463  trump, vote, state, election, point, find, chi...
464  vote, say, bad, warn people, transmit cause, c...
465  short seller, gamestop stock, hedge fund, stoc...
466  kill cancer, stop, poop paint, arrest suspect,...
467  say, big business, scandal picture, business s...

[468 rows x 7 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))
    #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_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())])
```

```
Exception ignored on calling ctypes callback function: <function
ThreadpoolController._find_libraries_with_dl_iterate_phdr.<locals>.mat
ch_library_callback at 0x7f6c62ab8a60>
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', 0.89850640113798, 0.03284970582743621), ('RF',
0.8849217638691323, 0.06284442861553607), ('LogisticRegression',
0.8500711237553343, 0.0427071944931153), ('MultinomialNB',
0.8130867709815078, 0.04858726047868619), ('CART', 0.7970128022759602,
0.04048834952516968), ('KNN', 0.6147937411095307,
0.09652161835036506)]
```

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_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': [0.001, 0.01, 0.1, 1,2,5,7,10]},
             {'gamma': [0.001, 0.01, 0.1,0.2,0.3,0.5,0.7,1]},
             {'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 18 candidates, totalling 90 fits
meilleur score 0.885

meilleur estimateur SVC(gamma=0.1, random_state=42)

Accuracy : 0.936
Classification Report
              precision    recall  f1-score   support

       OTHER    1.00000   0.88889   0.94118        54
  TRUE/FALSE    0.86957   1.00000   0.93023        40

    accuracy                        0.93617        94
   macro avg    0.93478   0.94444   0.93570        94
weighted avg    0.94450   0.93617   0.93652        94

Ensemble des meilleurs paramètres :
     C: 1.0
     gamma: 0.1
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.885

meilleur estimateur SVC(gamma=0.1, random_state=42)

Accuracy : 0.936
Classification Report
              precision    recall  f1-score   support

       OTHER    1.00000   0.88889   0.94118        54
  TRUE/FALSE    0.86957   1.00000   0.93023        40

    accuracy                        0.93617        94
   macro avg    0.93478   0.94444   0.93570        94
weighted avg    0.94450   0.93617   0.93652        94

Ensemble des meilleurs paramètres :
     C: 1.0
     gamma: 0.1
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.885
```

```
meilleur estimateur SVC(gamma=0.1, random_state=42)

Accuracy : 0.936
Classification Report
              precision    recall  f1-score   support

       OTHER    1.00000   0.88889   0.94118        54
  TRUE/FALSE    0.86957   1.00000   0.93023        40

    accuracy                        0.93617        94
   macro avg    0.93478   0.94444   0.93570        94
weighted avg    0.94450   0.93617   0.93652        94

Ensemble des meilleurs paramètres :
     C: 1.0
     gamma: 0.1
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.885

meilleur estimateur SVC(gamma=0.1, random_state=42)

Accuracy : 0.936
Classification Report
              precision    recall  f1-score   support

       OTHER    1.00000   0.88889   0.94118        54
  TRUE/FALSE    0.86957   1.00000   0.93023        40

    accuracy                        0.93617        94
   macro avg    0.93478   0.94444   0.93570        94
weighted avg    0.94450   0.93617   0.93652        94

Ensemble des meilleurs paramètres :
     C: 1.0
     gamma: 0.1
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.869

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

Accuracy : 0.904
Classification Report
              precision    recall  f1-score   support

       OTHER    0.94118   0.88889   0.91429        54
```

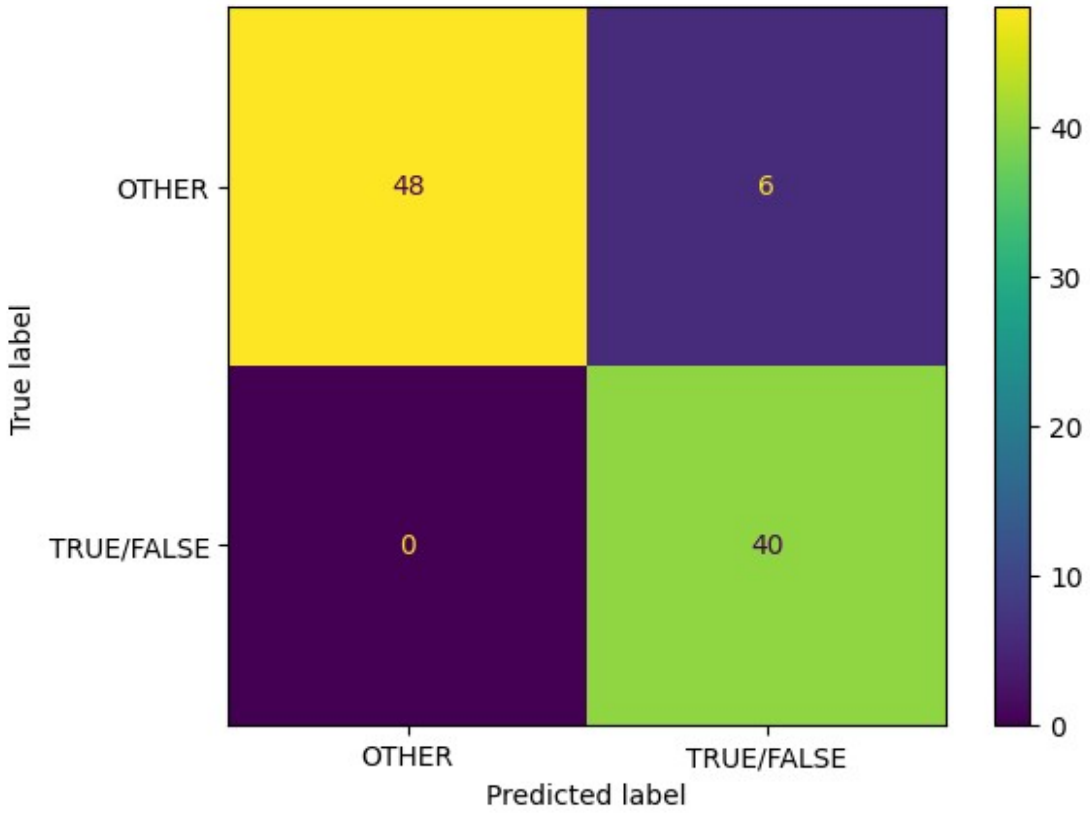
```
  TRUE/FALSE    0.86047   0.92500   0.89157        40

    accuracy                        0.90426        94
   macro avg    0.90082   0.90694   0.90293        94
weighted avg    0.90683   0.90426   0.90462        94

Ensemble des meilleurs paramètres :
     C: 1
     gamma: 'scale'
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.872

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

Accuracy : 0.926
Classification Report
              precision    recall  f1-score   support

       OTHER    0.97959   0.88889   0.93204        54
  TRUE/FALSE    0.86667   0.97500   0.91765        40

    accuracy                        0.92553        94
   macro avg    0.92313   0.93194   0.92484        94
weighted avg    0.93154   0.92553   0.92591        94

Ensemble des meilleurs paramètres :
     C: 1
     gamma: 'scale'
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.861

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

Accuracy : 0.926
Classification Report
              precision    recall  f1-score   support

       OTHER    0.97959   0.88889   0.93204        54
  TRUE/FALSE    0.86667   0.97500   0.91765        40

    accuracy                        0.92553        94
   macro avg    0.92313   0.93194   0.92484        94
weighted avg    0.93154   0.92553   0.92591        94

Ensemble des meilleurs paramètres :
```

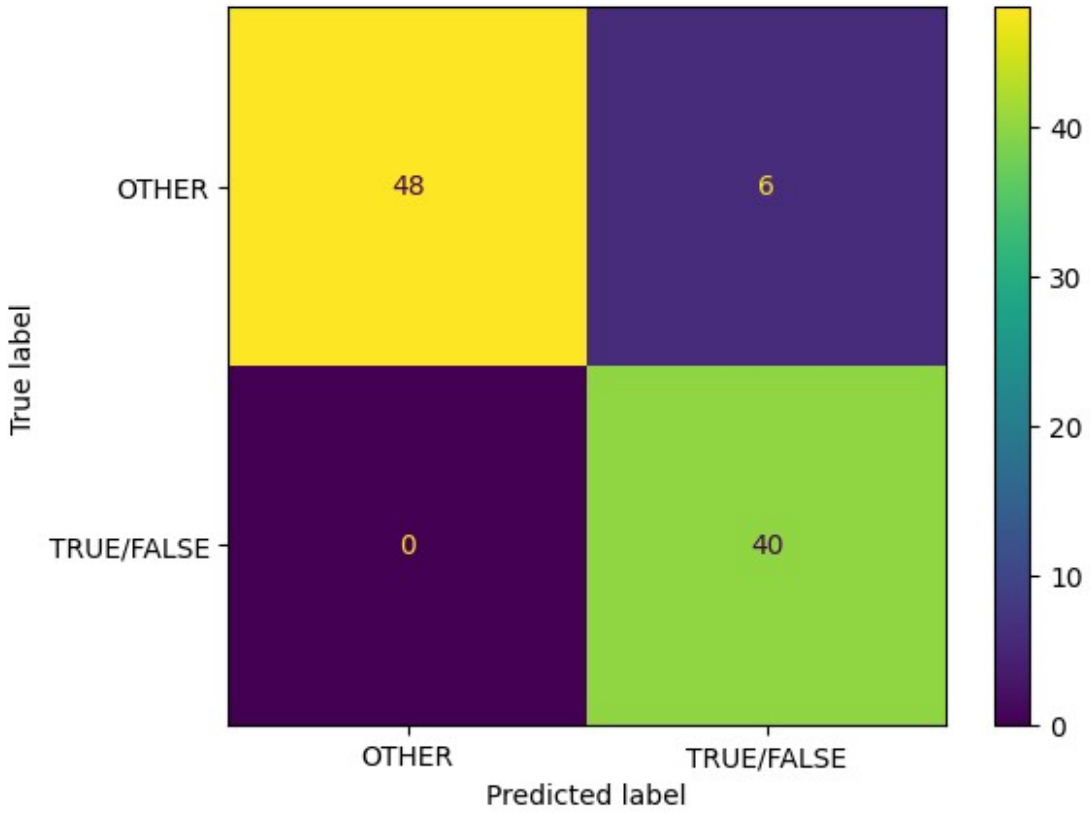
```
     C: 1
     gamma: 'scale'
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.872

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

Accuracy : 0.915
Classification Report
              precision    recall  f1-score   support

       OTHER    0.96000   0.88889   0.92308        54
  TRUE/FALSE    0.86364   0.95000   0.90476        40

    accuracy                        0.91489        94
   macro avg    0.91182   0.91944   0.91392        94
weighted avg    0.91899   0.91489   0.91528        94

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 0.864

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

Accuracy : 0.936
Classification Report
              precision    recall  f1-score   support

       OTHER    1.00000   0.88889   0.94118        54
  TRUE/FALSE    0.86957   1.00000   0.93023        40

    accuracy                        0.93617        94
   macro avg    0.93478   0.94444   0.93570        94
weighted avg    0.94450   0.93617   0.93652        94

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 0.851
```

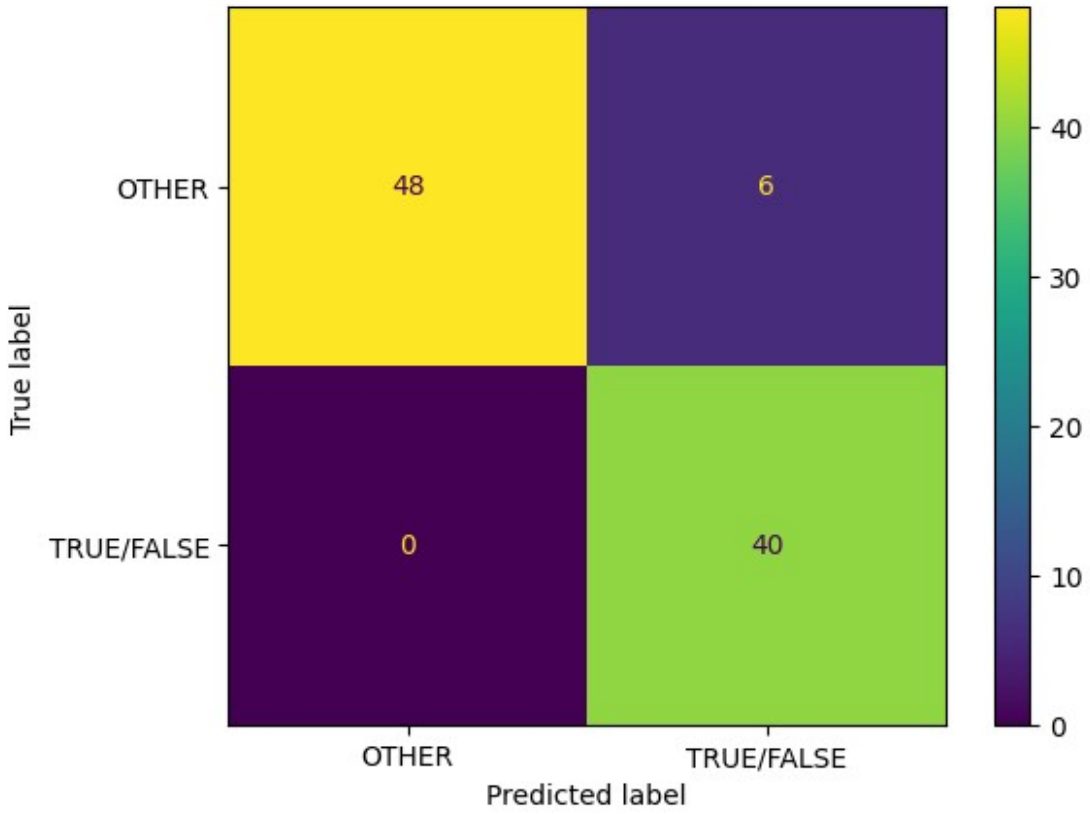
```
meilleur estimateur RandomForestClassifier(random_state=42)

Accuracy : 0.915
Classification Report
              precision    recall  f1-score   support

       OTHER    0.96000   0.88889   0.92308        54
  TRUE/FALSE    0.86364   0.95000   0.90476        40

    accuracy                        0.91489        94
   macro avg    0.91182   0.91944   0.91392        94
weighted avg    0.91899   0.91489   0.91528        94

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 0.845

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

Accuracy : 0.936
Classification Report
              precision    recall  f1-score   support

       OTHER    1.00000   0.88889   0.94118        54
  TRUE/FALSE    0.86957   1.00000   0.93023        40

    accuracy                        0.93617        94
   macro avg    0.93478   0.94444   0.93570        94
weighted avg    0.94450   0.93617   0.93652        94

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 0.840

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

Accuracy : 0.947
Classification Report
              precision    recall  f1-score   support

       OTHER    0.98039   0.92593   0.95238        54
```

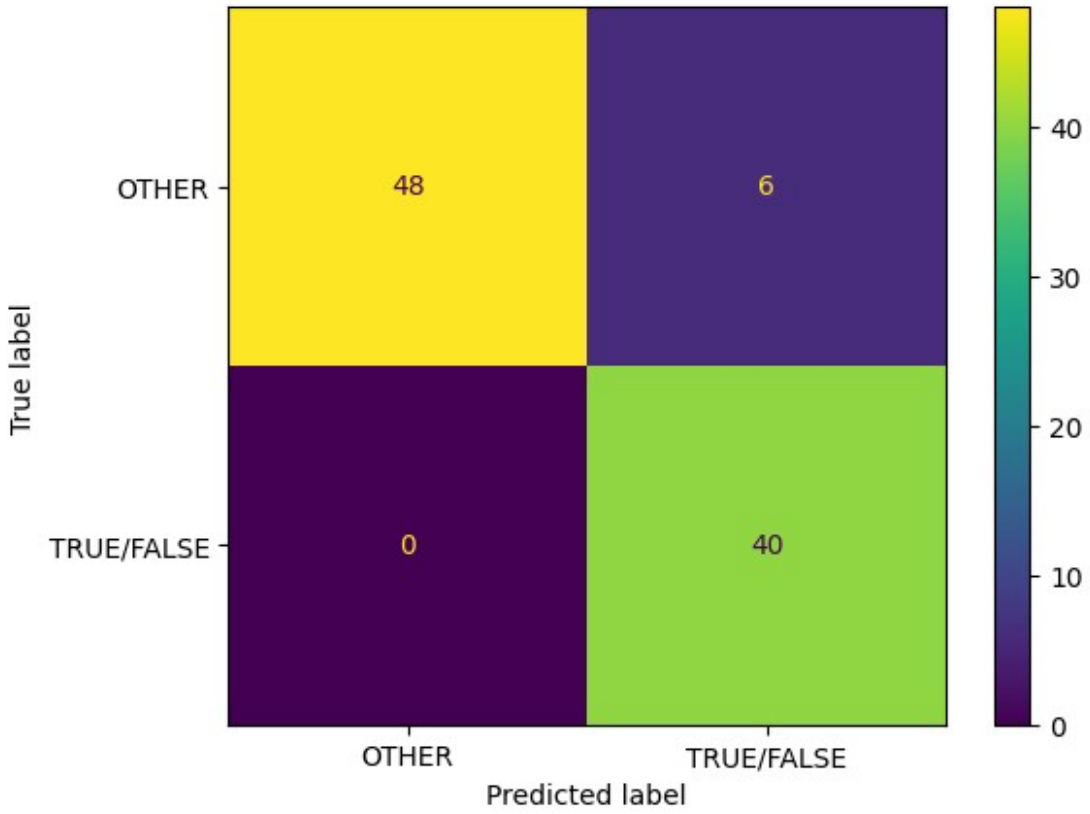
```
  TRUE/FALSE    0.90698   0.97500   0.93976        40

    accuracy                        0.94681        94
   macro avg    0.94368   0.95046   0.94607        94
weighted avg    0.94915   0.94681   0.94701        94

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 0.869

meilleur estimateur RandomForestClassifier(random_state=42)

Accuracy : 0.926
Classification Report
              precision    recall  f1-score   support

       OTHER    0.97959   0.88889   0.93204        54
  TRUE/FALSE    0.86667   0.97500   0.91765        40

    accuracy                        0.92553        94
   macro avg    0.92313   0.93194   0.92484        94
weighted avg    0.93154   0.92553   0.92591        94

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 0.866

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

Accuracy : 0.904
Classification Report
              precision    recall  f1-score   support

       OTHER    0.94118   0.88889   0.91429        54
  TRUE/FALSE    0.86047   0.92500   0.89157        40

    accuracy                        0.90426        94
   macro avg    0.90082   0.90694   0.90293        94
weighted avg    0.90683   0.90426   0.90462        94

Ensemble des meilleurs paramètres :
     n_estimators: 200
```

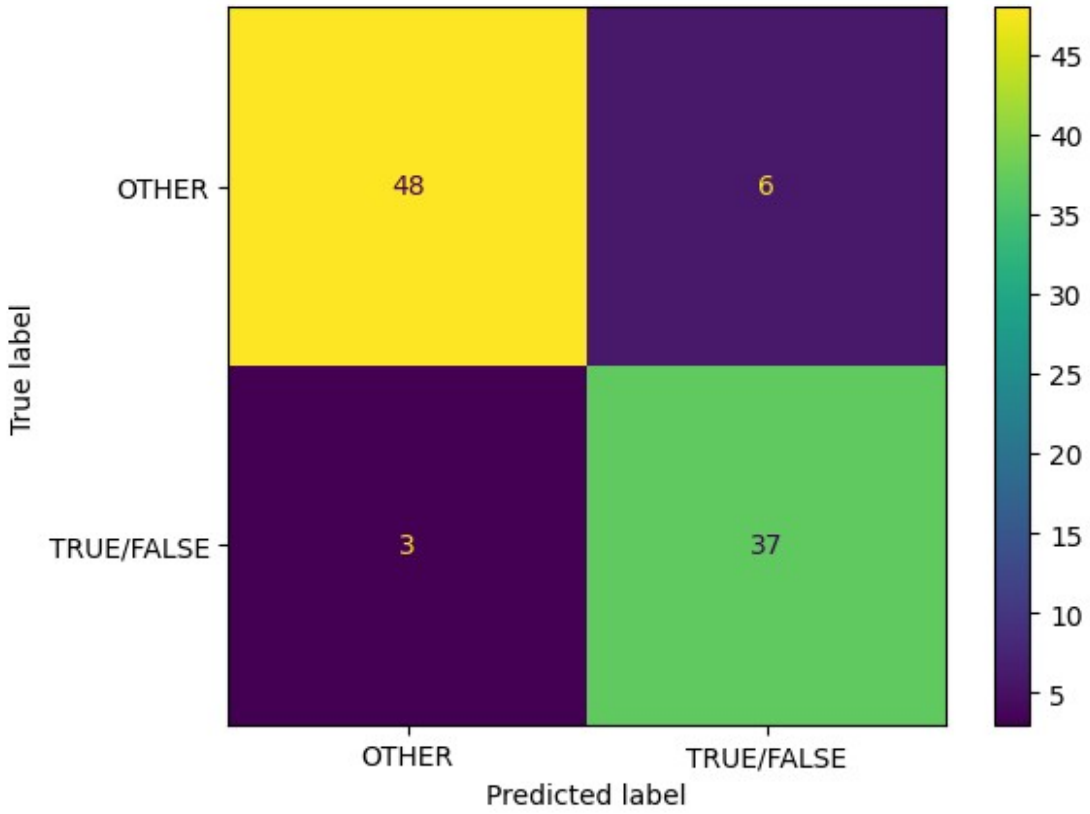
```
     max_features: 'sqrt'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.859

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

Accuracy : 0.947
Classification Report
              precision    recall  f1-score   support

       OTHER    0.98039   0.92593   0.95238        54
  TRUE/FALSE    0.90698   0.97500   0.93976        40

    accuracy                        0.94681        94
   macro avg    0.94368   0.95046   0.94607        94
weighted avg    0.94915   0.94681   0.94701        94

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 0.864

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

Accuracy : 0.957
Classification Report
              precision    recall  f1-score   support

       OTHER    1.00000   0.92593   0.96154        54
  TRUE/FALSE    0.90909   1.00000   0.95238        40

    accuracy                        0.95745        94
   macro avg    0.95455   0.96296   0.95696        94
weighted avg    0.96132   0.95745   0.95764        94

Ensemble des meilleurs paramètres :
     n_estimators: 100
     max_features: 'log2'
```

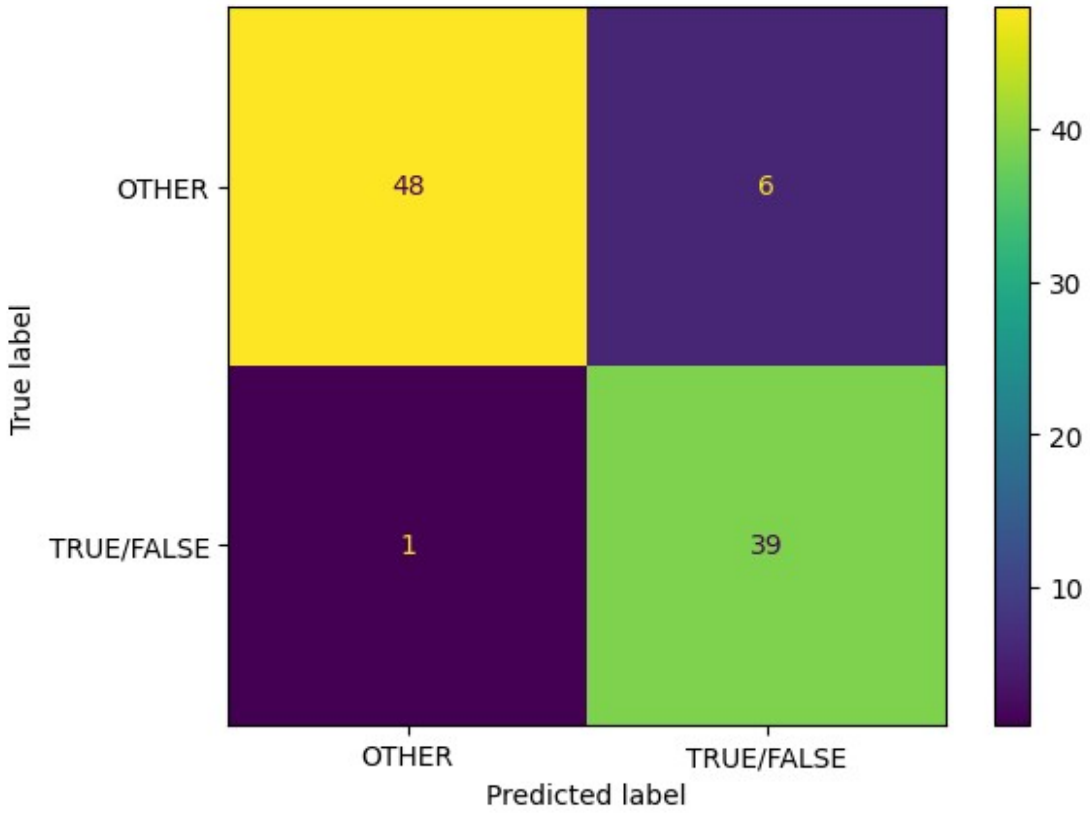

OTHER
TRUE/FALSE
True label
48
6
0
40
OTHER
TRUE/FALSE
Predicted label
40
30
20
10
0

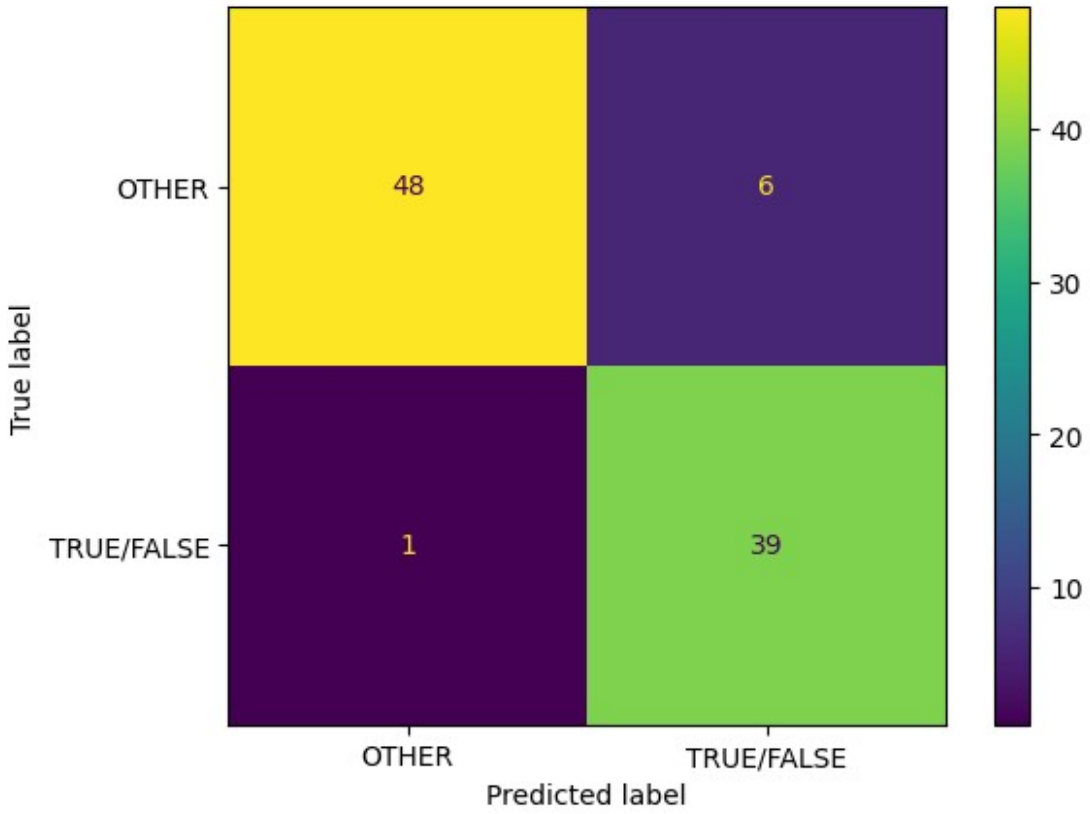

OTHER
TRUE/FALSE
True label
48
6
0
40
OTHER
TRUE/FALSE
Predicted label
40
30
20
10
0

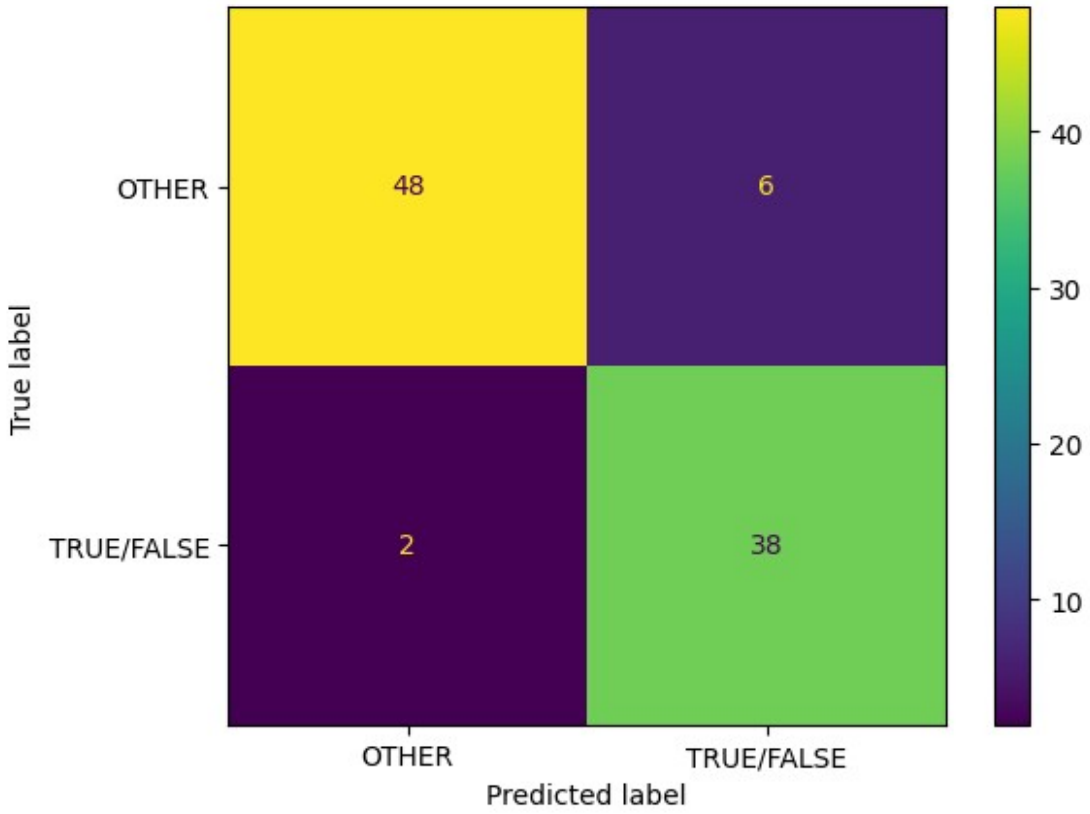

OTHER
TRUE/FALSE
True label
48
6
0
40
OTHER
TRUE/FALSE
Predicted label
40
30
20
10
0

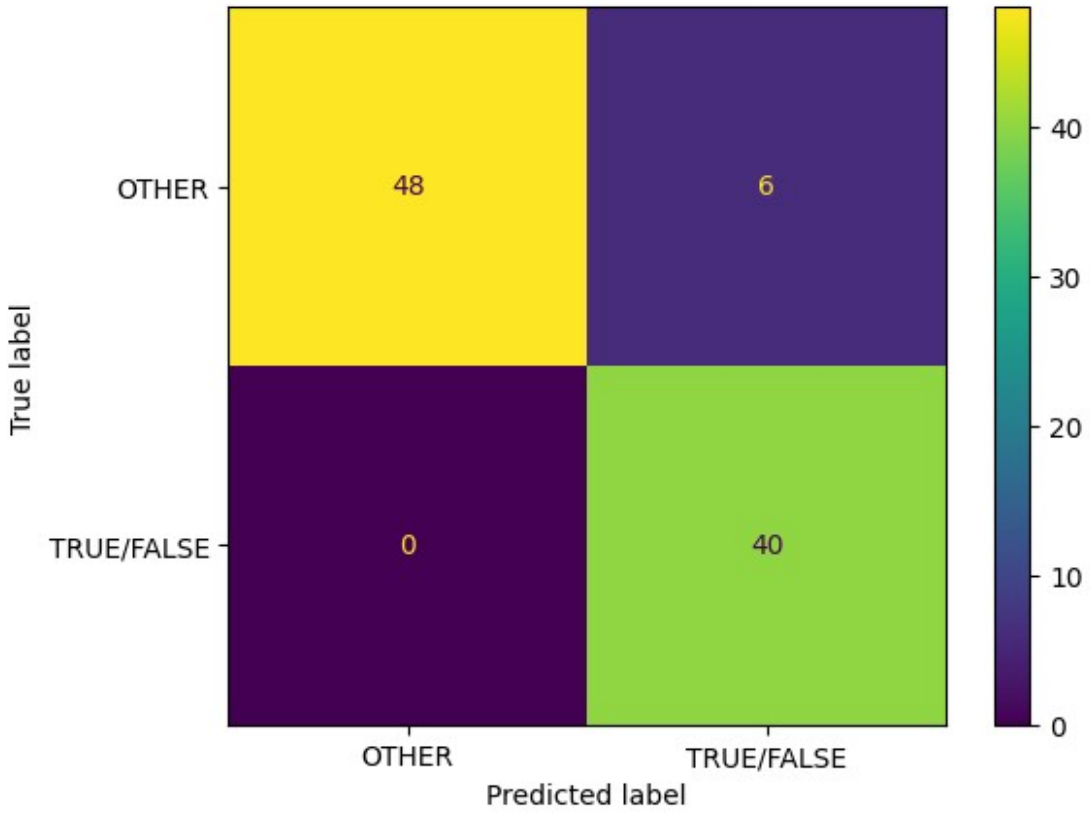

OTHER
TRUE/FALSE
True label
48
6
0
40
OTHER
TRUE/FALSE
Predicted label
40
30
20
10
0

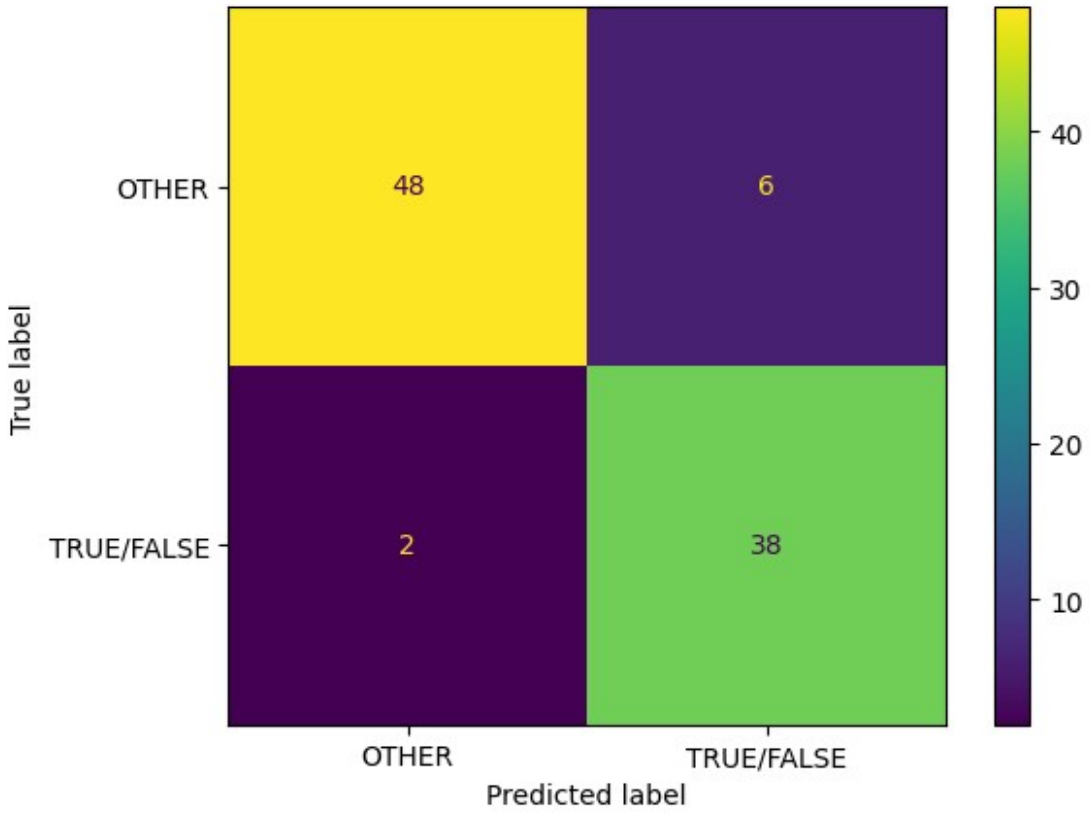

OTHER
TRUE/FALSE
True label
48
6
3
37
OTHER
TRUE/FALSE
Predicted label
45
40
35
30
25
20
15
10
5

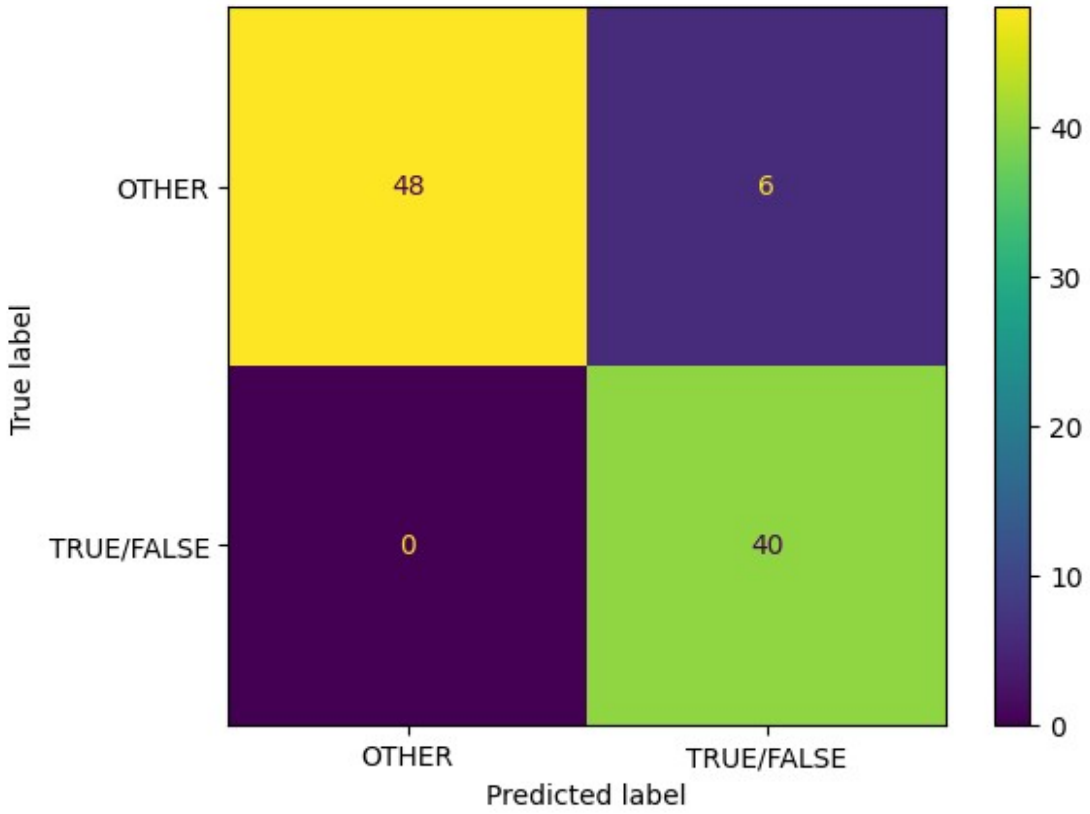

OTHER
TRUE/FALSE
True label
48
6
1
39
OTHER
TRUE/FALSE
Predicted label
40
30
20
10

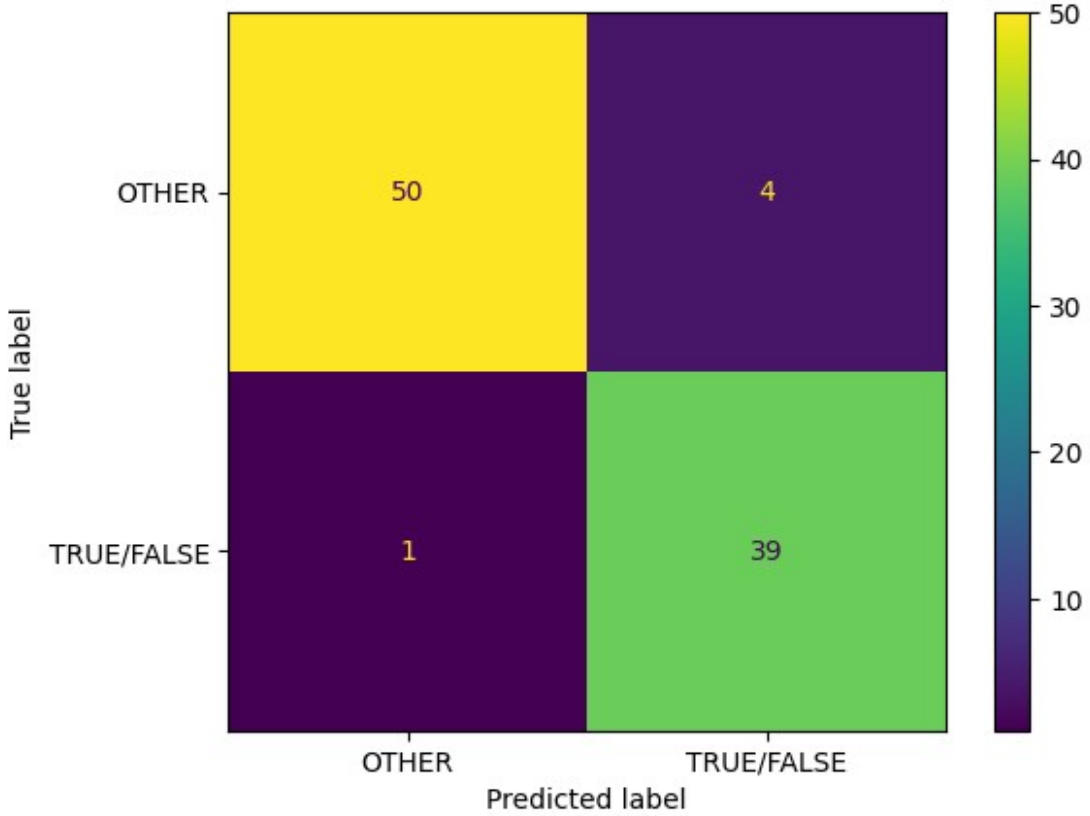

OTHER
TRUE/FALSE
True label
48
6
1
39
OTHER
TRUE/FALSE
Predicted label
40
30
20
10

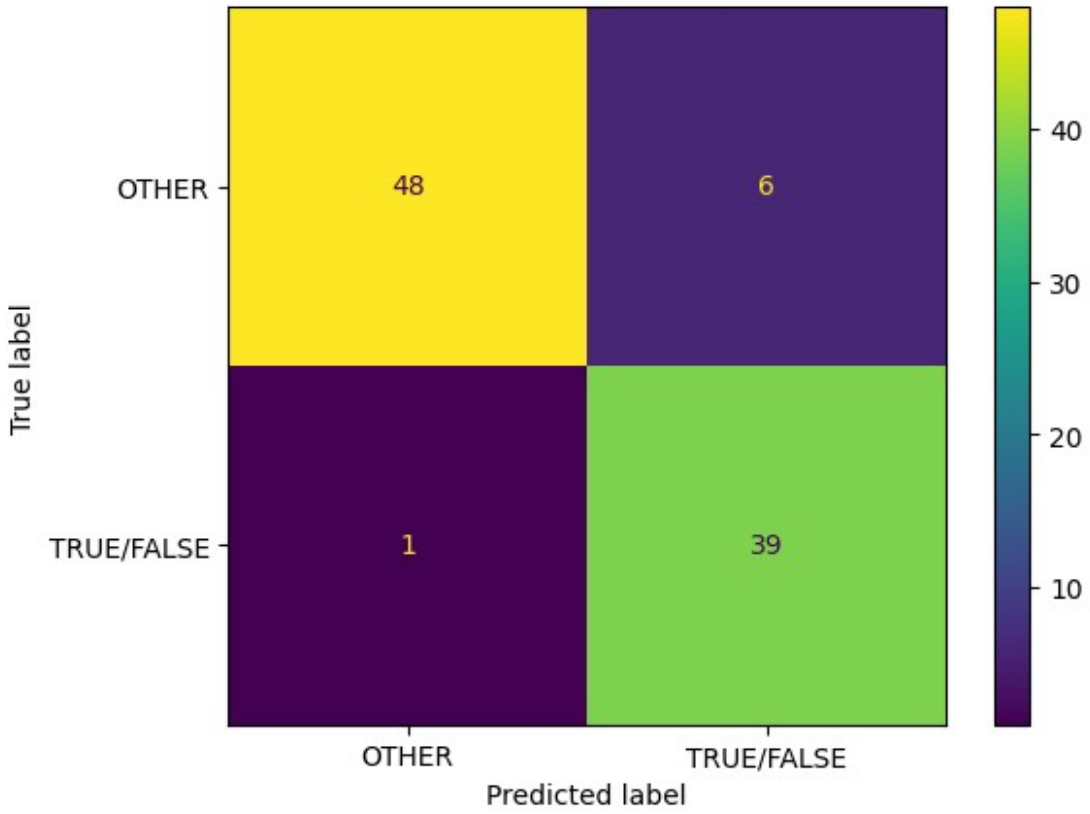

OTHER
TRUE/FALSE
True label
48
6
2
38
OTHER
TRUE/FALSE
Predicted label
40
30
20
10

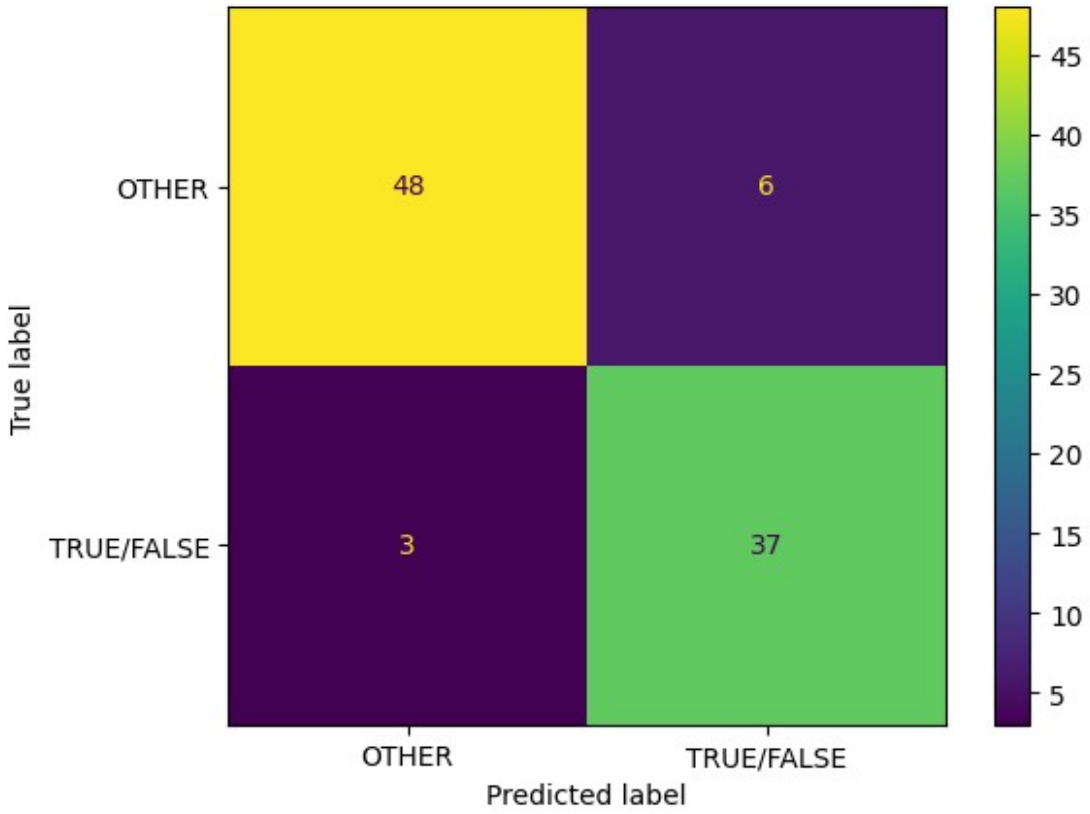

OTHER
TRUE/FALSE
True label
48
6
0
40
OTHER
TRUE/FALSE
Predicted label
40
30
20
10
0

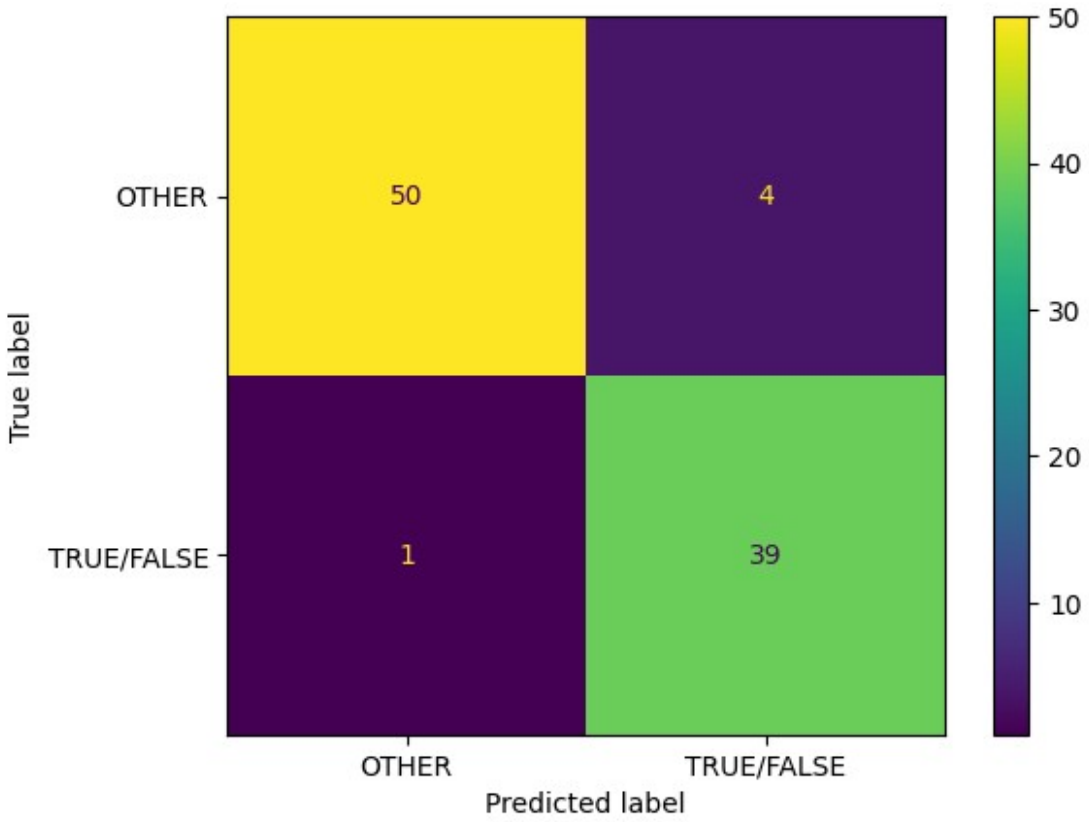

OTHER
TRUE/FALSE
True label
48
6
2
38
OTHER
TRUE/FALSE
Predicted label
40
30
20
10

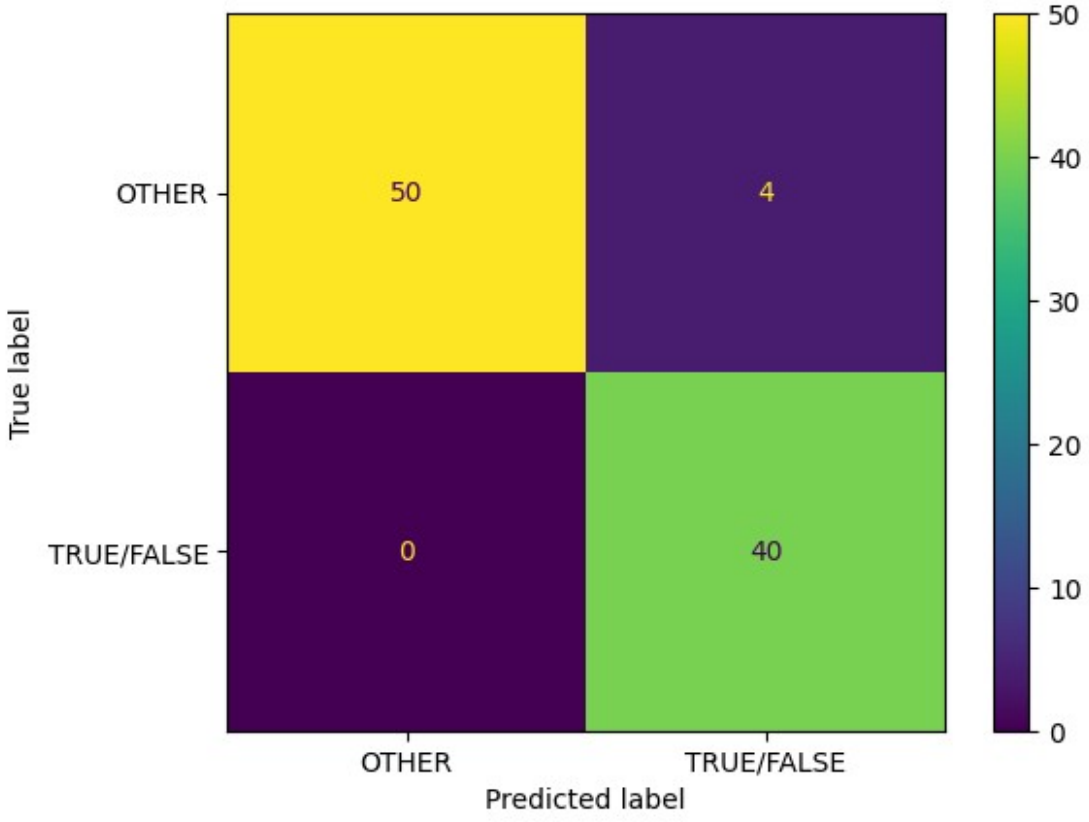

OTHER
TRUE/FALSE
True label
48
6
0
40
OTHER
TRUE/FALSE
Predicted label
40
30
20
10
0


OTHER
TRUE/FALSE
True label
50
4
1
39
OTHER
TRUE/FALSE
Predicted label
50
40
30
20
10


OTHER
TRUE/FALSE
True label
48
6
1
39
OTHER
TRUE/FALSE
Predicted label
40
30
20
10


OTHER
TRUE/FALSE
True label
48
6
3
37
OTHER
TRUE/FALSE
Predicted label
45
40
35
30
25
20
15
10
5


OTHER
TRUE/FALSE
True label
50
4
1
39
OTHER
TRUE/FALSE
Predicted label
50
40
30
20
10


OTHER
TRUE/FALSE
True label
50
4
0
40
OTHER
TRUE/FALSE
Predicted label
50
40
30
20
10
0

## 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 [('RF', 0.8210526315789475, 0.05718916060098471),
('SVM', 0.7996443812233286, 0.06288003704595112), ('CART',
0.788904694167852, 0.06359975125948711), ('MultinomialNB',
0.7278805120910385, 0.07587821429389406), ('LogisticRegression',
```

```
0.7251066856330014, 0.06988773070842395), ('KNN', 0.6876955903271693,
0.05897223074770526)]
```

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_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, 1,2,5,7,10]},
             {'gamma': [0.001, 0.01, 0.1,0.2,0.3,0.5,0.7,1]},
             {'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_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 18 candidates, totalling 90 fits
```

```
meilleur score 0.891

meilleur estimateur SVC(gamma=0.3, random_state=42)

Accuracy : 0.926
Classification Report
              precision    recall  f1-score   support

       OTHER    0.97959   0.88889   0.93204        54
  TRUE/FALSE    0.86667   0.97500   0.91765        40

    accuracy                        0.92553        94
   macro avg    0.92313   0.93194   0.92484        94
weighted avg    0.93154   0.92553   0.92591        94

Ensemble des meilleurs paramètres :
     C: 1.0
     gamma: 0.3
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.891

meilleur estimateur SVC(gamma=0.3, random_state=42)

Accuracy : 0.926
Classification Report
              precision    recall  f1-score   support

       OTHER    0.97959   0.88889   0.93204        54
  TRUE/FALSE    0.86667   0.97500   0.91765        40

    accuracy                        0.92553        94
   macro avg    0.92313   0.93194   0.92484        94
weighted avg    0.93154   0.92553   0.92591        94

Ensemble des meilleurs paramètres :
     C: 1.0
     gamma: 0.3
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.891

meilleur estimateur SVC(gamma=0.5, random_state=42)

Accuracy : 0.936
Classification Report
              precision    recall  f1-score   support
```

```
       OTHER    1.00000   0.88889   0.94118        54
  TRUE/FALSE    0.86957   1.00000   0.93023        40

    accuracy                        0.93617        94
   macro avg    0.93478   0.94444   0.93570        94
weighted avg    0.94450   0.93617   0.93652        94

Ensemble des meilleurs paramètres :
     C: 1.0
     gamma: 0.5
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.891

meilleur estimateur SVC(gamma=0.5, random_state=42)

Accuracy : 0.936
Classification Report
              precision    recall  f1-score   support

       OTHER    1.00000   0.88889   0.94118        54
  TRUE/FALSE    0.86957   1.00000   0.93023        40

    accuracy                        0.93617        94
   macro avg    0.93478   0.94444   0.93570        94
weighted avg    0.94450   0.93617   0.93652        94

Ensemble des meilleurs paramètres :
     C: 1.0
     gamma: 0.5
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.824

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

Accuracy : 0.926
Classification Report
              precision    recall  f1-score   support

       OTHER    0.91228   0.96296   0.93694        54
  TRUE/FALSE    0.94595   0.87500   0.90909        40

    accuracy                        0.92553        94
   macro avg    0.92911   0.91898   0.92301        94
weighted avg    0.92661   0.92553   0.92509        94
```

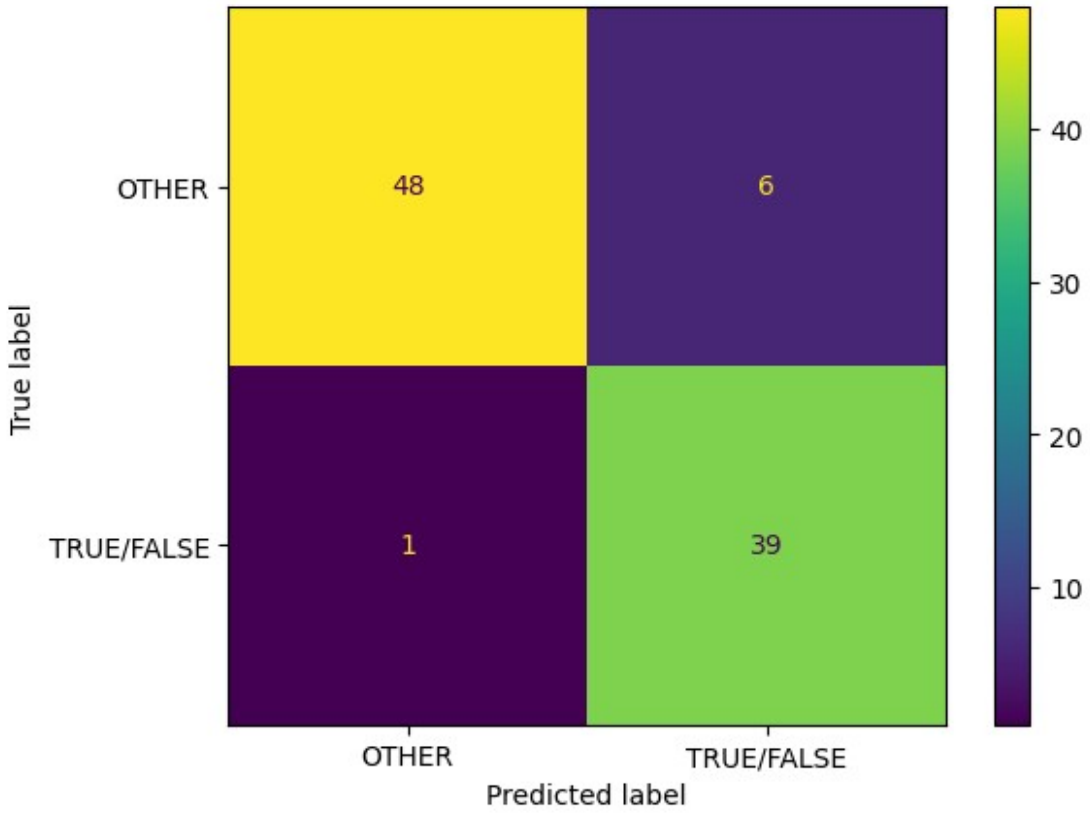
```
Ensemble des meilleurs paramètres :
     C: 2
     gamma: 'scale'
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.821

meilleur estimateur SVC(C=5, random_state=42)

Accuracy : 0.915
Classification Report
              precision    recall  f1-score   support

       OTHER    0.89655   0.96296   0.92857        54
  TRUE/FALSE    0.94444   0.85000   0.89474        40

    accuracy                        0.91489        94
   macro avg    0.92050   0.90648   0.91165        94
weighted avg    0.91693   0.91489   0.91417        94

Ensemble des meilleurs paramètres :
     C: 5
     gamma: 'scale'
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.824

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

Accuracy : 0.926
Classification Report
              precision    recall  f1-score   support

       OTHER    0.91228   0.96296   0.93694        54
  TRUE/FALSE    0.94595   0.87500   0.90909        40

    accuracy                        0.92553        94
   macro avg    0.92911   0.91898   0.92301        94
weighted avg    0.92661   0.92553   0.92509        94

Ensemble des meilleurs paramètres :
     C: 2
     gamma: 'scale'
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
```

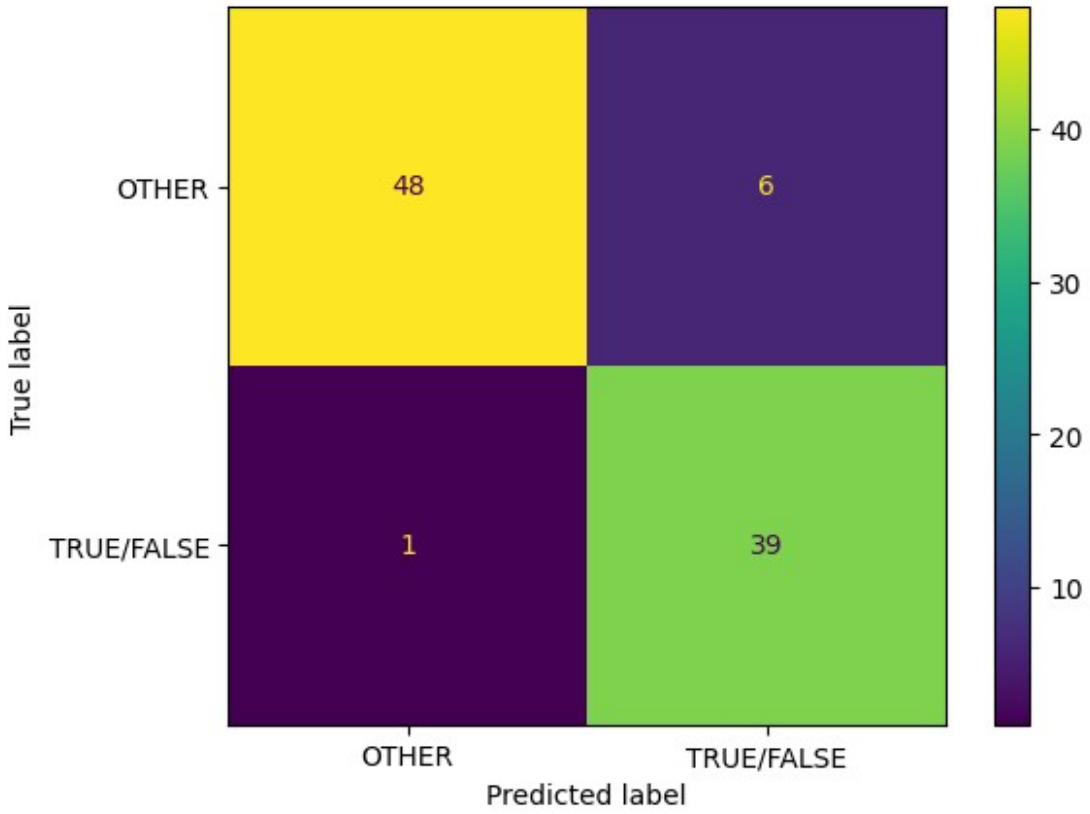
```
meilleur score 0.818

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

Accuracy : 0.883
Classification Report
              precision    recall  f1-score   support

       OTHER    0.87719   0.92593   0.90090        54
  TRUE/FALSE    0.89189   0.82500   0.85714        40

    accuracy                        0.88298        94
   macro avg    0.88454   0.87546   0.87902        94
weighted avg    0.88345   0.88298   0.88228        94

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 0.805

meilleur estimateur RandomForestClassifier(random_state=42)

Accuracy : 0.936
Classification Report
              precision    recall  f1-score   support

       OTHER    0.92857   0.96296   0.94545        54
  TRUE/FALSE    0.94737   0.90000   0.92308        40

    accuracy                        0.93617        94
   macro avg    0.93797   0.93148   0.93427        94
weighted avg    0.93657   0.93617   0.93593        94

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 0.821

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

Accuracy : 0.926
Classification Report
              precision    recall  f1-score   support
```

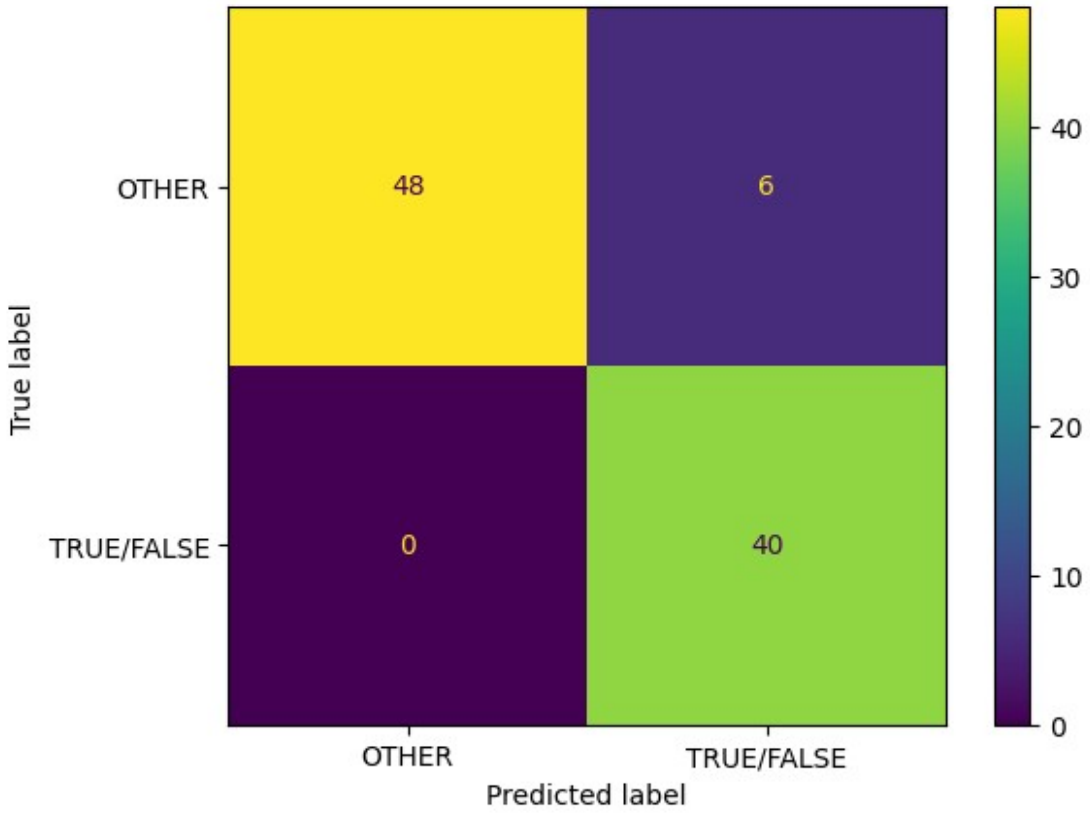
```
       OTHER    0.91228   0.96296   0.93694        54
  TRUE/FALSE    0.94595   0.87500   0.90909        40

    accuracy                        0.92553        94
   macro avg    0.92911   0.91898   0.92301        94
weighted avg    0.92661   0.92553   0.92509        94

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 0.813

meilleur estimateur RandomForestClassifier(n_estimators=50,
random_state=42)

Accuracy : 0.894
Classification Report
              precision    recall  f1-score   support

       OTHER    0.89286   0.92593   0.90909        54
  TRUE/FALSE    0.89474   0.85000   0.87179        40

    accuracy                        0.89362        94
   macro avg    0.89380   0.88796   0.89044        94
weighted avg    0.89366   0.89362   0.89322        94

Ensemble des meilleurs paramètres :
     n_estimators: 50
     max_features: 'sqrt'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.805

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

Accuracy : 0.926
Classification Report
              precision    recall  f1-score   support

       OTHER    0.91228   0.96296   0.93694        54
  TRUE/FALSE    0.94595   0.87500   0.90909        40

    accuracy                        0.92553        94
   macro avg    0.92911   0.91898   0.92301        94
weighted avg    0.92661   0.92553   0.92509        94
```

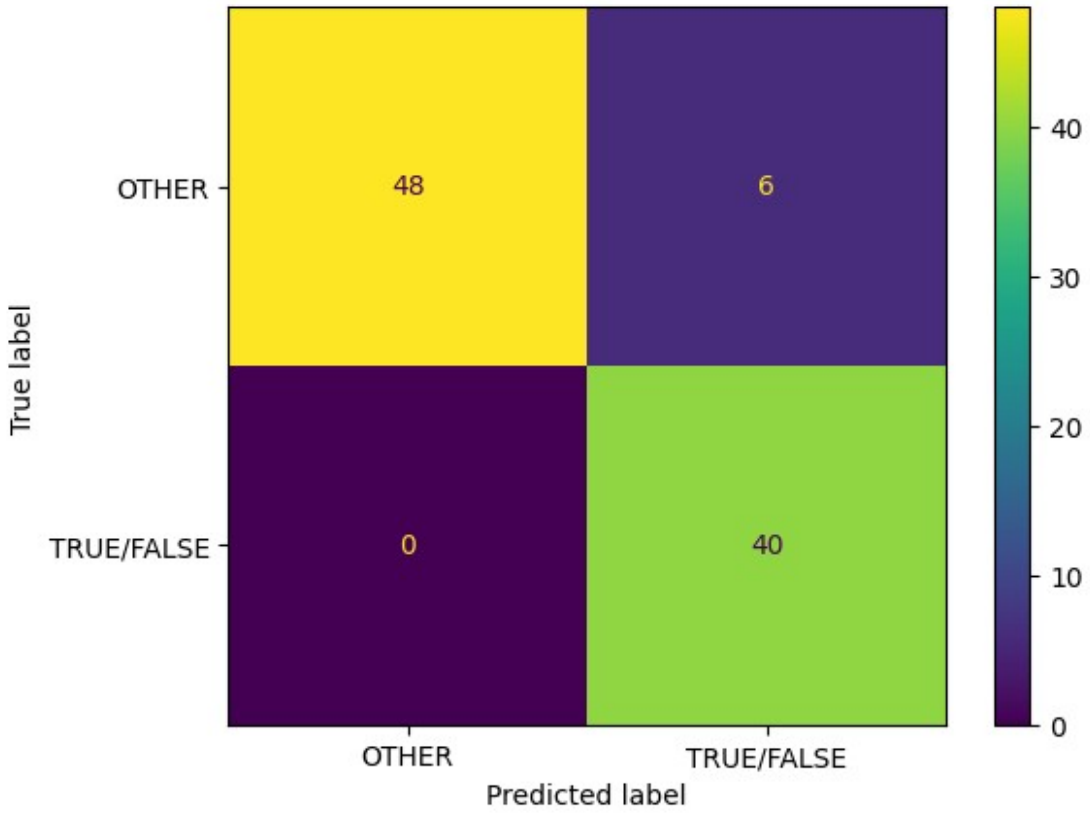
```
Ensemble des meilleurs paramètres :
     n_estimators: 300
     max_features: 'sqrt'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.824

meilleur estimateur RandomForestClassifier(random_state=42)

Accuracy : 0.926
Classification Report
              precision    recall  f1-score   support

       OTHER    0.91228   0.96296   0.93694        54
  TRUE/FALSE    0.94595   0.87500   0.90909        40

    accuracy                        0.92553        94
   macro avg    0.92911   0.91898   0.92301        94
weighted avg    0.92661   0.92553   0.92509        94

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 0.826

meilleur estimateur RandomForestClassifier(n_estimators=50,
random_state=42)

Accuracy : 0.904
Classification Report
              precision    recall  f1-score   support

       OTHER    0.88136   0.96296   0.92035        54
  TRUE/FALSE    0.94286   0.82500   0.88000        40

    accuracy                        0.90426        94
   macro avg    0.91211   0.89398   0.90018        94
weighted avg    0.90753   0.90426   0.90318        94

Ensemble des meilleurs paramètres :
     n_estimators: 50
     max_features: 'sqrt'
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.821
```

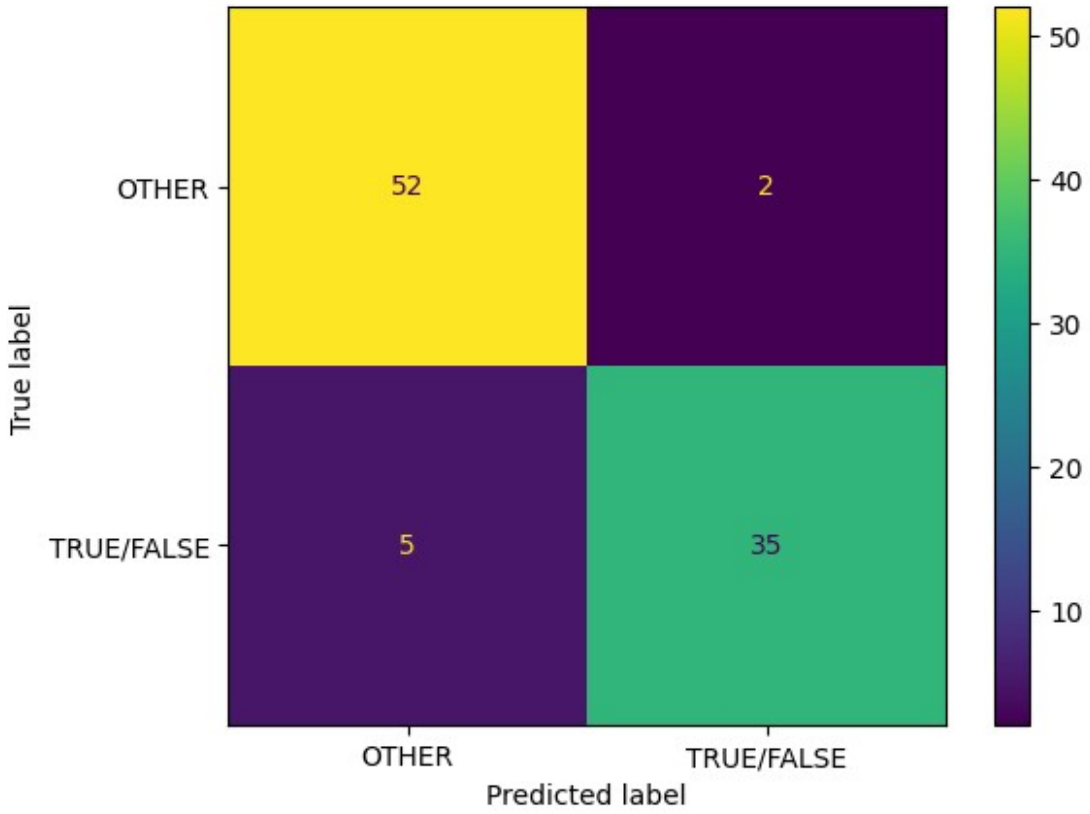
```
meilleur estimateur RandomForestClassifier(max_features='log2',
random_state=42)

Accuracy : 0.904
Classification Report
              precision    recall  f1-score   support

       OTHER    0.88136   0.96296   0.92035        54
  TRUE/FALSE    0.94286   0.82500   0.88000        40

    accuracy                        0.90426        94
   macro avg    0.91211   0.89398   0.90018        94
weighted avg    0.90753   0.90426   0.90318        94

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 0.818

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

Accuracy : 0.894
Classification Report
              precision    recall  f1-score   support

       OTHER    0.86667   0.96296   0.91228        54
  TRUE/FALSE    0.94118   0.80000   0.86486        40

    accuracy                        0.89362        94
   macro avg    0.90392   0.88148   0.88857        94
weighted avg    0.89837   0.89362   0.89210        94

Ensemble des meilleurs paramètres :
     n_estimators: 100
     max_features: 'log2'
```

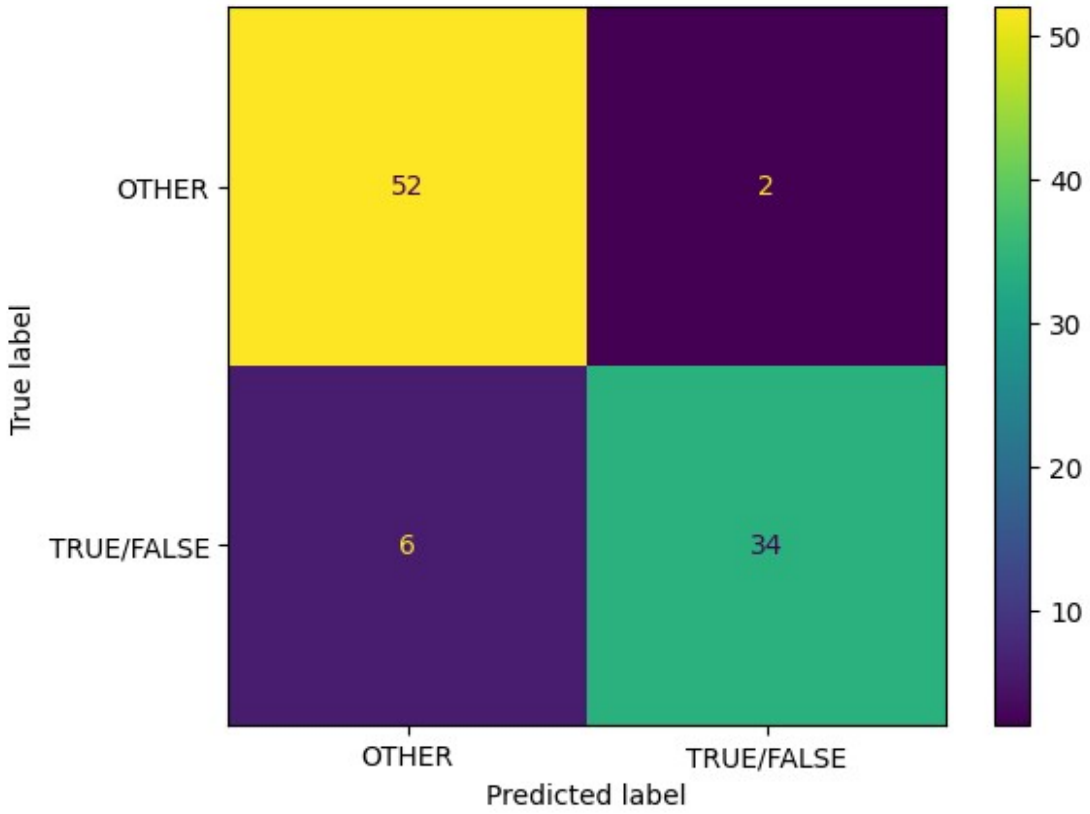

OTHER
TRUE/FALSE
True label
48
6
1
39
OTHER
TRUE/FALSE
Predicted label
40
30
20
10

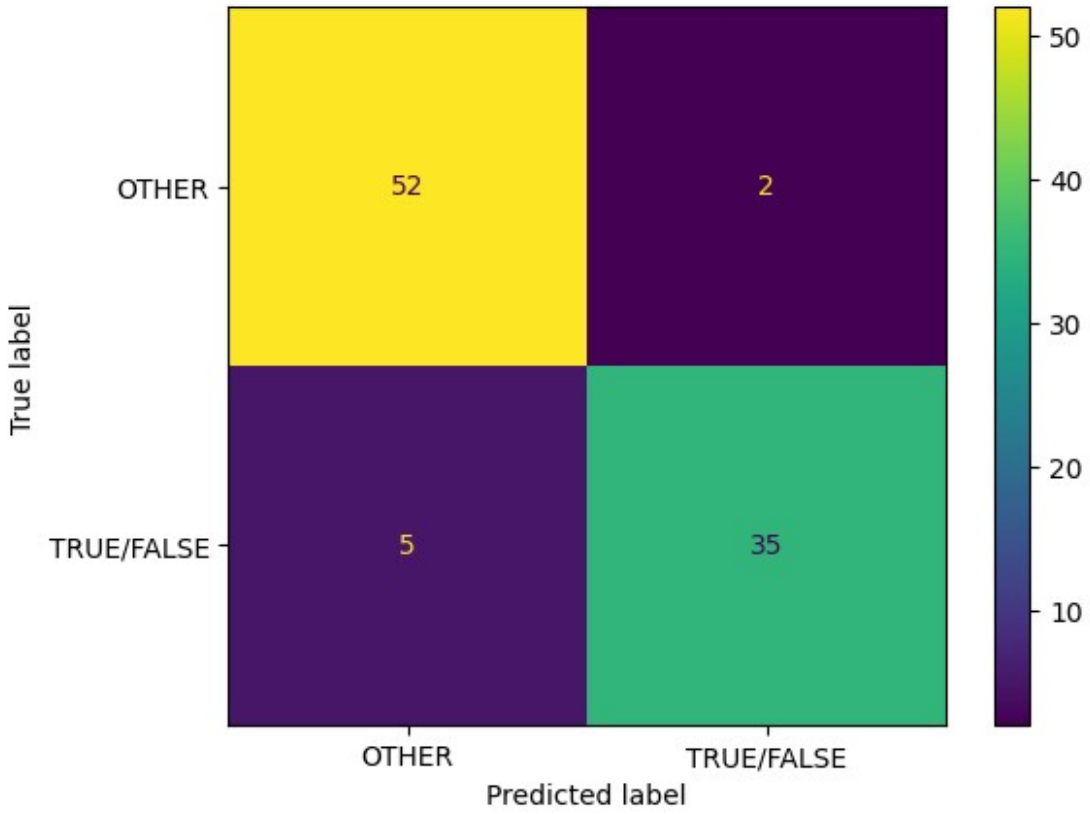

OTHER
TRUE/FALSE
True label
48
6
1
39
OTHER
TRUE/FALSE
Predicted label
40
30
20
10

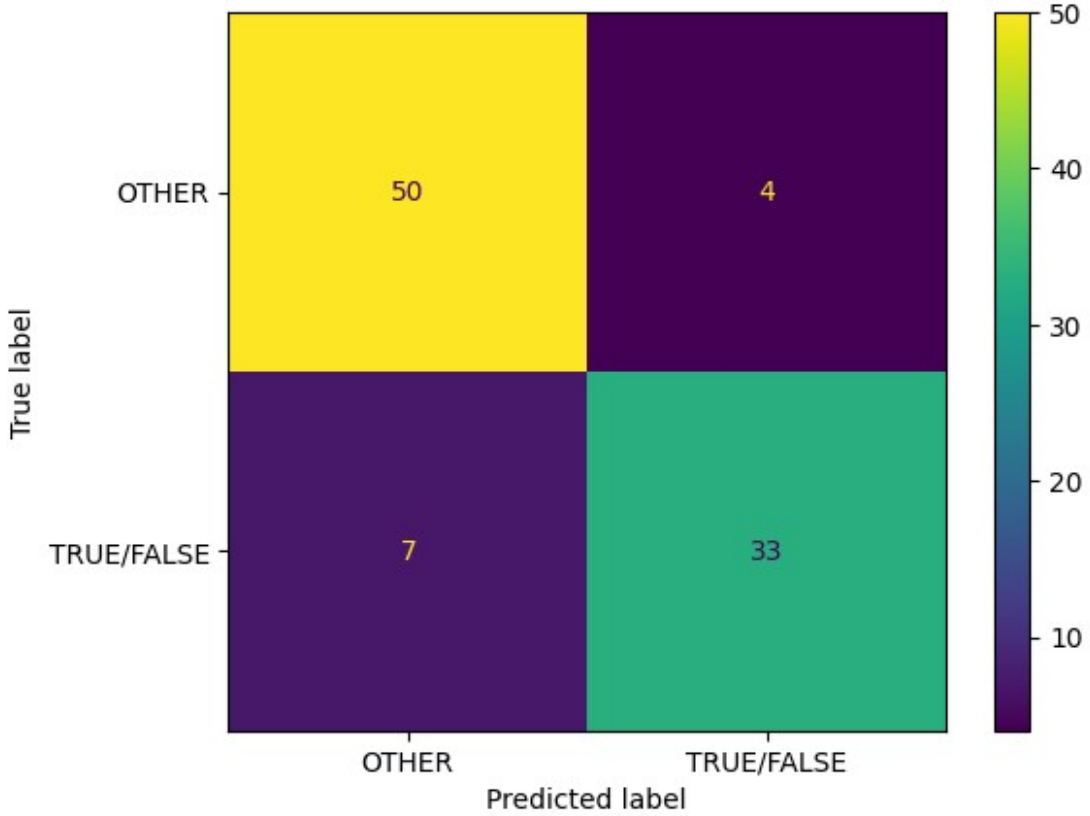

OTHER
TRUE/FALSE
True label
48
6
0
40
OTHER
TRUE/FALSE
Predicted label
40
30
20
10
0

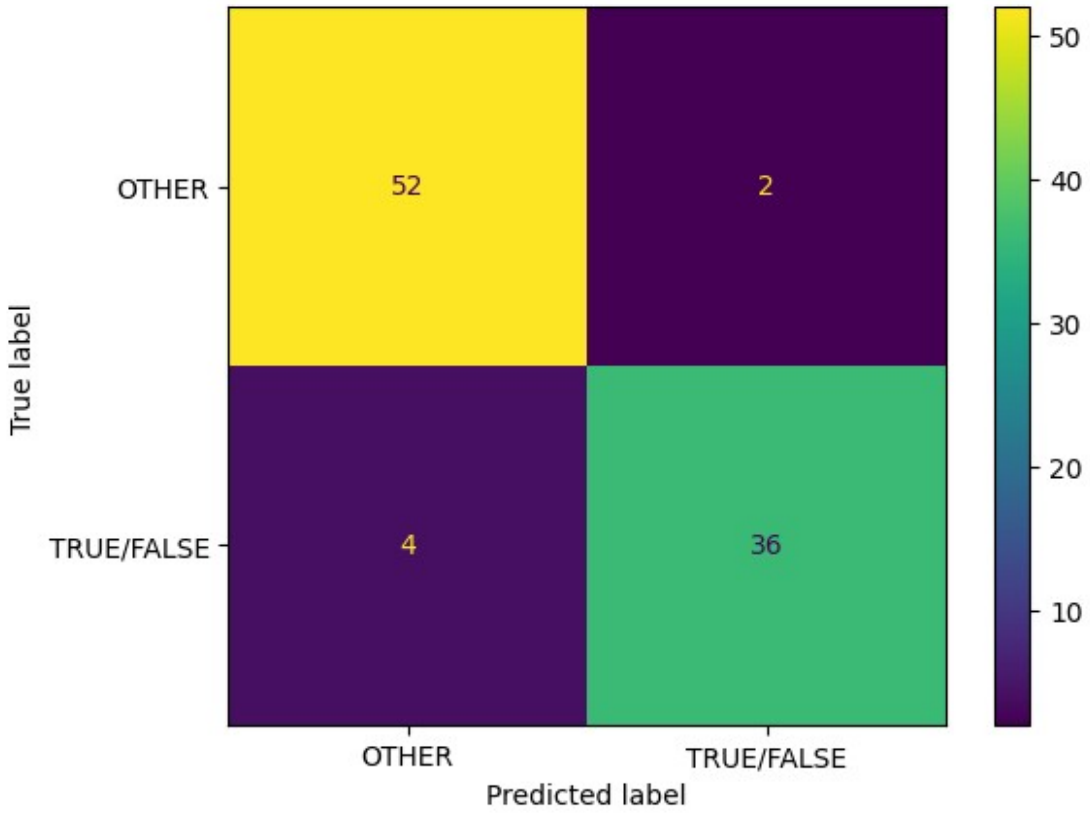

OTHER
TRUE/FALSE
True label
48
6
0
40
OTHER
TRUE/FALSE
Predicted label
40
30
20
10
0

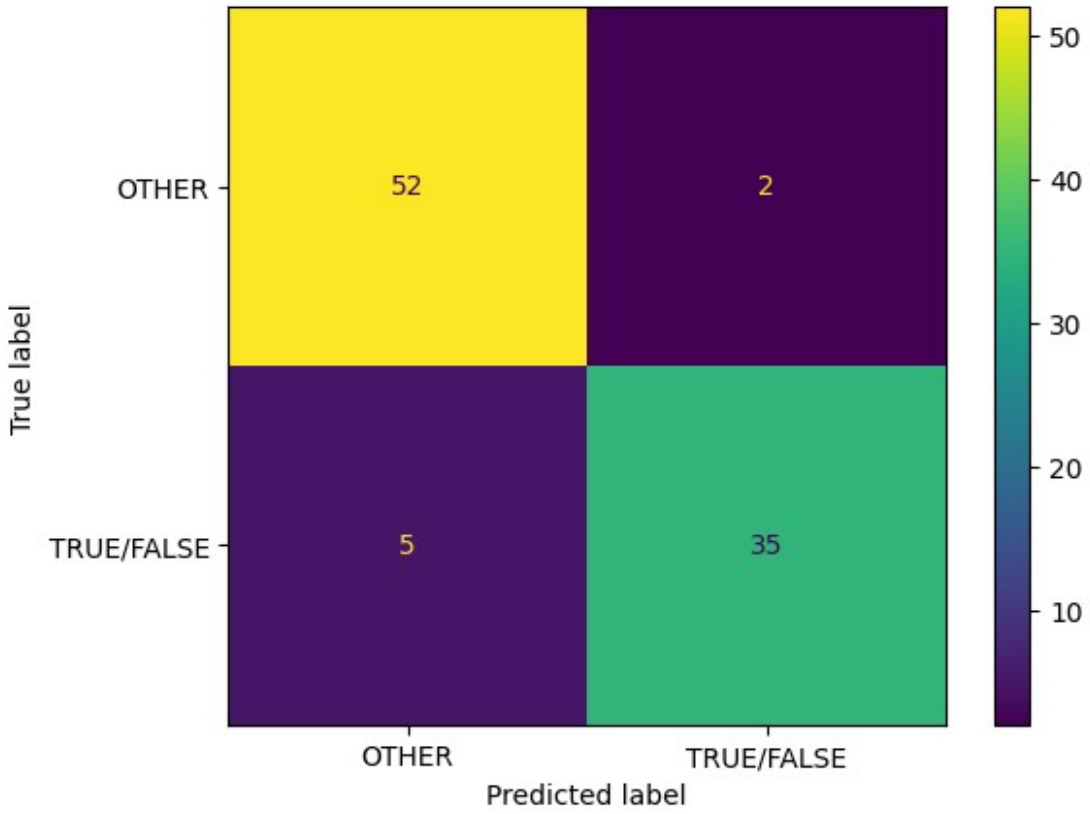

OTHER
TRUE/FALSE
True label
52
2
5
35
OTHER
TRUE/FALSE
Predicted label
50
40
30
20
10

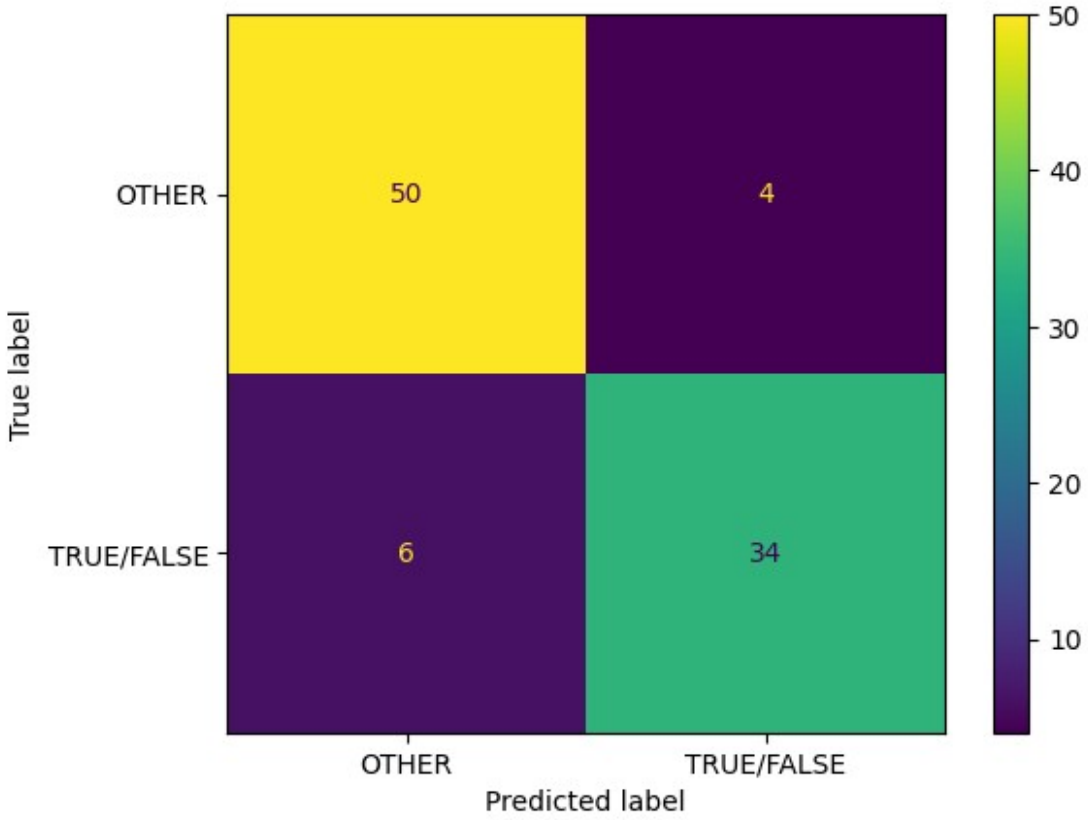

OTHER
TRUE/FALSE
True label
52
2
6
34
OTHER
TRUE/FALSE
Predicted label
50
40
30
20
10

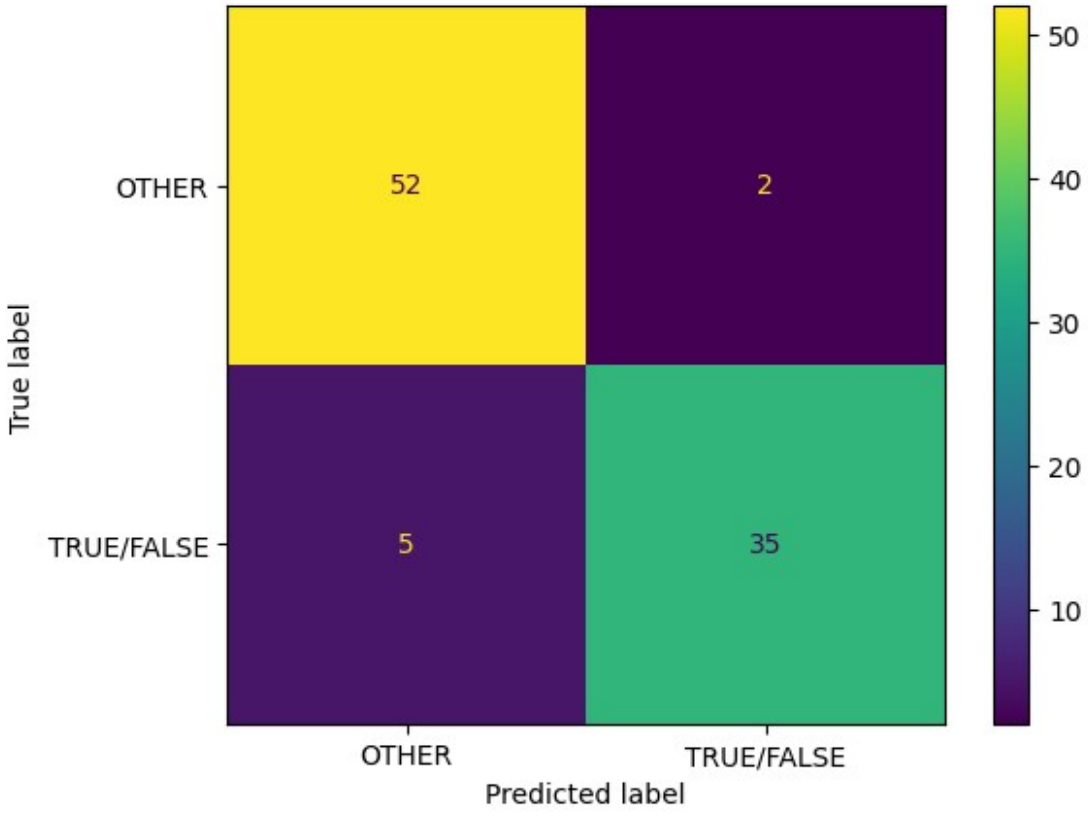

OTHER
TRUE/FALSE
True label
52
2
5
35
OTHER
TRUE/FALSE
Predicted label
50
40
30
20
10

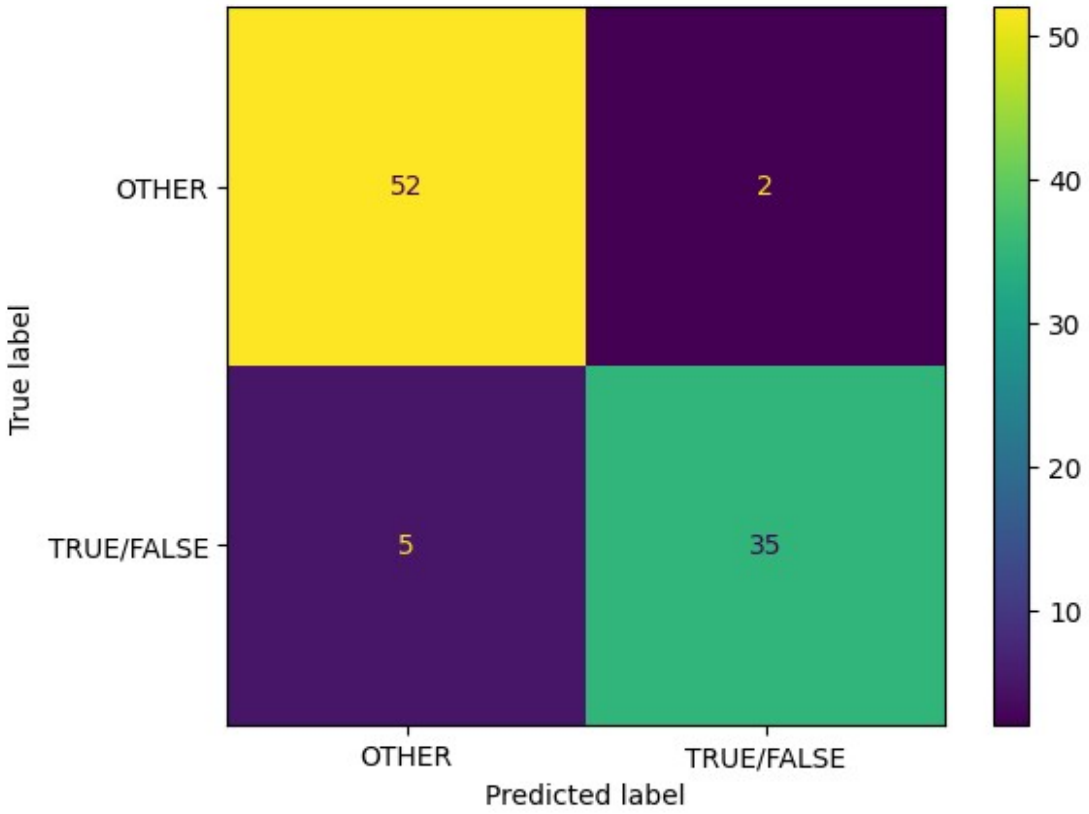

OTHER
TRUE/FALSE
True label
50
4
7
33
OTHER
TRUE/FALSE
Predicted label
50
40
30
20
10

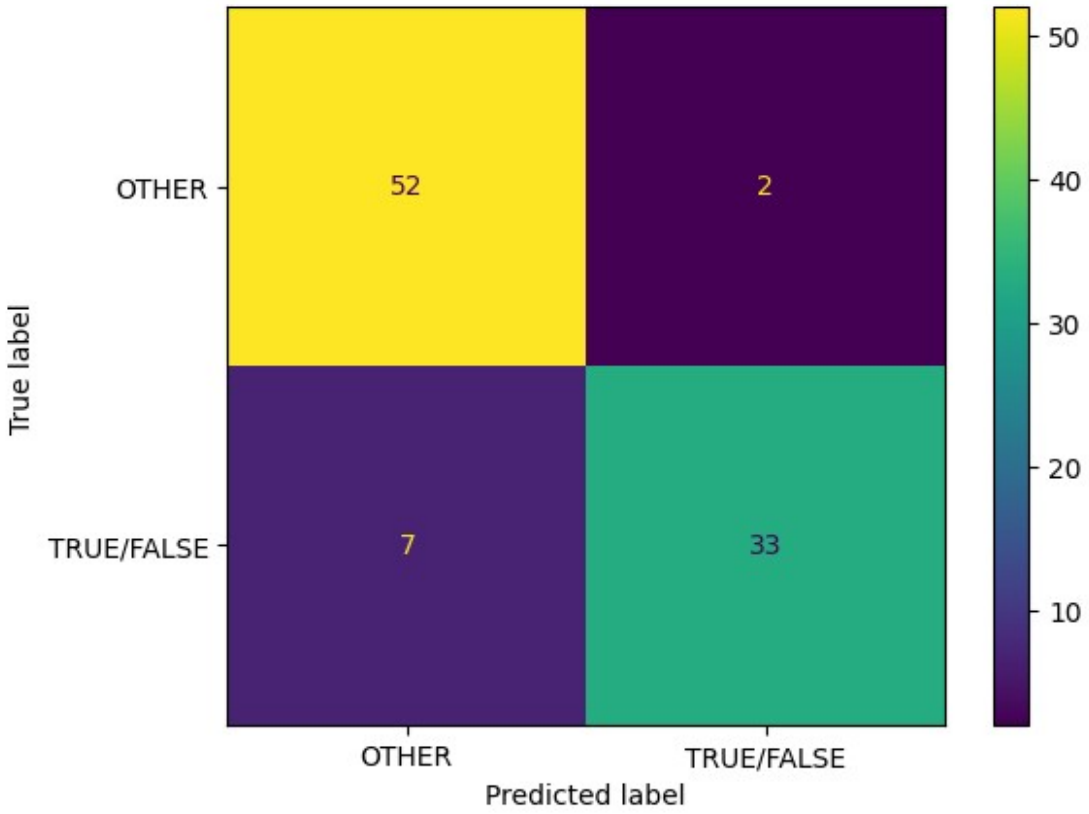

OTHER
TRUE/FALSE
True label
52
2
4
36
OTHER
TRUE/FALSE
Predicted label
50
40
30
20
10

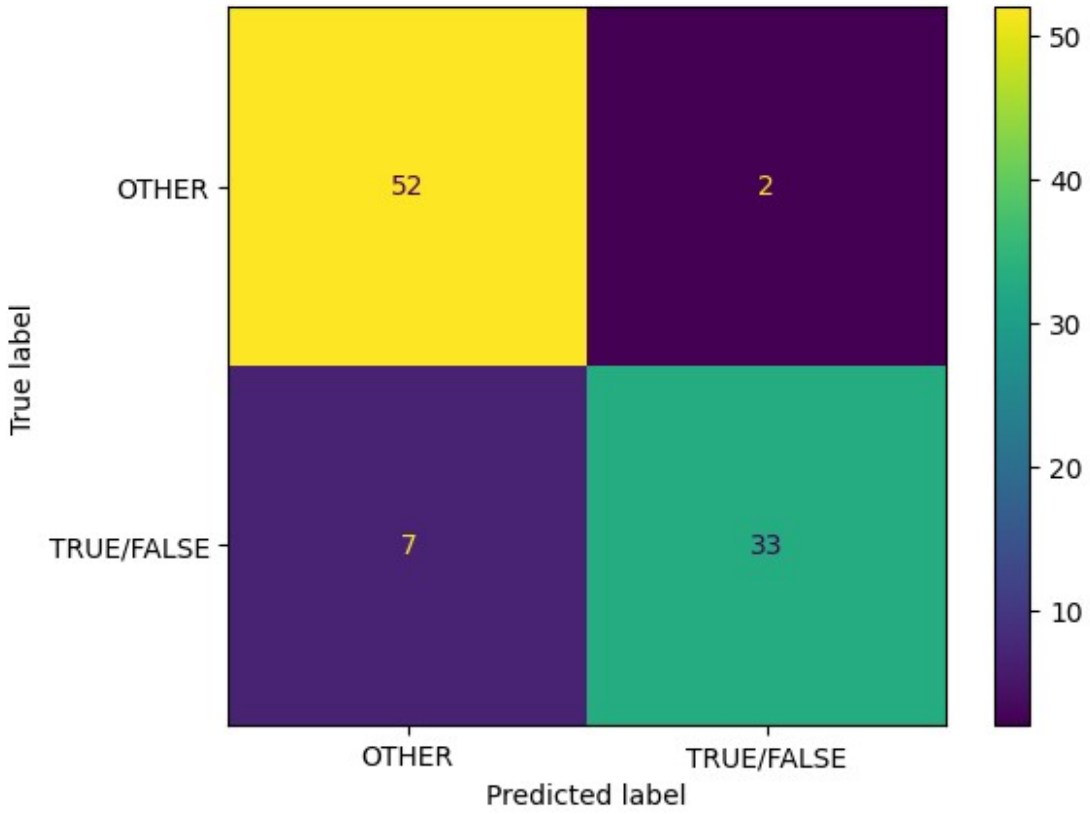

OTHER
TRUE/FALSE
True label
52
2
5
35
OTHER
TRUE/FALSE
Predicted label
50
40
30
20
10

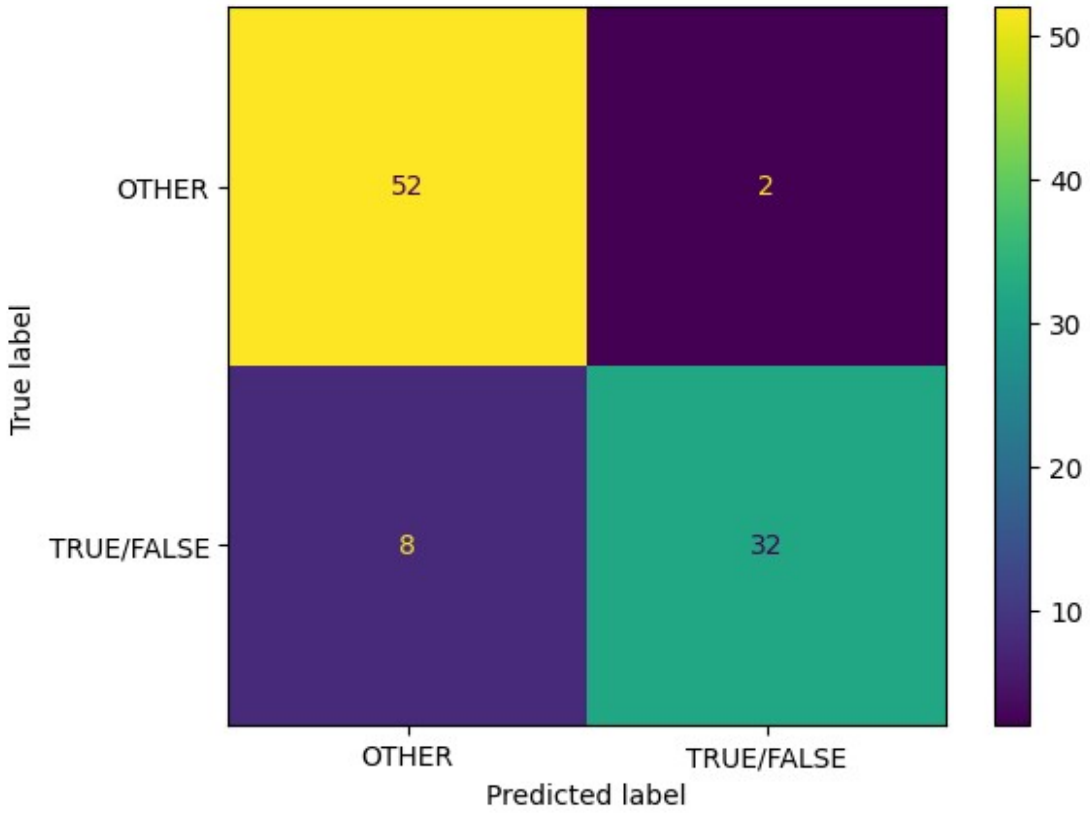

OTHER
TRUE/FALSE
True label
50
4
6
34
OTHER
TRUE/FALSE
Predicted label
50
40
30
20
10


OTHER
TRUE/FALSE
True label
52
2
5
35
OTHER
TRUE/FALSE
Predicted label
50
40
30
20
10


OTHER
TRUE/FALSE
True label
52
2
5
35
OTHER
TRUE/FALSE
Predicted label
50
40
30
20
10


OTHER
TRUE/FALSE
True label
52
2
7
33
OTHER
TRUE/FALSE
Predicted label
50
40
30
20
10


OTHER
TRUE/FALSE
True label
52
2
7
33
OTHER
TRUE/FALSE
Predicted label
50
40
30
20
10


OTHER
TRUE/FALSE
True label
52
2
8
32
OTHER
TRUE/FALSE
Predicted label
50
40
30
20
10

##**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', 0.8905405405405405, 0.027221821787829246),
('RF', 0.8799431009957326, 0.06448363802181939),
('LogisticRegression', 0.839402560455192, 0.052940923209927125),
```

```
('MultinomialNB', 0.8100995732574681, 0.03707451533081731), ('CART',
0.8076102418207682, 0.04190179250113181), ('KNN', 0.6737553342816501,
0.05973552910854227)]
```

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, 1,2,5,7,10]},
             {'gamma': [0.001, 0.01, 0.1,0.2,0.3,0.5,0.7,1]},
             {'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 18 candidates, totalling 90 fits
meilleur score 0.891

meilleur estimateur SVC(gamma=0.1, random_state=42)

Accuracy : 0.936
Classification Report
              precision    recall  f1-score   support

       OTHER    1.00000   0.88889   0.94118        54
  TRUE/FALSE    0.86957   1.00000   0.93023        40

    accuracy                        0.93617        94
   macro avg    0.93478   0.94444   0.93570        94
weighted avg    0.94450   0.93617   0.93652        94

Ensemble des meilleurs paramètres :
     C: 1.0
     gamma: 0.1
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.891

meilleur estimateur SVC(gamma=0.1, random_state=42)

Accuracy : 0.936
Classification Report
              precision    recall  f1-score   support

       OTHER    1.00000   0.88889   0.94118        54
  TRUE/FALSE    0.86957   1.00000   0.93023        40

    accuracy                        0.93617        94
   macro avg    0.93478   0.94444   0.93570        94
weighted avg    0.94450   0.93617   0.93652        94

Ensemble des meilleurs paramètres :
     C: 1.0
     gamma: 0.1
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.891

meilleur estimateur SVC(gamma=0.2, random_state=42)

Accuracy : 0.936
```

```
Classification Report
              precision    recall  f1-score   support

       OTHER    1.00000   0.88889   0.94118        54
  TRUE/FALSE    0.86957   1.00000   0.93023        40

    accuracy                        0.93617        94
   macro avg    0.93478   0.94444   0.93570        94
weighted avg    0.94450   0.93617   0.93652        94

Ensemble des meilleurs paramètres :
     C: 1.0
     gamma: 0.2
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.891

meilleur estimateur SVC(gamma=0.2, random_state=42)

Accuracy : 0.936
Classification Report
              precision    recall  f1-score   support

       OTHER    1.00000   0.88889   0.94118        54
  TRUE/FALSE    0.86957   1.00000   0.93023        40

    accuracy                        0.93617        94
   macro avg    0.93478   0.94444   0.93570        94
weighted avg    0.94450   0.93617   0.93652        94

Ensemble des meilleurs paramètres :
     C: 1.0
     gamma: 0.2
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.867

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

Accuracy : 0.894
Classification Report
              precision    recall  f1-score   support

       OTHER    0.94000   0.87037   0.90385        54
  TRUE/FALSE    0.84091   0.92500   0.88095        40

    accuracy                        0.89362        94
```

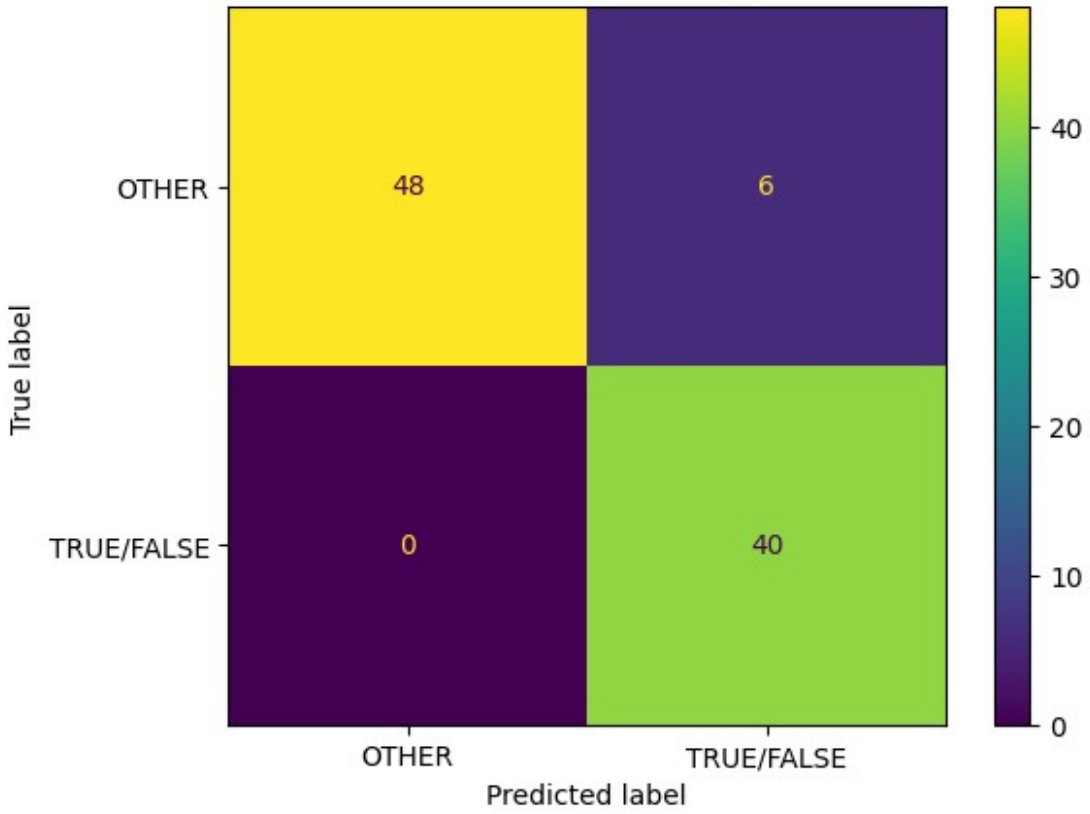
```
   macro avg    0.89045   0.89769   0.89240        94
weighted avg    0.89783   0.89362   0.89410        94

Ensemble des meilleurs paramètres :
     C: 1
     gamma: 'scale'
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.872

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

Accuracy : 0.926
Classification Report
              precision    recall  f1-score   support

       OTHER    0.97959   0.88889   0.93204        54
  TRUE/FALSE    0.86667   0.97500   0.91765        40

    accuracy                        0.92553        94
   macro avg    0.92313   0.93194   0.92484        94
weighted avg    0.93154   0.92553   0.92591        94

Ensemble des meilleurs paramètres :
     C: 1
     gamma: 'scale'
     kernel: 'rbf'
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.867

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

Accuracy : 0.915
Classification Report
              precision    recall  f1-score   support

       OTHER    0.96000   0.88889   0.92308        54
  TRUE/FALSE    0.86364   0.95000   0.90476        40

    accuracy                        0.91489        94
   macro avg    0.91182   0.91944   0.91392        94
weighted avg    0.91899   0.91489   0.91528        94

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

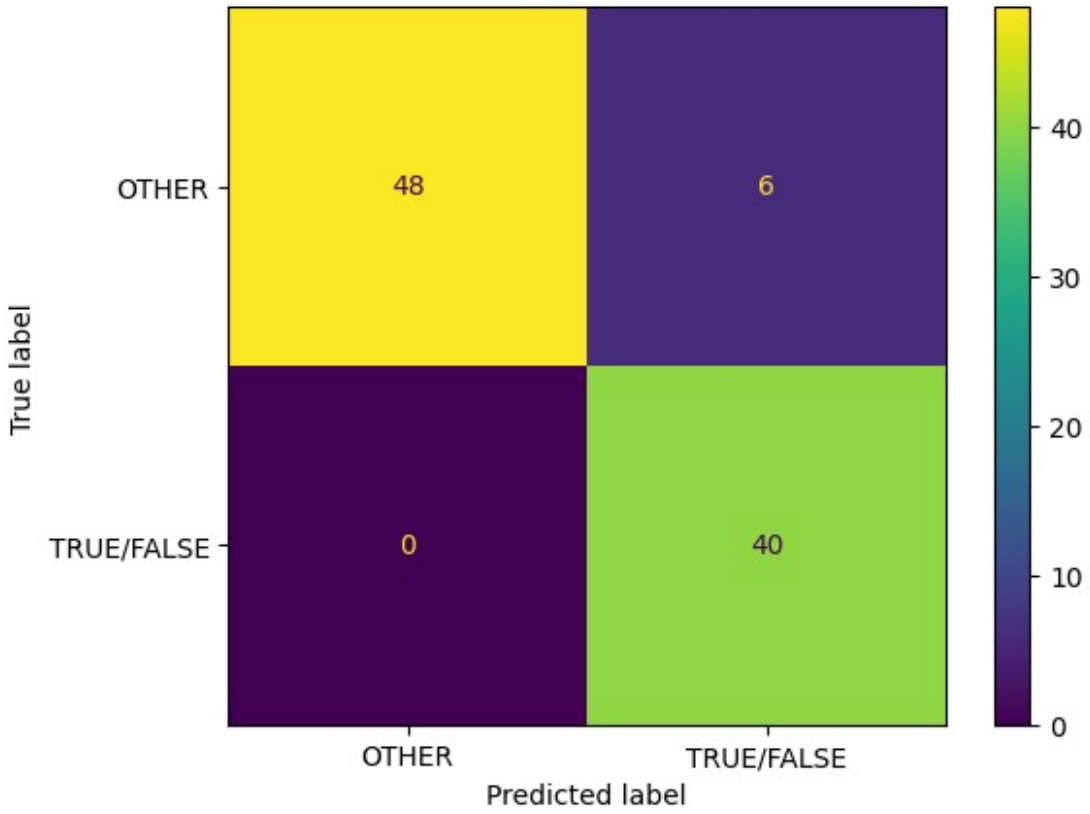
```
grid search fait
Fitting 5 folds for each of 18 candidates, totalling 90 fits
meilleur score 0.872

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

Accuracy : 0.904
Classification Report
              precision    recall  f1-score   support

       OTHER    0.95918   0.87037   0.91262        54
  TRUE/FALSE    0.84444   0.95000   0.89412        40

    accuracy                        0.90426        94
   macro avg    0.90181   0.91019   0.90337        94
weighted avg    0.91036   0.90426   0.90475        94

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 0.856

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

Accuracy : 0.936
Classification Report
              precision    recall  f1-score   support

       OTHER    1.00000   0.88889   0.94118        54
  TRUE/FALSE    0.86957   1.00000   0.93023        40

    accuracy                        0.93617        94
   macro avg    0.93478   0.94444   0.93570        94
weighted avg    0.94450   0.93617   0.93652        94

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 0.864

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

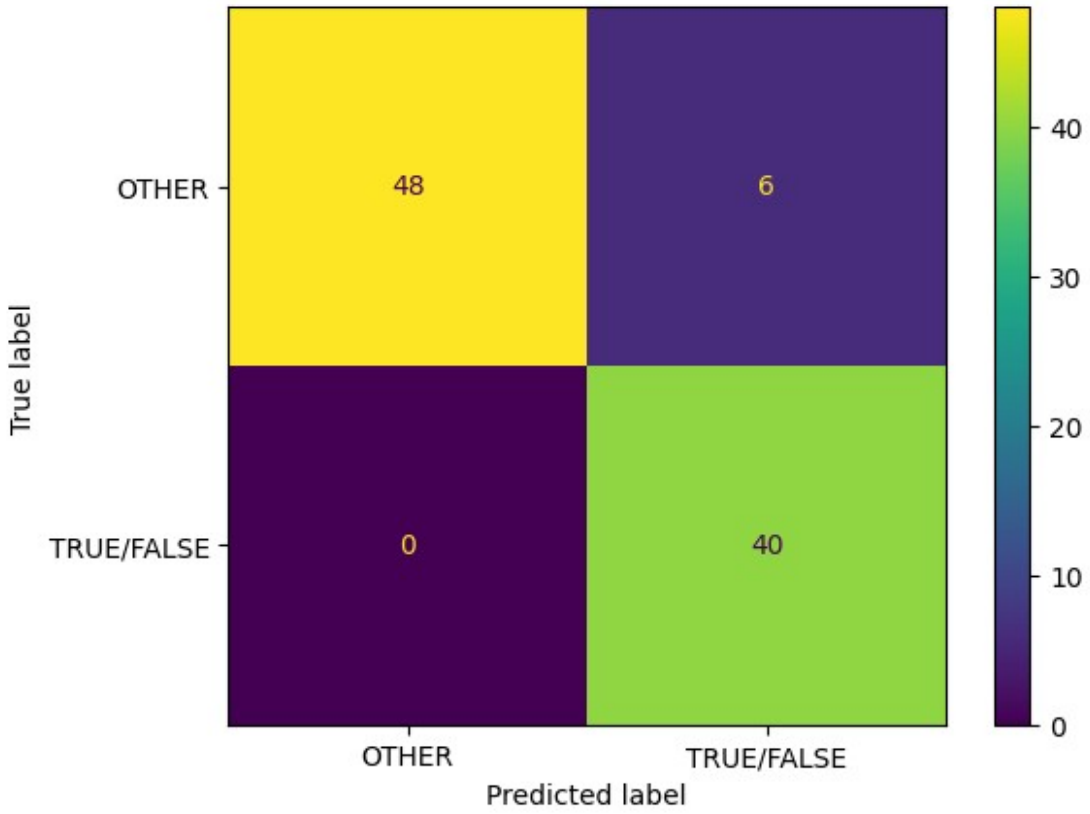
```
Accuracy : 0.936
Classification Report
              precision    recall  f1-score   support

       OTHER    1.00000   0.88889   0.94118        54
  TRUE/FALSE    0.86957   1.00000   0.93023        40

    accuracy                        0.93617        94
   macro avg    0.93478   0.94444   0.93570        94
weighted avg    0.94450   0.93617   0.93652        94

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 0.845

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

Accuracy : 0.936
Classification Report
              precision    recall  f1-score   support

       OTHER    0.96154   0.92593   0.94340        54
  TRUE/FALSE    0.90476   0.95000   0.92683        40

    accuracy                        0.93617        94
   macro avg    0.93315   0.93796   0.93511        94
weighted avg    0.93738   0.93617   0.93635        94

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 0.850

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

Accuracy : 0.936
Classification Report
              precision    recall  f1-score   support

       OTHER    1.00000   0.88889   0.94118        54
  TRUE/FALSE    0.86957   1.00000   0.93023        40
```

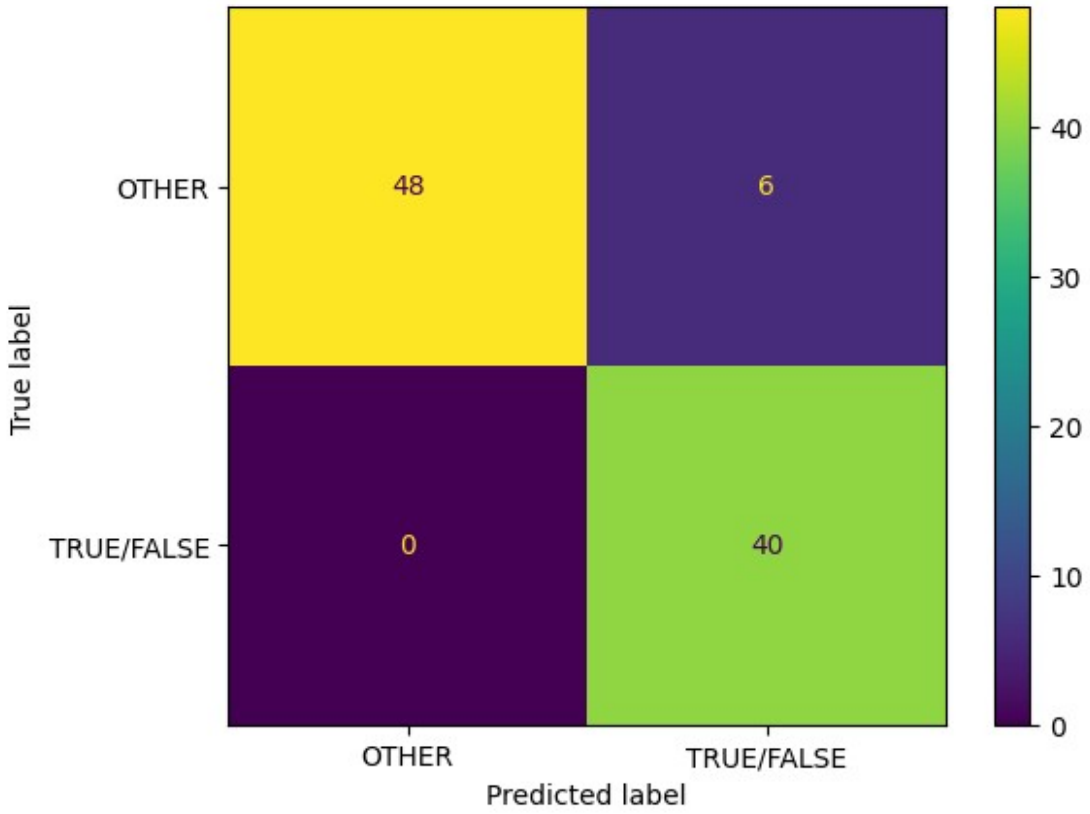
```
    accuracy                           0.93617        94
   macro avg    0.93478   0.94444   0.93570        94
weighted avg    0.94450   0.93617   0.93652        94

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 0.880

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

Accuracy : 0.936
Classification Report
              precision    recall  f1-score   support

       OTHER    1.00000   0.88889   0.94118        54
  TRUE/FALSE    0.86957   1.00000   0.93023        40

    accuracy                           0.93617        94
   macro avg    0.93478   0.94444   0.93570        94
weighted avg    0.94450   0.93617   0.93652        94

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 0.864

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

Accuracy : 0.915
Classification Report
              precision    recall  f1-score   support

       OTHER    0.96000   0.88889   0.92308        54
  TRUE/FALSE    0.86364   0.95000   0.90476        40

    accuracy                           0.91489        94
   macro avg    0.91182   0.91944   0.91392        94
weighted avg    0.91899   0.91489   0.91528        94

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

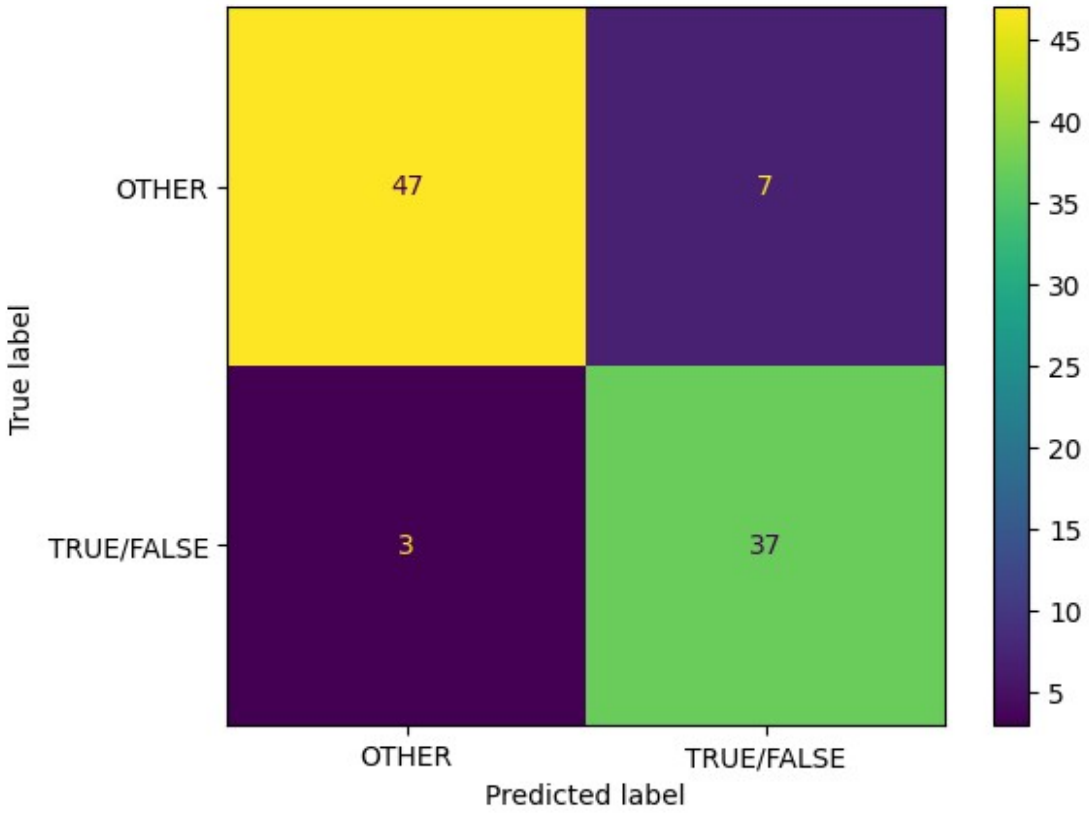
```
grid search fait
Fitting 5 folds for each of 8 candidates, totalling 40 fits
meilleur score 0.864

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

Accuracy : 0.926
Classification Report
              precision    recall  f1-score   support

       OTHER    0.97959   0.88889   0.93204        54
  TRUE/FALSE    0.86667   0.97500   0.91765        40

    accuracy                        0.92553        94
   macro avg    0.92313   0.93194   0.92484        94
weighted avg    0.93154   0.92553   0.92591        94

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 0.896

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

Accuracy : 0.936
Classification Report
              precision    recall  f1-score   support

       OTHER    1.00000   0.88889   0.94118        54
  TRUE/FALSE    0.86957   1.00000   0.93023        40

    accuracy                        0.93617        94
   macro avg    0.93478   0.94444   0.93570        94
weighted avg    0.94450   0.93617   0.93652        94

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

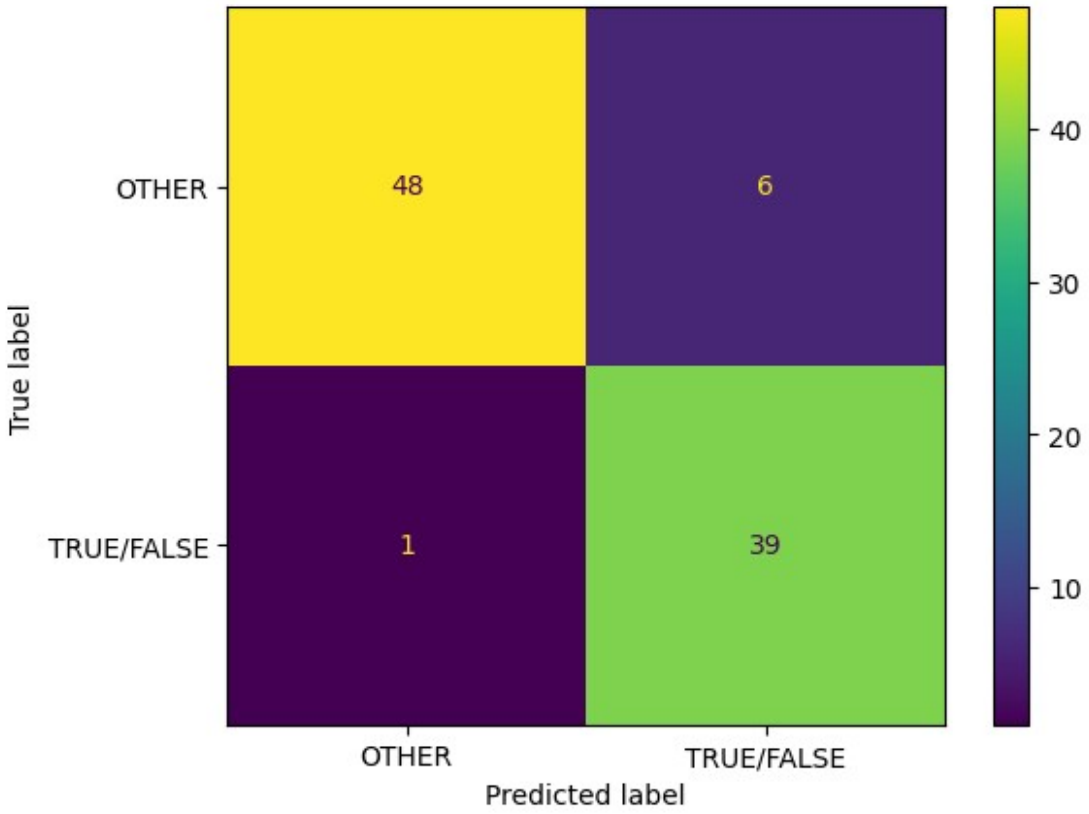

OTHER
TRUE/FALSE
True label
48
6
0
40
OTHER
TRUE/FALSE
Predicted label
40
30
20
10
0

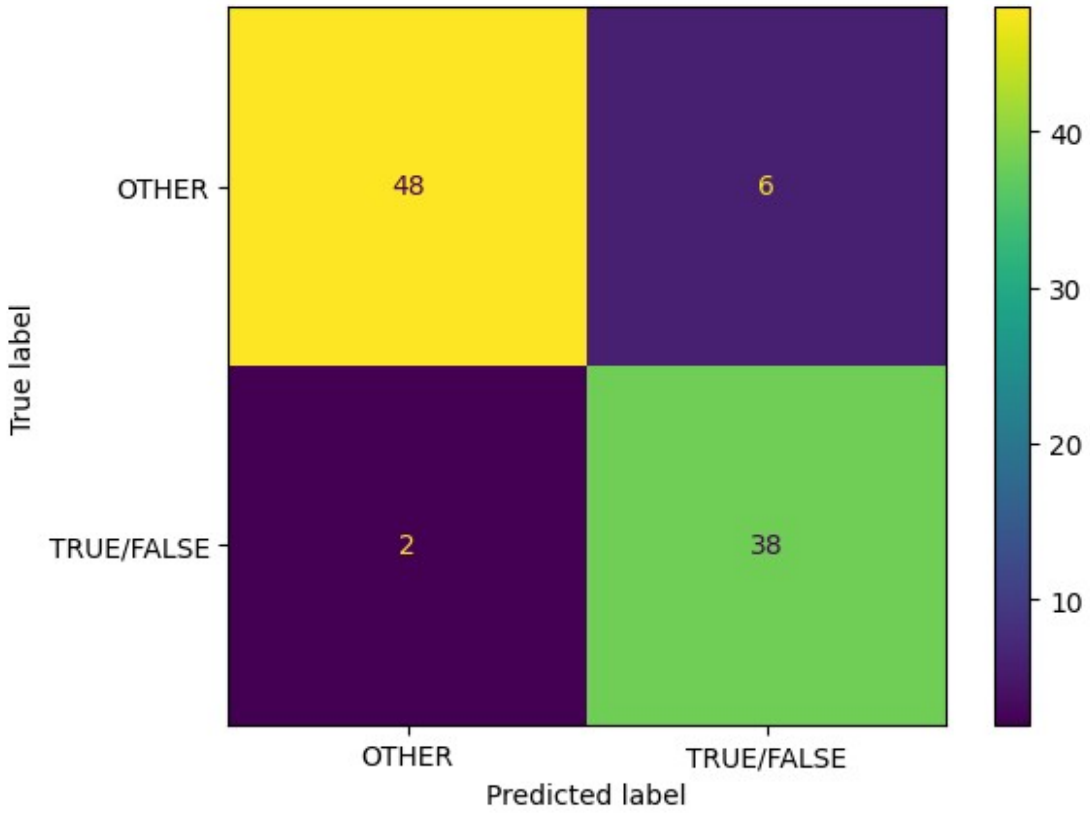

OTHER
TRUE/FALSE
True label
48
6
0
40
OTHER
TRUE/FALSE
Predicted label
40
30
20
10
0

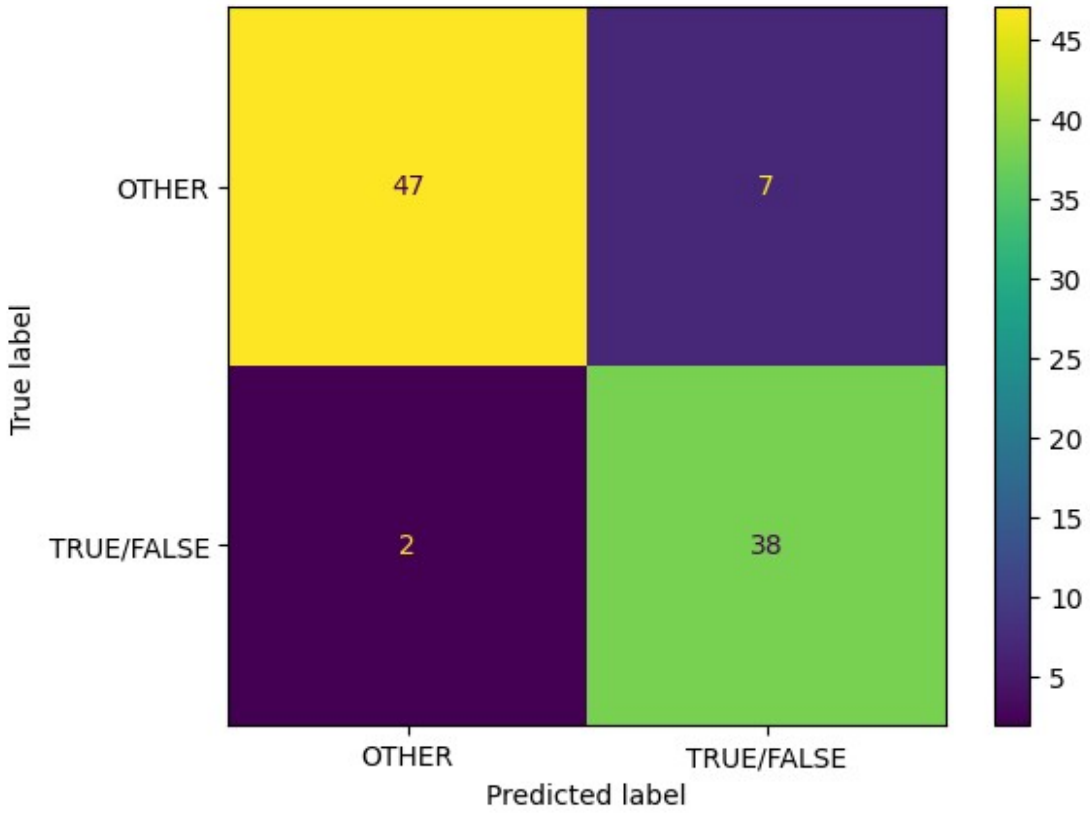

OTHER
TRUE/FALSE
True label
48
6
0
40
OTHER
TRUE/FALSE
Predicted label
40
30
20
10
0

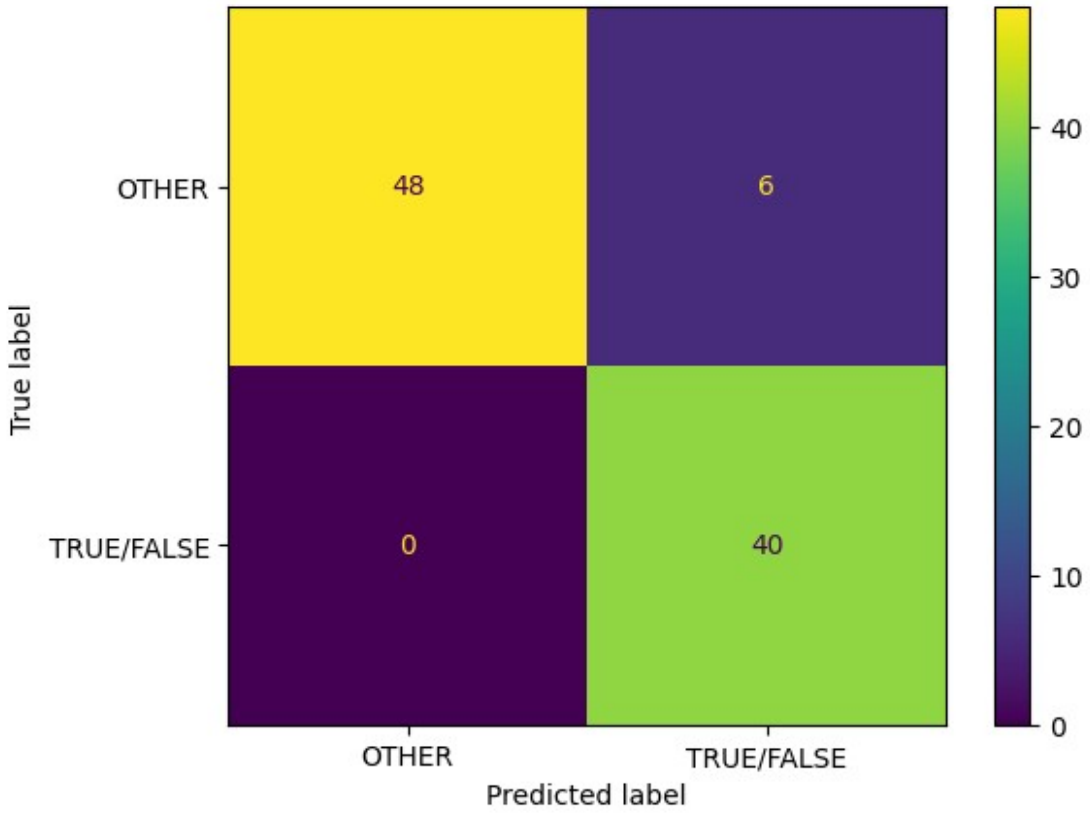

OTHER
TRUE/FALSE
True label
48
6
0
40
OTHER
TRUE/FALSE
Predicted label
40
30
20
10
0

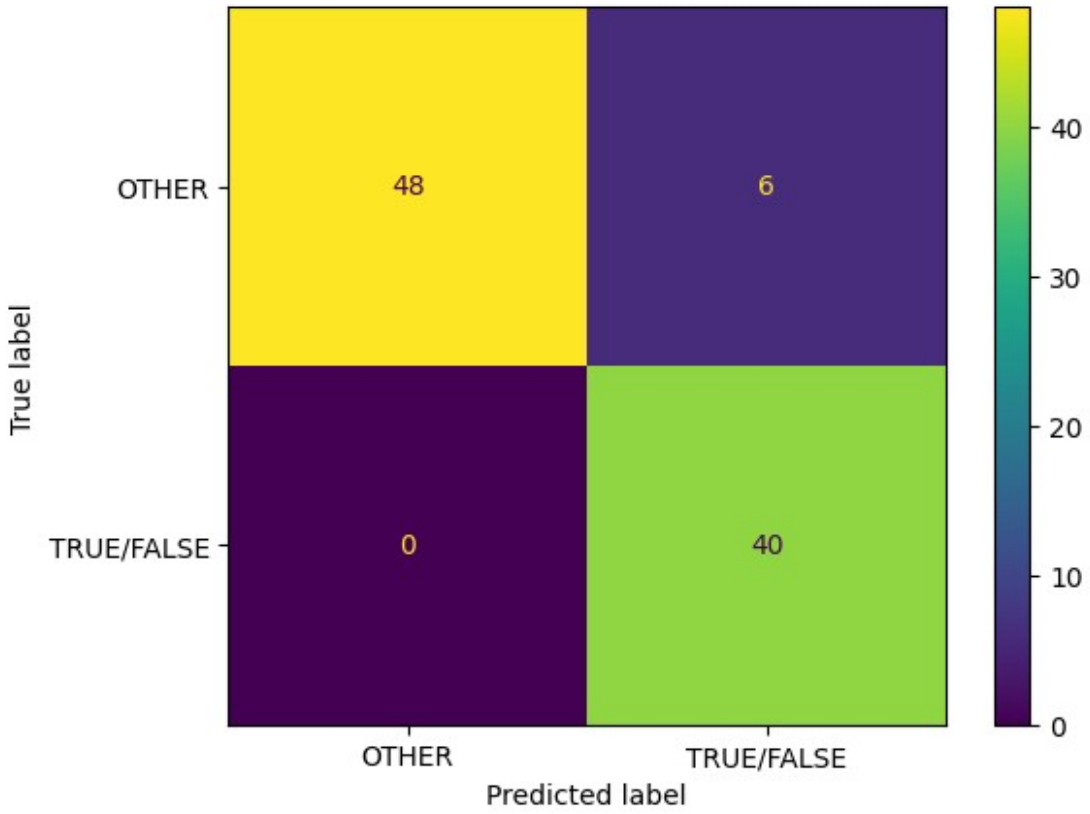

OTHER
TRUE/FALSE
True label
47
7
3
37
OTHER
TRUE/FALSE
Predicted label
45
40
35
30
25
20
15
10
5

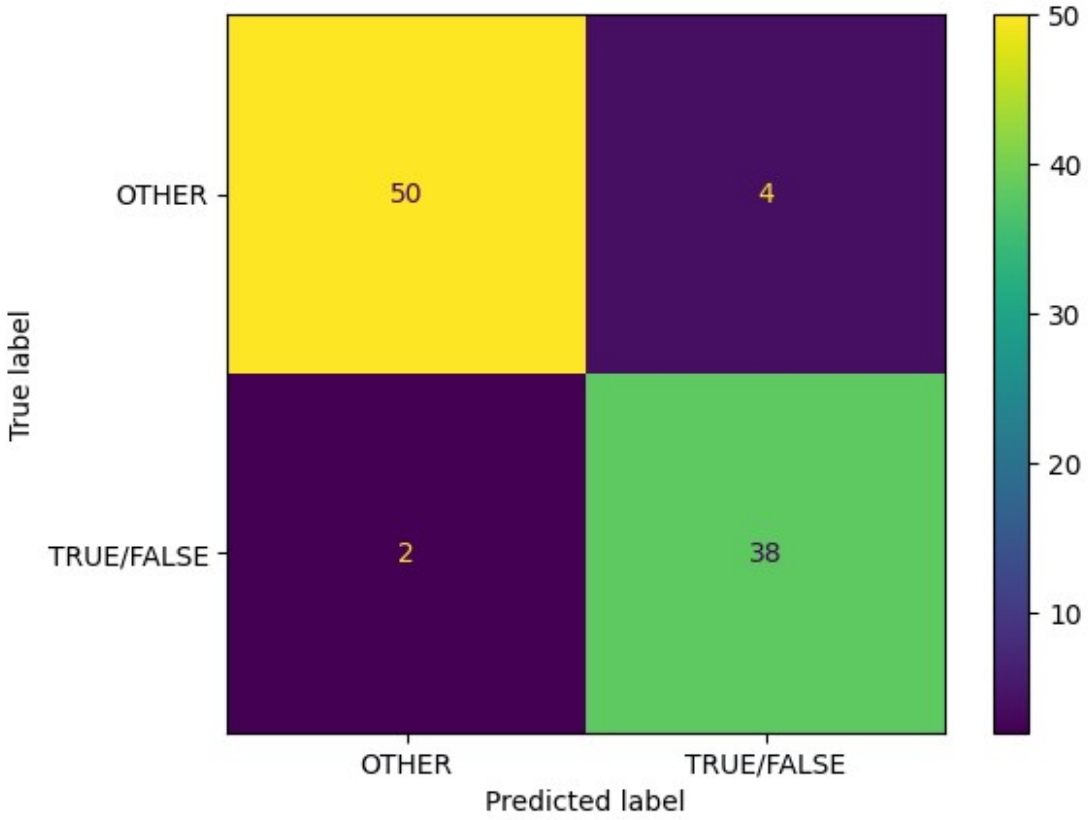

OTHER
TRUE/FALSE
True label
48
6
1
39
OTHER
TRUE/FALSE
Predicted label
40
30
20
10

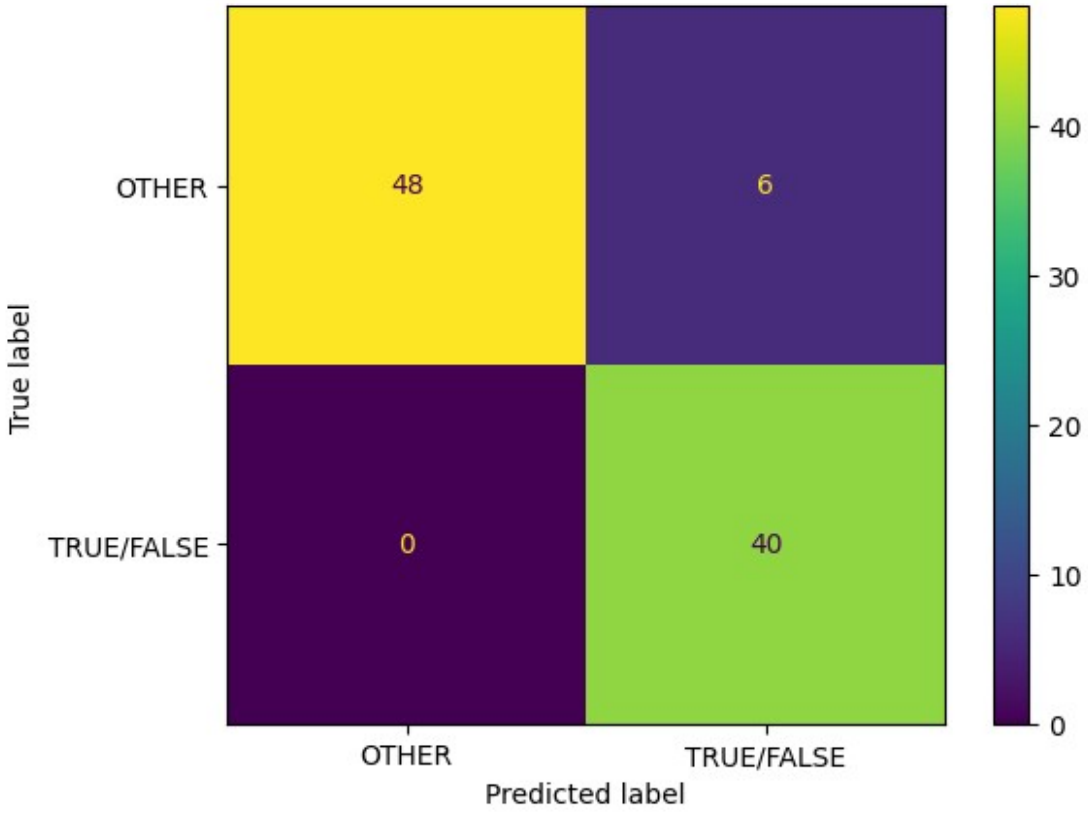

OTHER
TRUE/FALSE
True label
48
6
2
38
OTHER
TRUE/FALSE
Predicted label
40
30
20
10

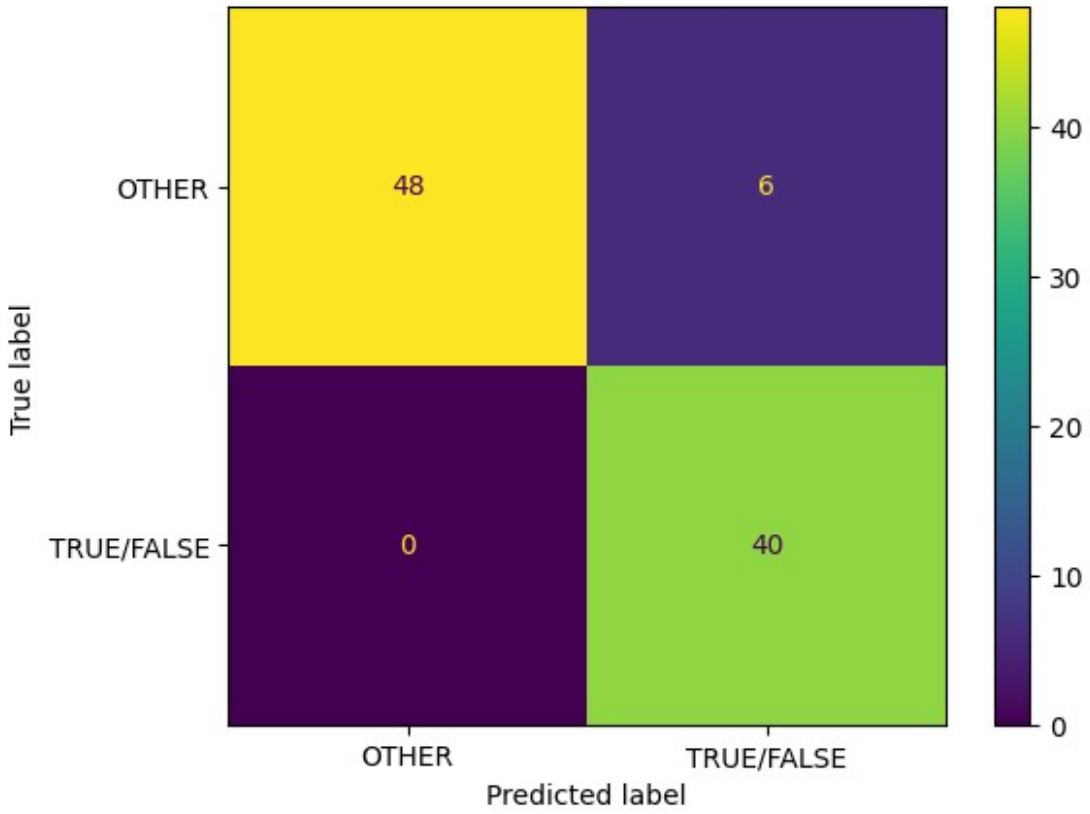

OTHER
TRUE/FALSE
True label
47
7
2
38
OTHER
TRUE/FALSE
Predicted label
45
40
35
30
25
20
15
10
5

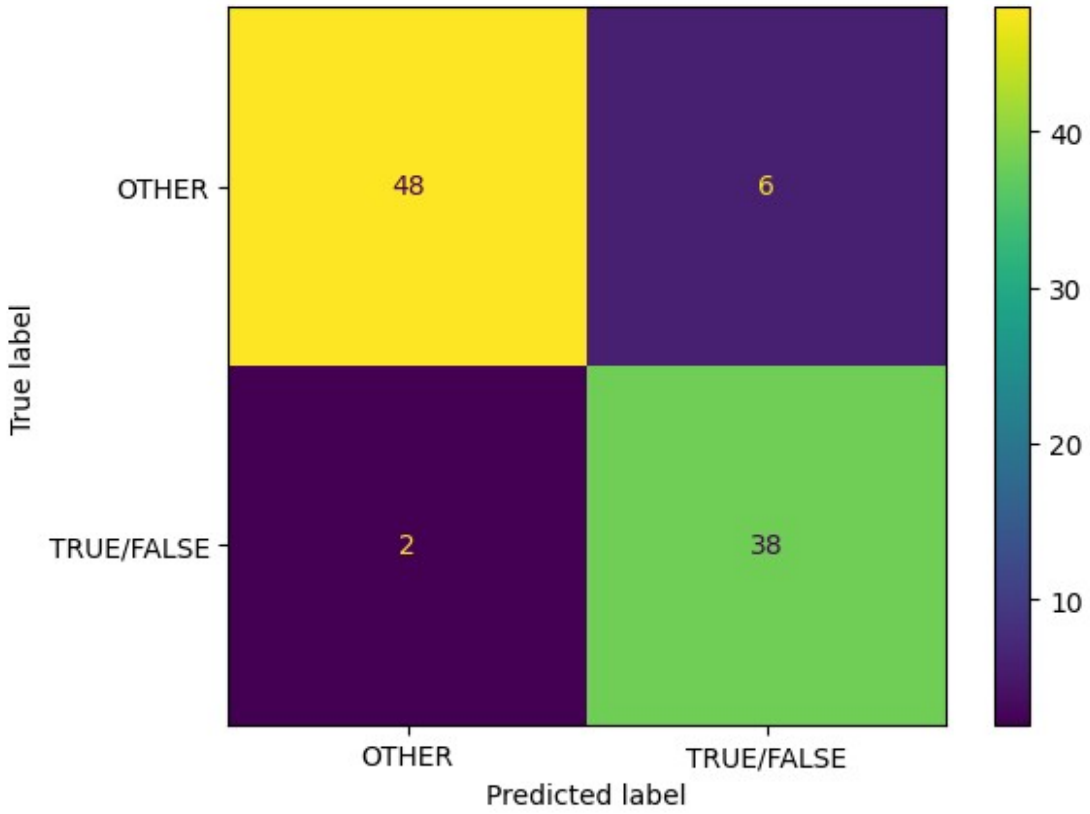

OTHER
TRUE/FALSE
True label
48
6
0
40
OTHER
TRUE/FALSE
Predicted label
40
30
20
10
0

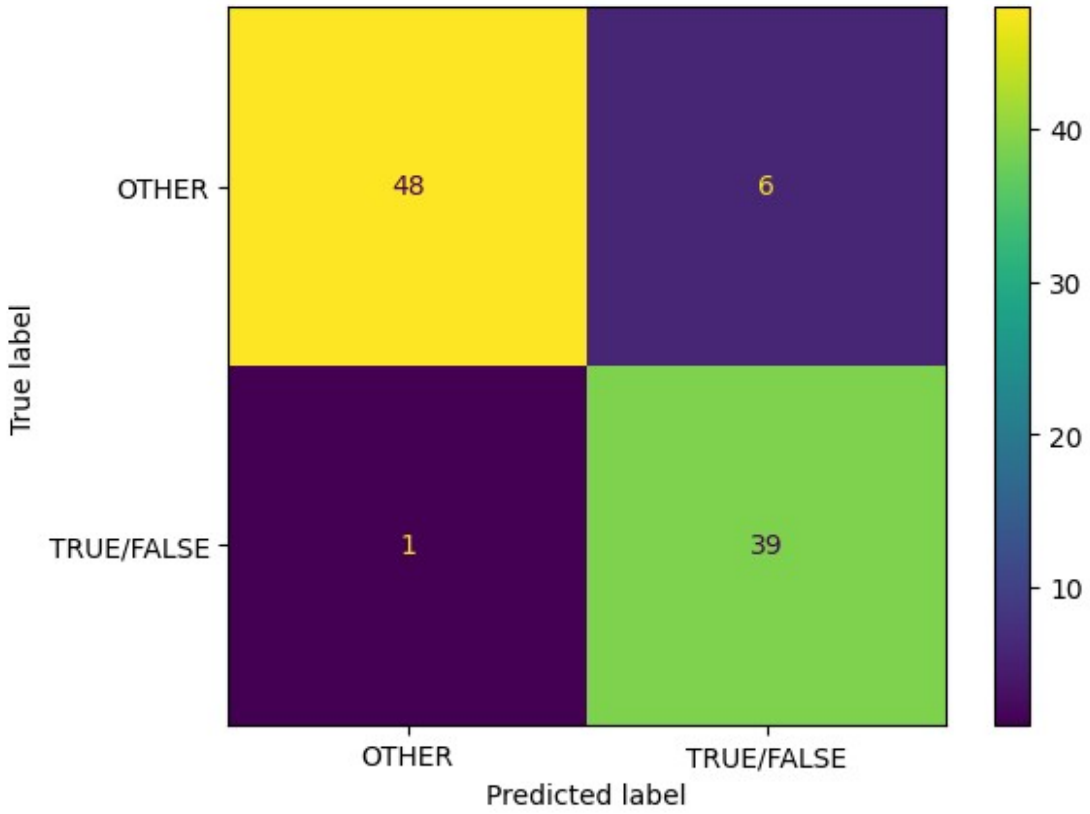

OTHER
TRUE/FALSE
True label
48
6
0
40
OTHER
TRUE/FALSE
Predicted label
40
30
20
10
0

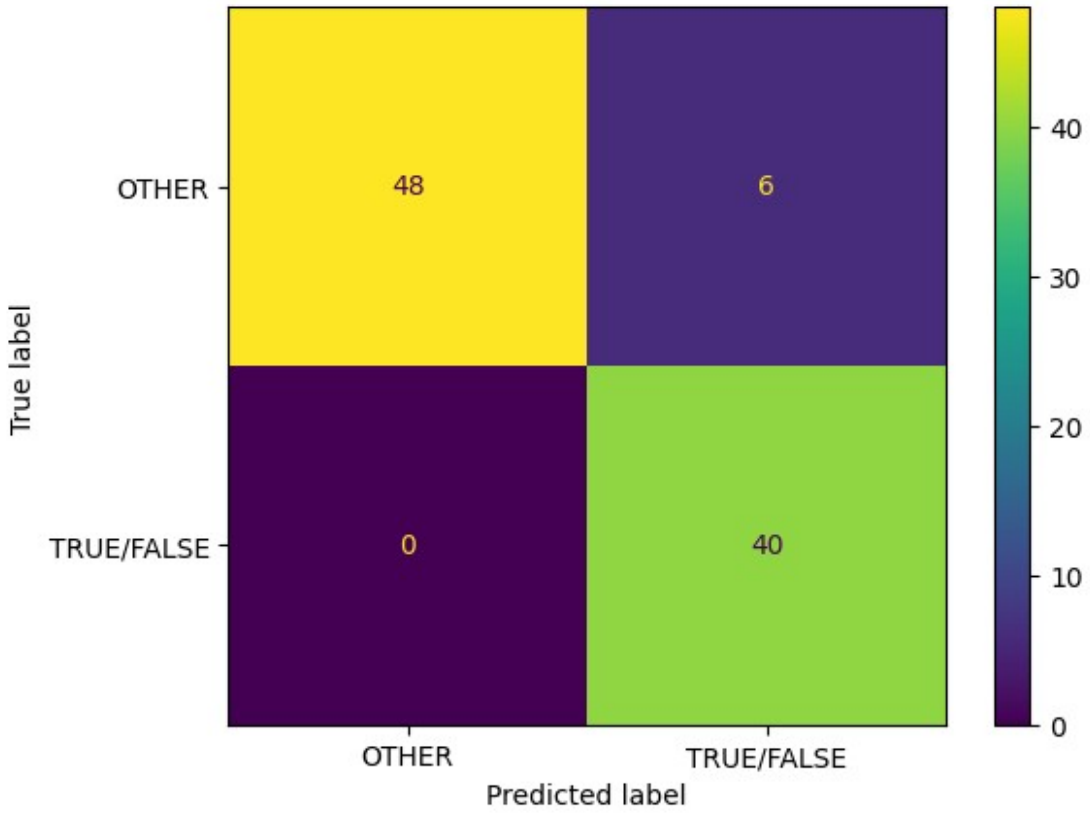

OTHER
TRUE/FALSE
True label
50
4
2
38
OTHER
TRUE/FALSE
Predicted label
50
40
30
20
10


OTHER
TRUE/FALSE
True label
48
6
0
40
OTHER
TRUE/FALSE
Predicted label
40
30
20
10
0


OTHER
TRUE/FALSE
True label
48
6
0
40
OTHER
TRUE/FALSE
Predicted label
40
30
20
10
0


OTHER
TRUE/FALSE
True label
48
6
2
38
OTHER
TRUE/FALSE
Predicted label
40
30
20
10


OTHER
TRUE/FALSE
True label
48
6
1
39
OTHER
TRUE/FALSE
Predicted label
40
30
20
10


OTHER
TRUE/FALSE
True label
48
6
0
40
OTHER
TRUE/FALSE
Predicted label
40
30
20
10
0