Importation des bibliothèques

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import string

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split, cross_val_predict
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score, matthews_corrcoef, precision_score, recall_score, roc_curve, auc
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk import pos_tag
import nltk

from tqdm import tqdm
```

In [2]:
```
# Charger les données
data_fake = pd.read_csv('Fake.csv')
data_real = pd.read_csv('True.csv')
```

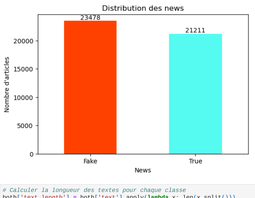
In [3]:
```
# Ajouter une colonne 'classement' pour identifier la véracité des textes
data_fake['classement'] = 0
data_real['classement'] = 1
```

In [4]:
```
# Fusionner les deux ensembles de données en un + nettoyage
both = pd.concat([data_fake, data_real], axis=0).dropna().drop_duplicates()
```

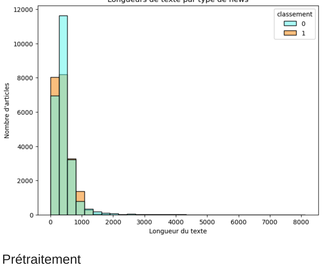
In [5]:
```
# Mélange des données
both = both.sample(frac=1).reset_index(drop=True)
```

Affichage d'un graphique montrant le nombre d'articles

In [6]:
```
# Afficher la distribution des classes
plt.figure(figsize=(6, 4))
class_distribution = both['classement'].value_counts()
bars = class_distribution.plot(kind='bar', color=['#FF4101', '#59FBF4'])
plt.title('Distribution des news')
plt.xlabel('News')
plt.ylabel('Nombre d\'articles')
plt.xticks(ticks=[0, 1], labels=['Fake', 'True'], rotation=0)

# Ajouter des étiquettes avec les valeurs exactes
for i, value in enumerate(class_distribution):
    plt.text(i, value + 20, str(value), ha='center', va='bottom')

plt.show()
```



In [7]:
```
# Calculer la longueur des textes pour chaque classe
both['text_length'] = both['text'].apply(lambda x: len(x.split()))
#print(both)

# Afficher l'histogramme des longueurs de texte pour chaque classe
plt.figure(figsize=(8, 6))
sns.histplot(data=both, x='text_length', hue='classement', bins=30, palette=['#59FBF4', '#FE8001'])
plt.title('Longueurs de texte par type de news')
plt.xlabel('Longueur du texte')
plt.ylabel('Nombre d\'articles')
plt.show()
```



Prétraitement

In [8]:
```
pos_tag_to_pos = {'J':'a', 'V':'v', 'N':'n', 'R':'r', 'I': 'n'} # trouver ce que veut dire le parametre de pos_tag

def pos_tag_to_pos_lemmas(tag):
    return pos_tag_to_pos.get(tag[0], 'n')

def preprocess_text(text):
    text = text.lower() # Convertir le texte en minuscules


    text = re.sub(r'[^a-zA-Z0-9\s]', '', text) # Supprimer les caractères spéciaux et les espaces supplémentaires
    text = re.sub(r'\s+', ' ', text) # remplacer plusieurs espaces/ par un seul espace
    text = re.sub(r'https?://\S+|www\.\S+', '', text) #supp les liens
    text = re.sub(r'\n', ' ', text)  # Remplacer les sauts de ligne par un espace
    #text = re.sub(r'<.??>+', '', text)  # Supprimer les balises HTML
    text = re.sub(r'\[.??\]', '', text)  # Supprimer les expressions entre crochets
    text = re.sub(r'\{.??\}', '', text)


    # Tokenization
    tokens = word_tokenize(text)
    tokens_and_pos = pos_tag(tokens)

    # Supprimer les mots vides
    stop_words = set(stopwords.words('english'))
    tokens_and_pos = [word for word in tokens_and_pos if word[0] not in stop_words]

    # Lemmatization (stemming)
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word[0], pos=pos_tag_to_pos_lemmas(word[1])) for word in tokens_and_pos]

    # Rejoindre les tokens en une seule chaine de texte
    preprocessed_text = ' '.join(tokens)

    return preprocessed_text
```

In [9]:
```
#print(preprocess_text("this is a text tha we wanted or want to lemmatize for a better experience of machine learning"))
```

In [10]:
```
# Prétraitement du texte

tqdm.pandas()
both['text'] = both['text'].progress_apply(preprocess_text)
```

100%|▰▰▰