Fake News detection (Vrai and Faux) vs Mix

May 6, 2023

1 Projet final de Machine Learning 1 HAI817I

Ce projet s'inscrit dans le contexte de l'apprentissage supervisé, i.e. les données possèdent des labels. Il vise à trouver les modèles les plus performants pour prédire si des articles de presse sont vrais ou faux. Les articles contiennent des assertions (une assertion est une proposition que l'on avance et que l'on soutient comme vraie) faites, par exemple, par des hommes politiques

[75]:
```
# ! pip install langdetect
# !pip install contractions
# ! pip install wordcloud
```

1.1 Importation des bibliotheques et modules necessaires

[2]:
```
# 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

# librairies générales
import pickle
import pandas as pd
from scipy.stats import randint
import numpy as np
import string
import time
import base64
import re
import sys
import contractions
```

```
# librairie BeautifulSoup
from bs4 import BeautifulSoup
# librairie affichage



## detection de language
import langdetect
import wordcloud
import nltk

from nltk import sent_tokenize
from nltk import RegexpParser
from nltk import pos_tag
from nltk import word_tokenize


from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
# Lemmatisateur
from nltk.stem import WordNetLemmatizer
# Racinisateur
from nltk.stem.porter import PorterStemmer
# TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer

import spacy
from spacy.tokens import Span
from spacy.lang.en import English


from sklearn.base import BaseEstimator
from sklearn.base import TransformerMixin


# Modeles
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
```

```
# Metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_curve
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_recall_fscore_support as score
from tabulate import tabulate




# il est possible de charger l'ensemble des librairies en une seule fois
# décocher le commentaire de la ligne ci-dessous
#nltk.download('all')
# nltk.download('punkt')
# nltk.download('averaged_perceptron_tagger')
# nltk.download('tagsets')
# nltk.download("stopwords")
# nltk.download('wordnet')
```

[3]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import sys

import sklearn

# import some classifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import PassiveAggressiveClassifier

# import modules for vectorizing and pipe
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn import preprocessing
from sklearn.feature_extraction.text import TfidfTransformer
```

```
from sklearn.pipeline import make_pipeline

# scale data
from sklearn.preprocessing import StandardScaler

# upsampling downsampling
from sklearn.utils import resample

# modules for model selection
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score

# modules for evaluation
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import precision_recall_fscore_support as score

# modules for vizualisation
import seaborn as sns
import matplotlib.pyplot as plt
from tabulate import tabulate

# others
import itertools
import random

from sklearn.exceptions import ConvergenceWarning



# import some classifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import PassiveAggressiveClassifier

from sklearn.pipeline import Pipeline
from sklearn import preprocessing
```

```
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.pipeline import make_pipeline
```

[4]:
```
# Dataset initial
train_df_init = pd.read_csv('HAI817_Project_data/HAI817_Projet_train.csv')
test_df_init = pd.read_csv('HAI817_Project_data/HAI817_Projet_test.csv')

# Dataset recent
train_df_latest = pd.read_csv('HAI817_Project_data/HAI817_Project_data/
 ↪HAI817_Projet_train.csv')
test_df_latest = pd.read_csv('HAI817_Project_data/HAI817_Project_data/
 ↪HAI817_Projet_train.csv')

train_df_init = train_df_init.fillna(' ')
test_df_init = test_df_init.fillna(' ')
```

[5]:
```
# Initial
train_df_init.describe()

#␣
 ↪print("----------------------------------------------------------------------------------------

# Recent
test_df_latest.describe()
```

[5]:
```
       public_id                                                text
count       1264                                                1264  \
unique      1115                                                1086
top     cd9cd5e8  The late Robin Williams once called cocaine "G…
freq           2                                                   4

                        title our rating
count                    1241       1264
unique                   1070          4
top     - The Washington Post      false
freq                        4        578
```

[76]:
```
# Un petit echantillon de 15 lignes pour commencer
# sample = train_df_init.sample(15)
```

[77]:
```
# sample
```

[74]:
```
# sample.info()
# sample = sample.values.tolist()
# sample
```

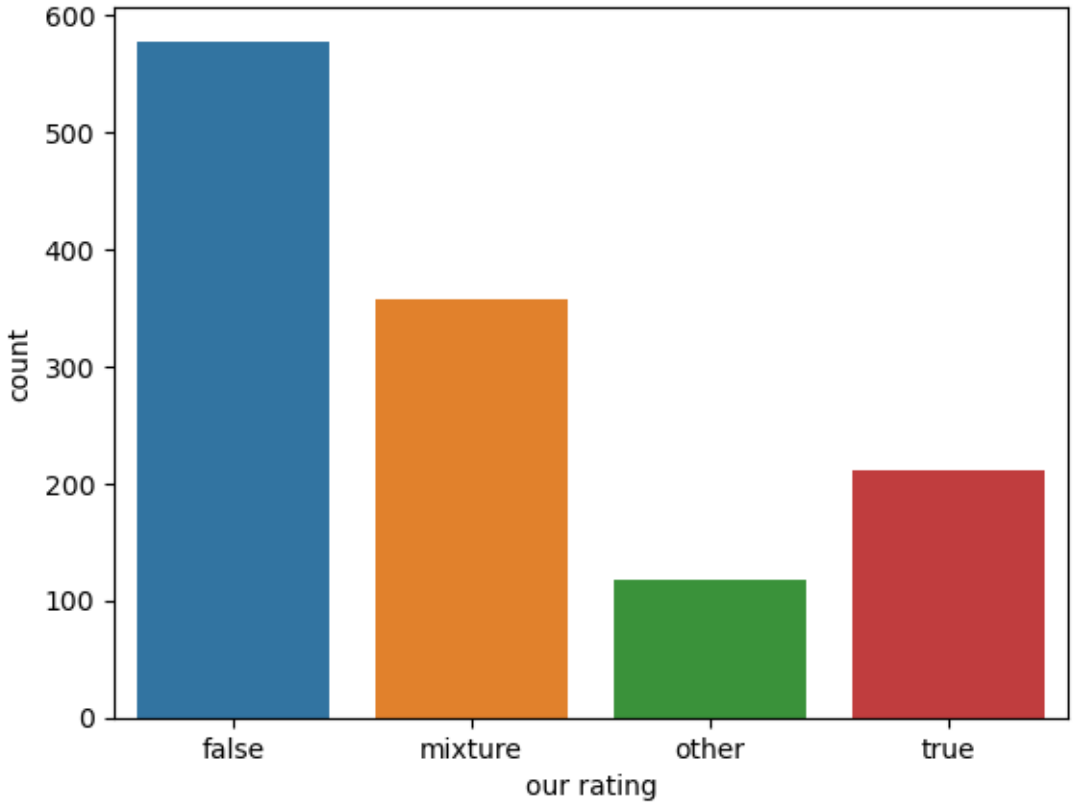
[9]:
```
# sample
```

[10]:
```
# For the content
# for record in sample:
#     text = record[1]
#     title = record[2]
# #     display(text)
# #     wc = wordcloud.WordCloud(background_color='black', max_font_size=30,␣
 ↪max_words=100)
# #     text_wc = wc.generate(str(text))
# #     title_wc = wc.generate(str(title))
# #     print("Text size: ",len(text), " Title size: ", len(title), end="\n")
#     text_wc = wordcloud.WordCloud(background_color='black', max_font_size=30,␣
 ↪max_words=100).generate_from_text(text=text)
#     title_wc = wordcloud.WordCloud(background_color='black',␣
 ↪max_font_size=30, max_words=100).generate_from_text(text=title)
# #     fig = plt.figure(num=1)

#     # Create a figure with two subplots
#     fig, axs = plt.subplots(ncols=2, figsize=(15, 10) ) # figsize=(10, 5)
#     fig.suptitle(title)

#     # Plot the first wordcloud in the first subplot
#     axs[0].imshow(text_wc, cmap=None) # , interpolation='bilinear'
#     axs[0].set_title('Text')

#     # Plot the second wordcloud in the second subplot
#     axs[1].imshow(title_wc, cmap=None)
#     axs[1].set_title('Title')

#     # Remove axis ticks and labels
#     for ax in axs:
#         ax.set_xticks([])
#         ax.set_yticks([])
#         ax.set_xticklabels([])
#         ax.set_yticklabels([])

# #     plt.axis('off')
# #     plt.imshow(text_wc, title_wc,  cmap=None)
#     plt.show()
# #     display(record)
```

1.2 !Apparemment le texte et titre sont mal positionnes dans certains row et certains rows ont un texte de taille presqu'egale a celle du titre, Est-ce du bruit?

[11]:
```
sns.countplot(x='our rating', data=train_df_init)
```

[11]: <AxesSubplot:xlabel='our rating', ylabel='count'>



2 Ingenierie des donnees

2.0.1 Renommons le rating

[12]:
```
train_df_init = train_df_init.rename(columns={"our rating": "rating"})
```

[13]:
```
train_df_init.isnull().sum()
```

[13]:
```
public_id    0
text         0
title        0
rating       0
dtype: int64
```

[14]:
```
train_df_init.shape[0]
```

[14]: 1264

[15]:
```
train_df_init.rating
```

[15]:
```
0         false
1       mixture
2       mixture
3         false
4         false
          …
1259       true
1260       true
1261      false
1262       true
1263       true
Name: rating, Length: 1264, dtype: object
```

2.1 Elements utilitaires du cours

TextCleaner

[16]:
```
nltk.download('wordnet')
nltk.download('stopwords')
nltk.download('punkt')
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 # lematisation des termes
               ):

    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
```

```
[nltk_data] Downloading package wordnet to /home/richard/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/richard/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /home/richard/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
```

TextNormailizer

[17]:
```
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
```

TrueFalseMixer

[18]:
```
def TrueFalseMixer(x):

    if x=="true" or x=="false":
        return "true_or_false"
    else:
        return "mixture"
```

[19]:
```
# train_df_init = pd.read_csv('HAI817_Project_data/HAI817_Projet_train.csv')
# train_df_init = train_df_init.rename(columns={"our rating": "rating"})
# test_df_init = pd.read_csv("HAI817_Project_data/HAI817_Projet_test.csv")
# test_df_init = test_df_init.rename(columns={"our rating": "rating"})
```

```
# test_df_init.head()
```

[20]:
```
train_df_init = pd.read_csv('HAI817_Project_data/HAI817_Projet_train.csv')
train_df_init = train_df_init.rename(columns={"our rating": "rating"})

test_df_init = pd.read_csv("HAI817_Project_data/HAI817_Projet_test.csv")
test_df_init = test_df_init.rename(columns={"our rating": "rating"})

train_df_init = train_df_init.fillna(' ')
test_df_init = test_df_init.fillna(' ')

train_df_init['text'] = train_df_init['title'] + " "+train_df_init['text']
test_df_init['text'] = test_df_init['title'] + " "+test_df_init['text']

test_df_init.head()
```

[20]:

	ID	text	title	rating
0	122653045997905671927713471889615536378	US Treasury deputy sec warns that shortages li…	US Treasury deputy sec warns that shortages li…	mixture
1	275389285957305997321446227088442471741	CNN Praises Taliban For Wearing Masks During A…	CNN Praises Taliban For Wearing Masks During A…	other
2	333248764296609831067233855420575814716	Tennessee Has Just LEGALIZED Government COVID …	Tennessee Has Just LEGALIZED Government COVID …	false
3	264019763253447756851916399533799891538	MEDICAL SHOCKER: Scientists at Sloan Kettering…	MEDICAL SHOCKER: Scientists at Sloan Kettering…	false
4	158073737187690682830899773280916034317	Study Results: Facemasks are Ineffective to Bl…	Study Results: Facemasks are Ineffective to Bl…	false

Resampler

[21]:
```
def Resampler( df, classif_type=0):
    smallest_size = 0
    smallest = ""
     # true false other mixture
    df["rating"].value_counts().plot(kind='pie',
                                     autopct='%1.1f%%',
                                     label = "Classe",
```

```
                                    title='Data overview',
                                    fontsize=11,
                                    legend=True)
    plt.show()

    if classif_type==0:
        """
        resampling pour Vrai vs Faux
        """
        df
        falses = df[df["rating"] == "false"]
        trues = df[df["rating"] == "true"]


        # find smallest between trues and falses and dropping other types
        smallest = "true" if trues.shape[0] <= falses.shape[0]   else "false"
        smallest_size = trues.shape[0] if trues.shape[0] <= falses.shape[0]␣
 ↪else falses.shape[0]

        df = pd.concat([falses, trues]) # df[df["rating"] == smallest]

        df["rating"].value_counts().plot(kind='pie',
                                         autopct='%1.1f%%',
                                         label = "Classe",
                                         title='Before',
                                         fontsize=11,
                                         legend=True)
        plt.show()
#         print("True: ", trues.shape[0]," False: ", falses.shape[0], "Smallest:
 ↪ ", smallest, " Size:", smallest_size)
        df = df[df["rating"] == smallest]
        if smallest == "true":
            df = pd.concat([df, falses.sample(smallest_size)])
        else:
            df = pd.concat([df, trues.sample(smallest_size)]) # df.append(trues.
 ↪sample(smallest_size))


    if classif_type==1:
        """
        resampling pour (Vrai, Faux) vs mixture
        """
        df["rating"] = df["rating"].apply(TrueFalseMixer)
        df["rating"].value_counts().plot(kind='pie',
                                         autopct='%1.1f%%',
                                         label = "Classe",
                                         title='Before',
```

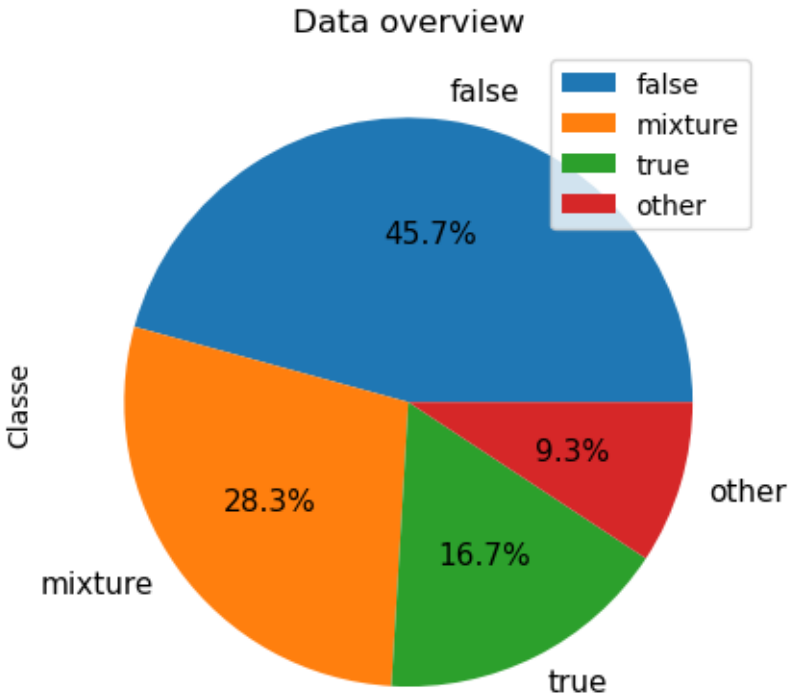
```
                                        fontsize=11,
                                        legend=True)
        plt.show()
        true_and_false = df[df["rating"] == "true_or_false"]
        mixtures = df[df["rating"] == "mixture"]


        # find smallest between trues and falses and dropping other types
        smallest = "true_or_false" if true_and_false.shape[0] <= mixtures.
 ↪shape[0]   else "mixture"
        smallest_size = true_and_false.shape[0] if true_and_false.shape[0] <=␣
 ↪mixtures.shape[0] else mixtures.shape[0]
#         print("True fs: ", true_and_false.shape[0]," Mix: ", mixtures.
 ↪shape[0], "Smallest: ", smallest, " Size:", smallest_size)

#         display(df)
        df = df[df["rating"] == smallest]


        if  "true_or_false" in smallest:
            df = pd.concat([df, mixtures.sample(smallest_size)])
        else:
            df = pd.concat([df, true_and_false.sample(smallest_size)])



    if classif_type==2:
        """
        resampling pour Vrai vs Faux
        """
        # true false other mixture
        falses = df[df["rating"] == "false"]
        trues = df[df["rating"] == "true"]
        others = df[df["rating"] == "other"]
        mixtures = df[df["rating"] == "mixture"]



        classes_dict = {
            "true": trues.shape[0],
            "false": falses.shape[0],
             "other": others.shape[0],
            "mixture": mixtures.shape[0],
        }

        # find smallest between trues and falses and dropping other types
        smallest_size = list(classes_dict.values())[0]
```

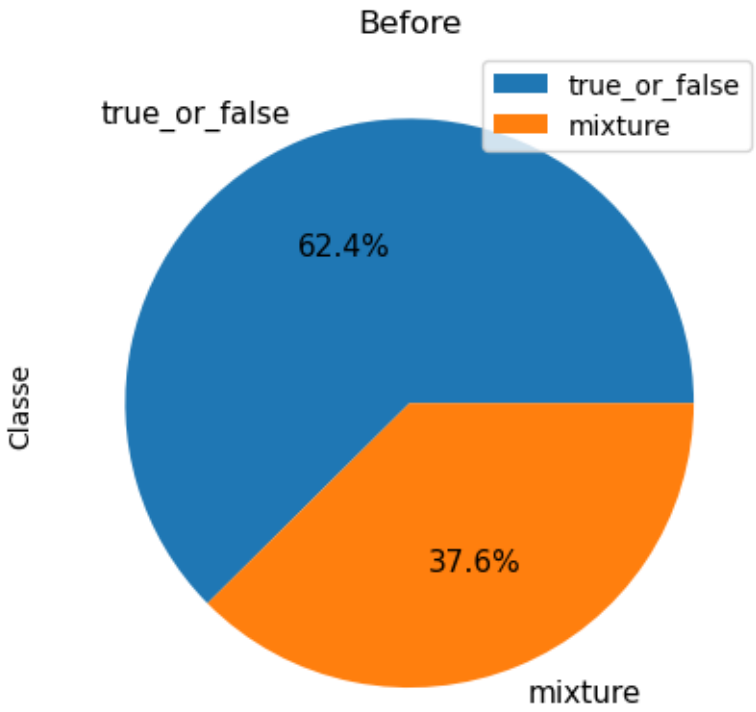
```
        smallest = list(classes_dict.keys())[0]

        for key, val in classes_dict.items():
            if val <= smallest_size:
                smallest_size = val
                smallest = key


        df = df[df["rating"] == smallest]

        if smallest == "true":
            df = pd.concat([df,
                            falses.sample(smallest_size),
                            others.sample(smallest_size),
                            mixtures.sample(smallest_size)])

        elif smallest == "false":
            df = pd.concat([df,
                            true.sample(smallest_size),
                            others.sample(smallest_size),
                            mixtures.sample(smallest_size)])

        elif smallest == "other":
            df = pd.concat([df,
                            trues.sample(smallest_size),
                            falses.sample(smallest_size),
                            mixtures.sample(smallest_size)])

        else:
            df = pd.concat([df,
                            trues.sample(smallest_size),
                            falses.sample(smallest_size),
                            others.sample(smallest_size)])


    # Shuffle the df to make data random
    df = df.sample(frac=1)
    df["rating"].value_counts().plot(kind='pie',
                                     autopct='%1.1f%%',
                                     label = "Classe",
                                     title='Resampled',
                                     fontsize=11,
                                     legend=True)
    plt.show()
    return df
```

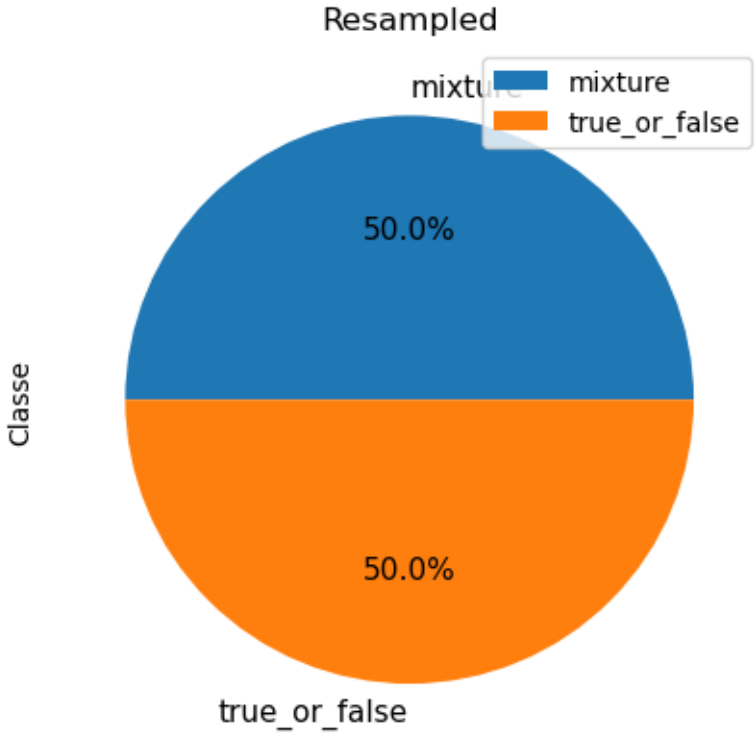
2.1.1 DataCleanerForClassification

[22]:
```
def DataCleanerForClassification( df, classif_type=0):

#     df['text'] = df['title'] + " "+df['text']

    if classif_type==1:
        df["rating"] = df["rating"].apply(TrueFalseMixer)


    df["rating"].value_counts().plot(kind='pie',
                                     autopct='%1.1f%%',
                                     label = "Classe",
                                     title='Data overview',
                                     fontsize=11,
                                     legend=True)
    plt.show()
    return df
```

[23]:
```
dft = Resampler(df=train_df_init, classif_type=1)
display(dft)
```




Before
true_or_false
mixture
62.4%
37.6%
Classe
mixture


Resampled
mixture
true_or_false
50.0%
50.0%
Classe
true_or_false

```
     public_id                                                text
907   863897d7  Dr. Didier Raoult published new results of 1,0…  \
918   054fefab  SUPREME COURT OF THE UNITED STATES No. 18A335 …
384   6f6addcc  Marine Corps. Rebukes Pelosi: "WE DON'T WORK F…
1155  c1af4bf0  When Will The Planet Be Too Hot For Humans? Mu…
1245  d0b0e459  Global Ocean Circulation Appears To Be Collaps…
…        …                                                  …
32    a329a5b3  SAD: Biden Introduces Granddaughter by Saying,…
794   f353ec98  Project G-2101: Pentagon biolab discovered MER…
3     f14e8eb6  Obama's Daughters Caught on Camera Burning US …
352   6612a10d  Deliveroo drivers demand union recognition and…
465   0a0604c8  FG bans alcohol in satchets, polythene Covid-1…

                                                    title         rating
907   Dr. Didier Raoult published new results of 1,0…        mixture
918   SUPREME COURT OF THE UNITED STATES No. 18A335 …        mixture
384   Marine Corps. Rebukes Pelosi: "WE DON'T WORK F…  true_or_false
1155  When Will The Planet Be Too Hot For Humans? Mu…        mixture
1245  Global Ocean Circulation Appears To Be Collaps…  true_or_false
…                                                    …            …
32    SAD: Biden Introduces Granddaughter by Saying,…        mixture
794   Project G-2101: Pentagon biolab discovered MER…        mixture
3     Obama's Daughters Caught on Camera Burning US …  true_or_false
352   Deliveroo drivers demand union recognition and…        mixture
465              FG bans alcohol in satchets, polythene        mixture

[950 rows x 4 columns]
```

[24]:
```
def preprocess_selection(model_name,model,X,y,lowercase=False):
    CV_brut = Pipeline([('cleaner', TextNormalizer()),
                        ('count_vectorizer', CountVectorizer(lowercase=False)),
                        (model_name, model)])
    CV_lowcase = Pipeline([('cleaner',␣
 ↪TextNormalizer(removestopwords=False,lowercase=True,
                                                    ␣
 ↪getstemmer=False,removedigit=False)),
                        ('count_vectorizer',␣
 ↪CountVectorizer(lowercase=lowercase)),
                        (model_name, model)])
    CV_lowStop = Pipeline([('cleaner',␣
 ↪TextNormalizer(removestopwords=True,lowercase=True,
                                                    ␣
 ↪getstemmer=False,removedigit=False)),
```

```
                      ('count_vectorizer',␣
↪CountVectorizer(lowercase=lowercase)),
                      (model_name, model)])

  CV_lowStopstem = Pipeline([('cleaner',␣
↪TextNormalizer(removestopwords=True,lowercase=True,
                                                  ␣
↪getstemmer=True,removedigit=False)),
                      ('count_vectorizer',␣
↪CountVectorizer(lowercase=lowercase)),
                      (model_name, model)])

  CV_lowStopna = Pipeline([('cleaner',␣
↪TextNormalizer(removestopwords=True,lowercase=True,
                                                  ␣
↪getstemmer=True,removedigit=True)),
                      ('count_vectorizer',␣
↪CountVectorizer(lowercase=lowercase)),
                      (model_name, model)])

  TFIDF_brut = Pipeline ([('cleaner', TextNormalizer()),
                      ('tfidf_vectorizer',␣
↪TfidfVectorizer(lowercase=lowercase)),
                      (model_name, model)])

  TFIDF_lowcase = Pipeline([('cleaner',␣
↪TextNormalizer(removestopwords=False,lowercase=True,
                                                  ␣
↪getstemmer=False,removedigit=False)),
                      ('tfidf_vectorizer',␣
↪TfidfVectorizer(lowercase=lowercase)),
                      (model_name, model)])
  TFIDF_lowStop = Pipeline([('cleaner',␣
↪TextNormalizer(removestopwords=True,lowercase=True,
                                                  ␣
↪getstemmer=False,removedigit=False)),
                      ('tfidf_vectorizer',␣
↪TfidfVectorizer(lowercase=lowercase)),
                      (model_name, model)])

  TFIDF_lowStopstem = Pipeline([('cleaner',␣
↪TextNormalizer(removestopwords=True,lowercase=True,
                                                  ␣
↪getstemmer=True,removedigit=False)),
                      ('tfidf_vectorizer',␣
↪TfidfVectorizer(lowercase=lowercase)),
```

```
                        (model_name, model)])
    TFIDF_lowStopna = Pipeline([('cleaner',␣
↪TextNormalizer(removestopwords=True,lowercase=True,
                                                    ␣
↪getstemmer=True,removedigit=True)),
                        ('tfidf_vectorizer',␣
↪TfidfVectorizer(lowercase=lowercase)),
                        (model_name, model)])

    all_models = [
        ("CV_brut", CV_brut),
        ("CV_lowcase", CV_lowcase),
        ("CV_lowStop", CV_lowStop),
        ("CV_lowStopstem",CV_lowStopstem),
        ("CV_lowStopna",CV_lowStopna),
        ("TFIDF_lowStopna", TFIDF_lowStopna),
        ("TFIDF_lowcase", TFIDF_lowcase),
        ("TFIDF_lowStop", TFIDF_lowStop),
        ("TFIDF_lowStopstem",TFIDF_lowStopstem),
        ("TFIDF_brut", TFIDF_brut),
    ]

    print ("Evaluation des différentes configurations : ")
    unsorted_scores = [(name, cross_val_score(model, X, y, cv=10).mean()) for␣
↪name, model in all_models]
    scores = sorted(unsorted_scores, key=lambda x: -x[1])


    print(tabulate(scores, floatfmt='.4f', headers=('Pipeline', 'Score')))
```

[25]:
```
X = dft['title']+" "+dft['text']
# X = X.drop("rating", axis=1)
# display(X)
y = dft['rating']
y
```

[25]:
```
907          mixture
918          mixture
384     true_or_false
1155         mixture
1245    true_or_false
            …
32           mixture
794          mixture
3       true_or_false
352          mixture
```

```
465           mixture
Name: rating, Length: 950, dtype: object
```

[26]:
```
X_s = X # X.sample(200)
y_s = y.loc[X_s.index]
preprocess_selection("multinomial_nb",MultinomialNB(),X_s,y_s)
```

```
Evaluation des différentes configurations :
Pipeline             Score
-----------------  -------
TFIDF_brut          0.7105
TFIDF_lowStop       0.7084
CV_brut             0.7042
TFIDF_lowcase       0.7011
CV_lowcase          0.6947
CV_lowStop          0.6937
TFIDF_lowStopna     0.6895
TFIDF_lowStopstem   0.6895
CV_lowStopstem      0.6853
CV_lowStopna        0.6853
```

[ ]:
```

```

Matrice de confusion

[27]:
```
def plot_confusion_matrix(cm, classes=[],
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    See full source and example:
    http://scikit-learn.org/stable/auto_examples/model_selection/
 ↪plot_confusion_matrix.html

    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
```

```
    else:
        print('Confusion matrix, without normalization')

    thresh = 2*cm.max() / 3.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('Predicted label')
    plt.xlabel('True label')


def encode_true_and_false(val):
    if val == "true_or_false":
        return 1
    else:
        return 0

def plot_quad_error(X,y,model):
    arr1=[];arr2=[]
#     print("X: ", X.shape," y: ", y.shape)
    size = np.linspace(200 ,int(len(X)), 25).astype('int32') # if X.shape[0] >=␣
 ↪200 else X.shape[0]
    for m in size:
        X_s =  X.sample(m)
        y_s = y.loc[X_s.index.intersection(y.index)] # .intersection(y.index)

        X_train,X_val,y_train,y_val = train_test_split(X_s,y_s,train_size=0.
 ↪7,random_state=42)
        clf = model.fit(X_train, y_train)
        y_pred_train = clf.predict(X_train)
#         y_pred_train = y_pred.shape[0]
        y_pred_val = clf.predict(X_val)
#         y_pred_val = y_pred.shape[0]
        y_pred_train=np.vectorize(encode_true_and_false)(y_pred_train)
        y_pred_val=np.vectorize(encode_true_and_false)(y_pred_val)
        y_train=np.vectorize(encode_true_and_false)(y_train)
        y_val=np.vectorize(encode_true_and_false)(y_val)
#         print("y_pred_train: ", y_pred_train, "y_train: ", y_train, sep='\n')
        squarred_error_train = (1/len(y_pred_train))*np.sum((y_pred_train -␣
 ↪y_train)**2)
        squared_error_CV = (1/len(y_pred_val))*np.sum((y_pred_val - y_val)**2)
        arr1.append(squarred_error_train)
        arr2.append(squared_error_CV)
```

```
    plt.plot(size,arr1,label='train error')
    plt.plot(size,arr2,label='cv error')
    plt.xlabel("Training Size")
    plt.gca().set_xlim([100,len(X)])
    plt.ylabel("Quadratic Error")
    plt.title("Model Evaluation")
    plt.legend()
    plt.show()
```

[28]:
```
#         print("X_s.index: ", X_s.index, " y_index: ", y_s.index, "X.index:",␣
 ↪X.index, "y.index:", y.index, sep='\n')
#         y_s = y.sample(m)
#         X_s = X.loc[y_s.index] # .intersection(X.index)
#         print("X_s: ", X_s.shape," y_s: ", y_s.shape)
```

[29]:
```
dft.head()
```

[29]:

	public_id	text	title	rating
907	863897d7	Dr. Didier Raoult published new results of 1,0…	Dr. Didier Raoult published new results of 1,0…	mixture
918	054fefab	SUPREME COURT OF THE UNITED STATES No. 18A335 …	SUPREME COURT OF THE UNITED STATES No. 18A335 …	mixture
384	6f6addcc	Marine Corps. Rebukes Pelosi: "WE DON'T WORK F…	Marine Corps. Rebukes Pelosi: "WE DON'T WORK F…	true_or_false
1155	c1af4bf0	When Will The Planet Be Too Hot For Humans? Mu…	When Will The Planet Be Too Hot For Humans? Mu…	mixture
1245	d0b0e459	Global Ocean Circulation Appears To Be Collaps…	Global Ocean Circulation Appears To Be Collaps…	true_or_false

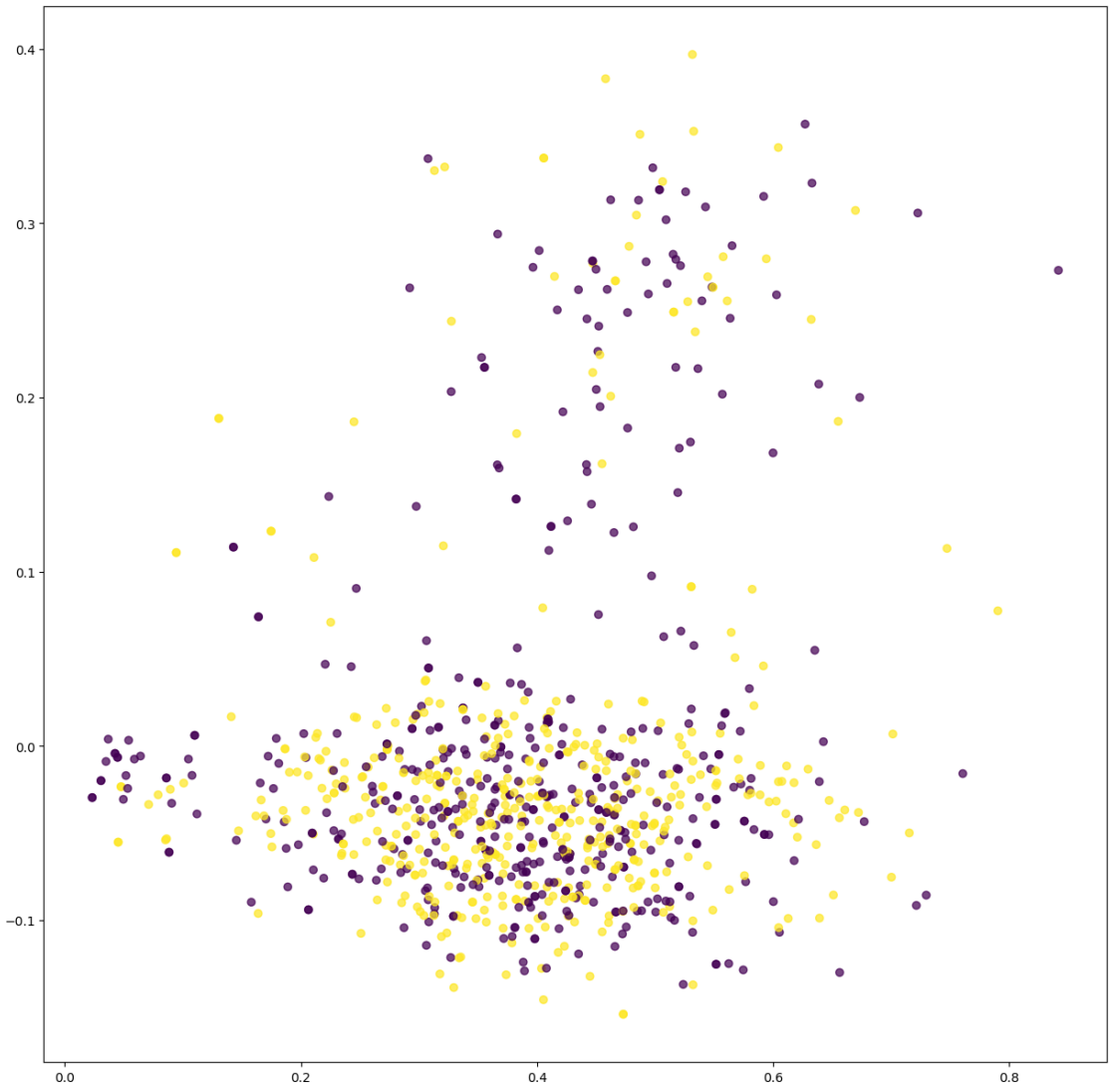
[30]:
```
X = dft["text"]
y = dft["rating"]
text_normalizer=␣
 ↪TextNormalizer(removestopwords=False,lowercase=True,getstemmer=False,removedigit=False)
X=text_normalizer.fit_transform(X)
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(X)
```

[31]:
```
# print(X, y)
```

[32]:
```
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import LabelEncoder

svd = TruncatedSVD(n_components=2)
X_svd = svd.fit_transform(X)
```

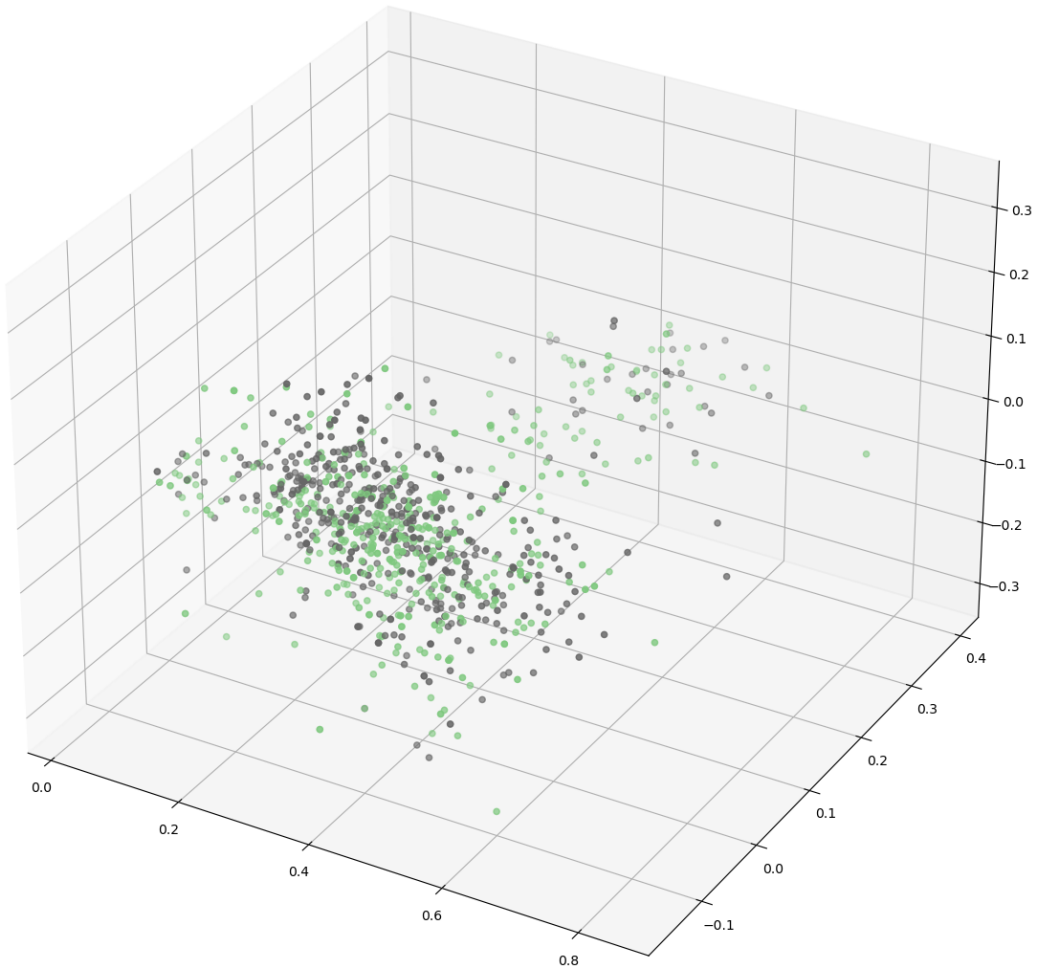
```
# Encode the classes' label as integers
le = LabelEncoder()
y_encoded = le.fit_transform(y)

print("variance ratio: ",svd.explained_variance_ratio_)

print("variance ratio sum: ",svd.explained_variance_ratio_.sum())
print("singular values:" ,svd.singular_values_)


fig,ax = plt.subplots(figsize=(15,15))
ax.scatter(X_svd[:,0], X_svd[:,1], alpha=0.72, c=y_encoded)
plt.show()
```

```
variance ratio:  [0.02051781 0.01229229]
variance ratio sum:  0.0328100962242348
singular values: [12.72291038  3.14966499]
```



[33]:
```
from mpl_toolkits.mplot3d import Axes3D

# Encode the classes' label as integers
le = LabelEncoder()
y_encoded = le.fit_transform(y)

svd = TruncatedSVD(n_components=3)
X_svd = svd.fit_transform(X)

fig = plt.figure(figsize=(15, 15))
ax = fig.add_subplot(projection='3d')
```

```
ax.scatter(X_svd[:1000,0],X_svd[:1000,1],
     X_svd[:1000,2],c=y_encoded, cmap='Accent')
plt.show()
```



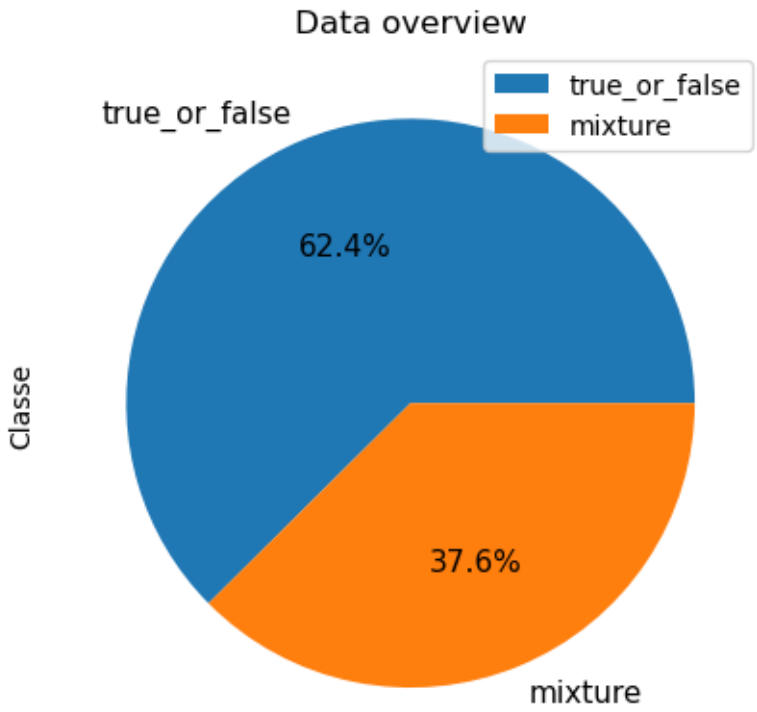
3 Classifications

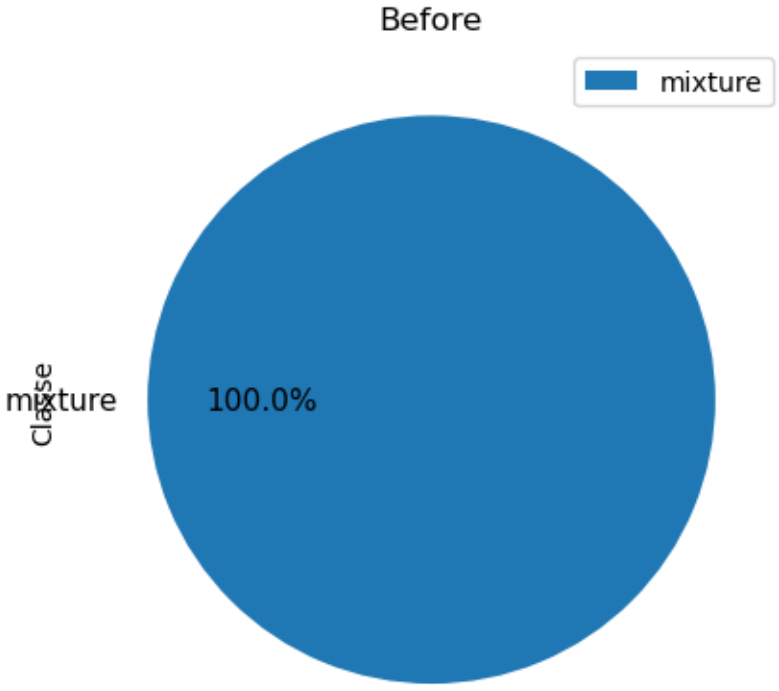
3.1 Classification Vrai vs Faux

3.1.1 Logistic Regression

[ ]:

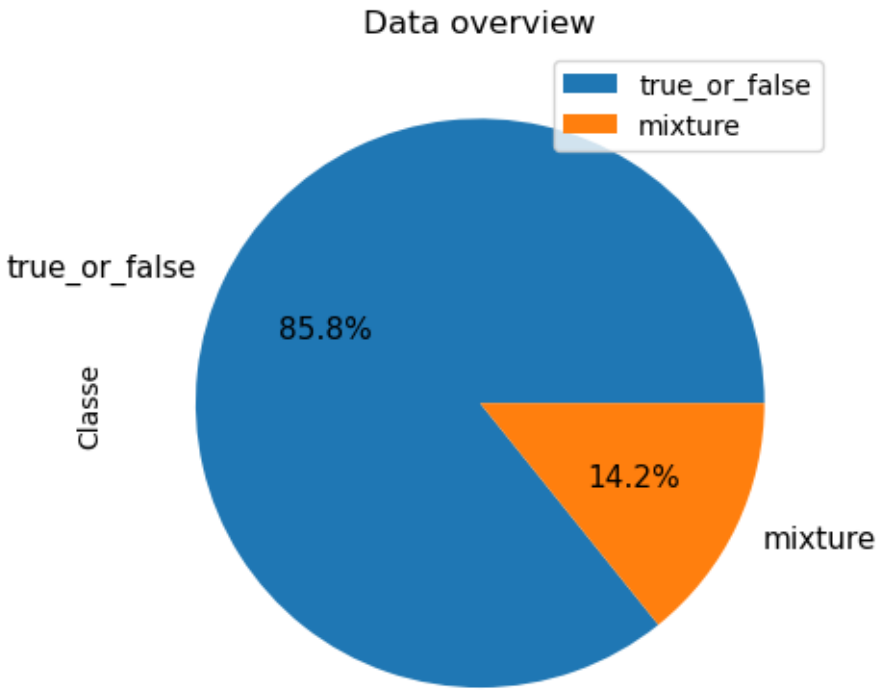
[34]:
```
df_train = Resampler(train_df_init, classif_type=1)
df_test = DataCleanerForClassification(test_df_init, classif_type=1)

text = df_train['text']
rating = df_train['rating']

arr = [i for i in range(len(df_train) + len(df_test))]
# print(arr, text)
idx = np.random.choice(arr,  250) #  int( len(df_train) / 2)
# print(idx)

text = np.concatenate((df_train["text"],df_test["text"]))
text = text[idx]


rating = np.concatenate(([df_train['rating'], df_test['rating']]))
rating = rating[idx]
# pd.DataFrame(rating).head()

text_normalizer=TextNormalizer(lowercase=True)
text=text_normalizer.fit_transform(text)
text = np.array(text)
# print(text)


vectorizer = TfidfVectorizer()
vectors = vectorizer.fit_transform(text)
corpus = np.array(vectorizer.get_feature_names())
print(corpus)
```


Data overview
true_or_false
mixture
true_or_false
62.4%
Classe
37.6%
mixture


Before
mixture
mixture
100.0%
Classe

Resampled

Classe




```
['00' '000' '0008' … 'óbito' ' g' '    '[
```

[35]:
```
def get_fake_no_fake_text(text):
    fake_text = np.array(text[rating == 'mixture'])
    non_fake_text = np.array(text[rating == 'true_and_false'])
    return fake_text,non_fake_text



def compute_frequency(fake_text,non_fake_text):
  # Computes 2 different frequency use of dictionary for O(1) time acces to␣
 ↪value of a paticular word
    fake_text = " ".join(fake_text)
    fake_text = word_tokenize(fake_text)
    non_fake_text = " ".join(non_fake_text)
    non_fake_text = word_tokenize(non_fake_text)
    corpus = np.unique(np.concatenate((fake_text,non_fake_text),axis=0))

    FakeFreq = {w:0 for w in corpus}
    NonFakeFreq = {w:0 for w in corpus}

    for word in fake_text:
```

```
        FakeFreq[word] += 1

    for word in non_fake_text:
        NonFakeFreq[word] += 1

    return FakeFreq,NonFakeFreq
```

[36]:
```
fake_text,non_fake_text= get_fake_no_fake_text(text)
FakeFreq,NonFakeFreq = compute_frequency(fake_text,non_fake_text)
```

[37]:
```
def feat_extraction_fake(row):
    row_text = word_tokenize(row)
    sum_fake_freq = 0
    for word in row_text:
        if word in FakeFreq.keys():
#             print("word: ", word, " FakeFreq[word]:", FakeFreq[word])
            sum_fake_freq += FakeFreq[word]
    return sum_fake_freq

def feat_extraction_no_fake(row):
    row_text = word_tokenize(row)
    sum_non_fake_freq = 0
    for word in row_text:
        if word in NonFakeFreq.keys():
            sum_non_fake_freq += NonFakeFreq[word]
    return sum_non_fake_freq
```

[38]:
```
def conf_matrix(model):
    ConfusionMatrixDisplay.from_estimator(
        model,
        X_test,
        y_test
    )



def class_report(model):
    print(classification_report(
        Y_test,
        model.predict(X_test)
    ))
```

[39]:
```
# text.shape
```

[40]:
```
dfbis_train = pd.DataFrame(text,columns=["text"])

# dfbis_train["bias"] = 1
```

```
dfbis_train["mixture"] = dfbis_train["text"].apply(feat_extraction_fake)

dfbis_train["true_or_false"] = dfbis_train["text"].
 ↪apply(feat_extraction_no_fake) # , args=[NonFakeFreq]

X = dfbis_train[['mixture', 'true_or_false']]
# X = dfbis_train['text']
y = pd.DataFrame(rating)
# dfbis_train

# text = pd.DataFrame(text)
# text.sample(10)
# dfbis_train.sample(10)
# X.sample(10)
# X
```

Training

[41]:
```
# ps = PorterStemmer()
# def stemming(content):
#     stemmed_content = re.sub('[^a-zA-Z]',' ',content)
#     stemmed_content = stemmed_content.lower()
#     stemmed_content = stemmed_content.split()
#     stemmed_content = [ps.stem(word) for word in stemmed_content if not word␣
 ↪in stopwords.words('english')]
#     stemmed_content = ' '.join(stemmed_content)
#     return stemmed_content
```

[42]:
```
# # dfbis_train.sample(2)
# dfbis_train["text"] = dfbis_train["text"].apply(stemming)

# X = dfbis_train["text"]
# y = rating

# vector = TfidfVectorizer()
# vector.fit(X)
# X = vector.transform(X)

# # X.sample(2)
# print(X)
```

[43]:
```
# X


X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=0.
 ↪7,random_state=42)
```

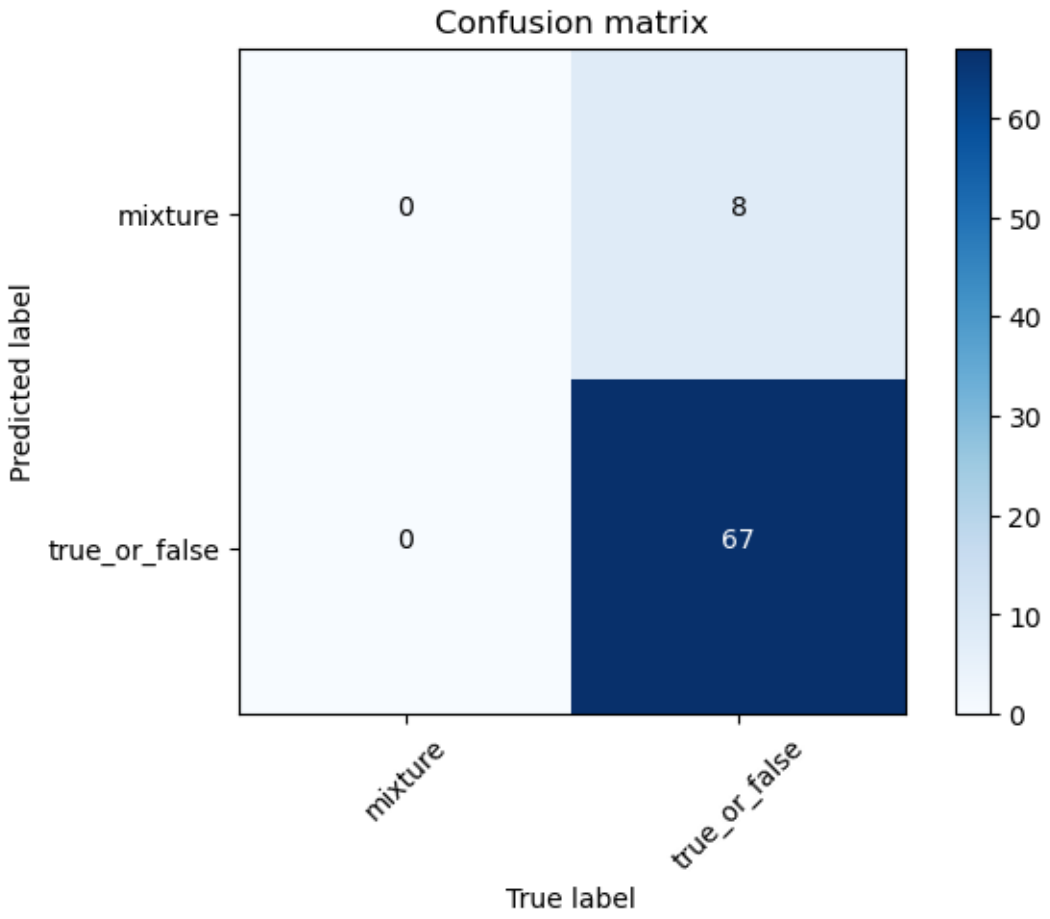
```
clf = LogisticRegression(random_state=42).fit(X_train, y_train)

y_pred = clf.predict(X_test)

target_names = ["mixture","true_or_false"]

print('accuracy %s' % accuracy_score(y_pred, y_test))
print(classification_report(y_test, y_pred)) # target_names=target_names
```

```
accuracy 0.8933333333333333
               precision    recall  f1-score   support

      mixture       0.00      0.00      0.00         8
true_or_false       0.89      1.00      0.94        67

     accuracy                           0.89        75
    macro avg       0.45      0.50      0.47        75
 weighted avg       0.80      0.89      0.84        75


/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/utils/validation.py:993: DataConversionWarning: A column-vector
y was passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[44]:

```
plot_confusion_matrix(confusion_matrix(y_test, y_pred), classes=target_names)
# confusion_matrix(y_test, y_pred)
# conf_matrix(clf)
```

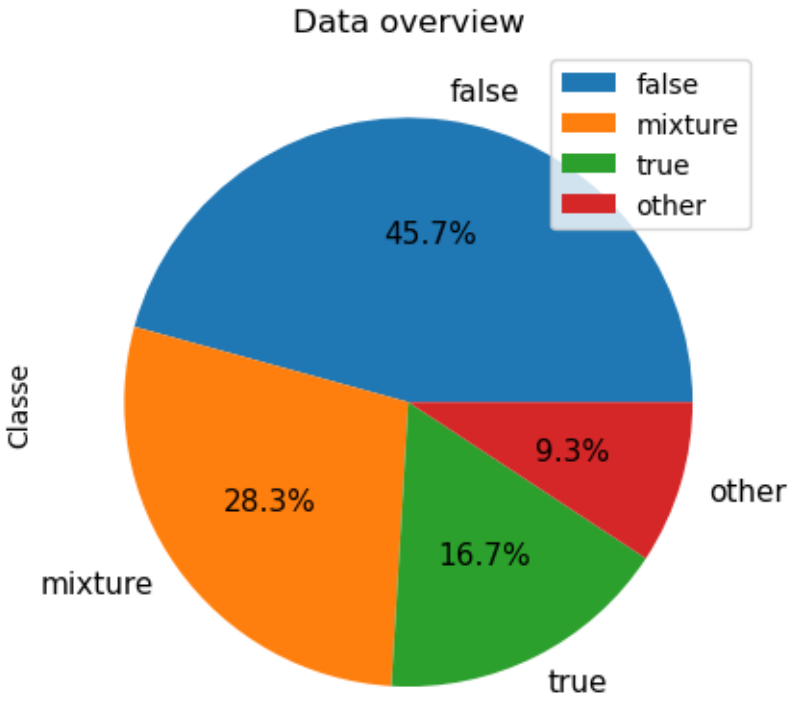
```
Confusion matrix, without normalization
```

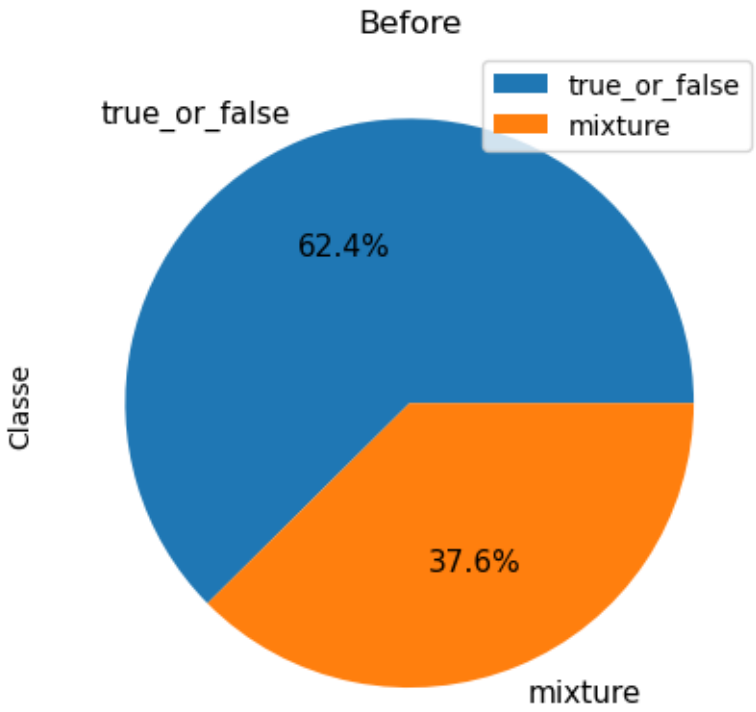


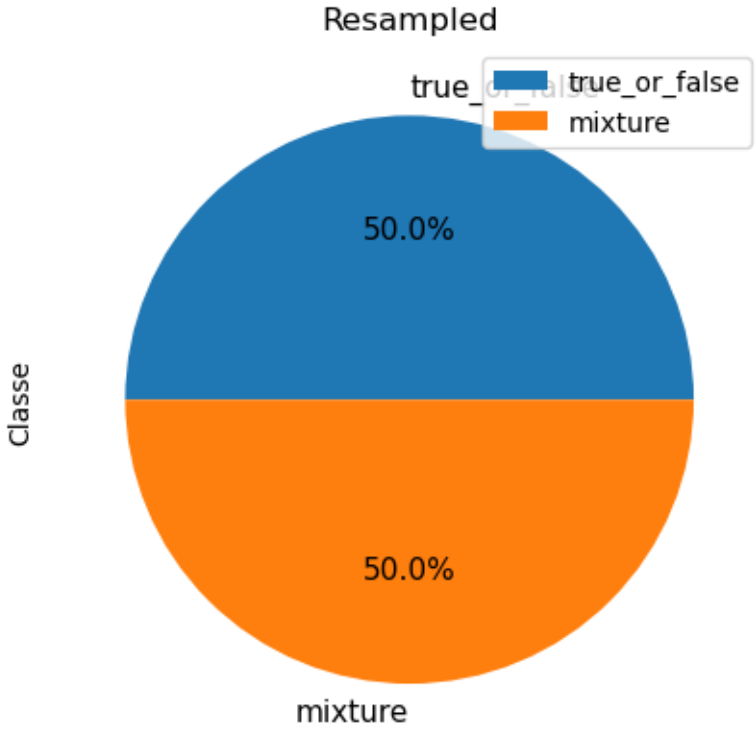
3.1.2 Naives Bayes

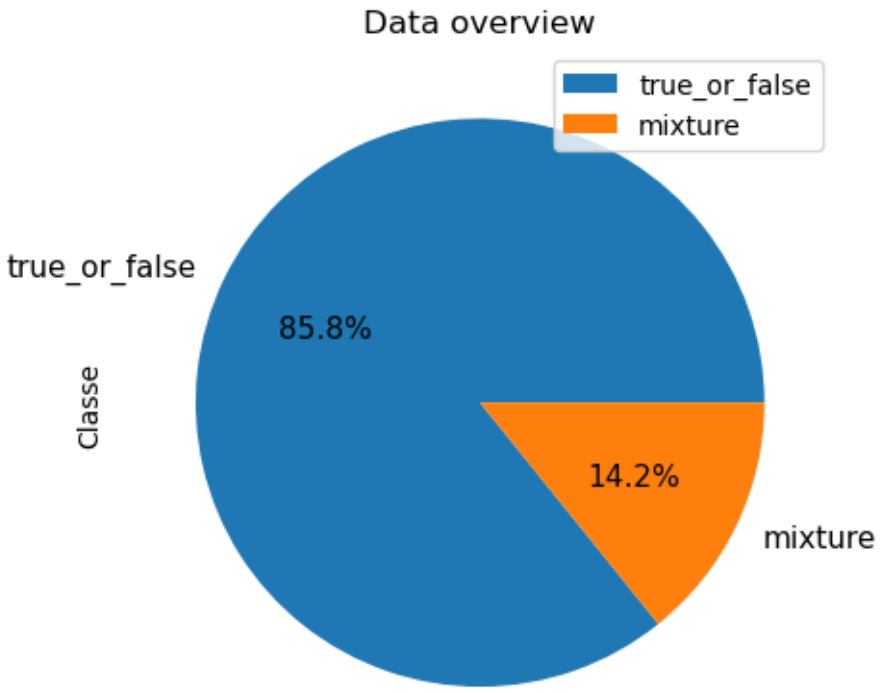
[45]:
```
train_df_init = pd.read_csv('HAI817_Project_data/HAI817_Projet_train.csv')
train_df_init = train_df_init.rename(columns={"our rating": "rating"})

test_df_init = pd.read_csv("HAI817_Project_data/HAI817_Projet_test.csv")
test_df_init = test_df_init.rename(columns={"our rating": "rating"})



train_df_init = train_df_init.fillna(' ')
test_df_init = test_df_init.fillna(' ')

train_df_init['text'] = train_df_init['title'] + " "+train_df_init['text']
test_df_init['text'] = test_df_init['title'] + " "+test_df_init['text']

df_train = Resampler(train_df_init, classif_type=1)
df_test = DataCleanerForClassification(test_df_init, classif_type=1)
```

```
X = df_train["text"]
y = df_train["rating"]

print(X)

text_normalizer=␣
  ↪TextNormalizer(removestopwords=True,lowercase=True,getstemmer=True,removedigit=True)
X=text_normalizer.fit_transform(X)
```




Before
true_or_false
mixture
true_or_false
62.4%
Classe
37.6%
mixture


Resampled
true_or_false
mixture
50.0%
Classe
50.0%
mixture




```
858     Up to a third of millennials 'face renting the…
467     Khrushchev tours America: 'No sour cabbage sou…
273     New Mexico House Committee Passes Bill to Lega…
561     Rand Paul: We Must Demilitarize the Police The…
222     Pro-Life Groups Upset UN Coronavirus Relief Fu…
                              …
167     Bibles Pulled From Shelves For Outdated Idea T…
722     A very fresh look at climate change Milwaukee …
1202    Australia's Great Barrier Reef has worst coral…
232     Langevin, Cicilline Call for Stronger Backgrou…
1158    The three-degree world: cities that will be dr…
Name: text, Length: 950, dtype: object
```

[46]:
```
df_train
```

[46]:
```
     public_id                                               text
858   6600699c  Up to a third of millennials 'face renting the…  \
467   45525605  Khrushchev tours America: 'No sour cabbage sou…
273   5dc31757  New Mexico House Committee Passes Bill to Lega…
561   0192a5cc  Rand Paul: We Must Demilitarize the Police The…
```

```
222   b906fccf  Pro-Life Groups Upset UN Coronavirus Relief Fu…
…        …                                                  …
167   6c95fb76  Bibles Pulled From Shelves For Outdated Idea T…
722   95020291  A very fresh look at climate change Milwaukee …
1202  0a68da63  Australia's Great Barrier Reef has worst coral…
232   bfd9ddf6  Langevin, Cicilline Call for Stronger Backgrou…
1158  93db83ce  The three-degree world: cities that will be dr…

                                                    title         rating
858   Up to a third of millennials 'face renting the…  true_or_false
467   Khrushchev tours America: 'No sour cabbage sou…        mixture
273   New Mexico House Committee Passes Bill to Lega…  true_or_false
561          Rand Paul: We Must Demilitarize the Police  true_or_false
222   Pro-Life Groups Upset UN Coronavirus Relief Fu…  true_or_false
…                                                    …             …
167   Bibles Pulled From Shelves For Outdated Idea T…  true_or_false
722                  A very fresh look at climate change  true_or_false
1202  Australia's Great Barrier Reef has worst coral…  true_or_false
232   Langevin, Cicilline Call for Stronger Backgrou…        mixture
1158  The three-degree world: cities that will be dr…        mixture

[950 rows x 4 columns]
```

[73]:
```
# print(X)
```

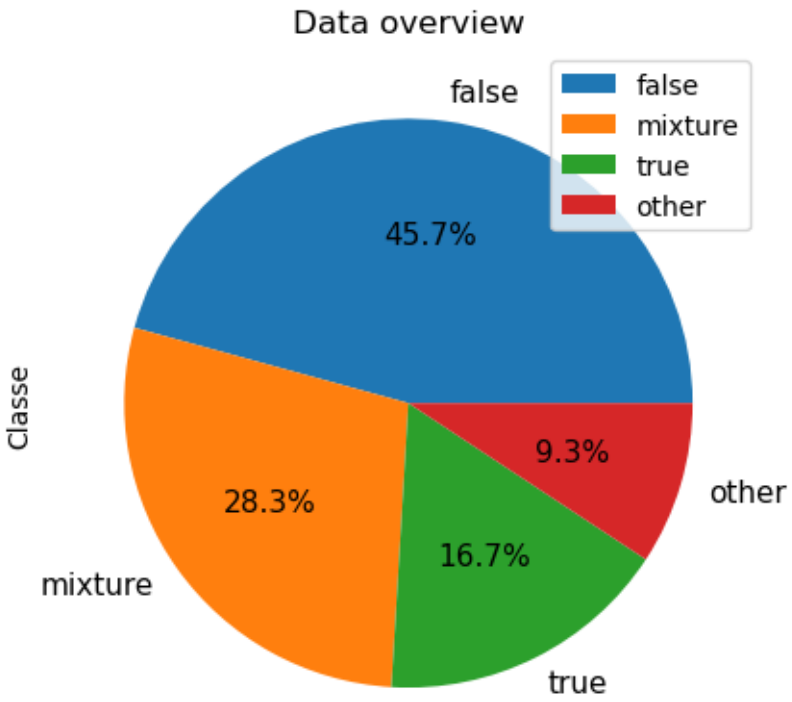
[48]:
```
X_train,X_val,y_train,y_val = train_test_split(X,y,train_size=0.
↪7,random_state=42)

nb = Pipeline([('vect', TfidfVectorizer()),
               ('tfidf', TfidfTransformer()),
               ('clf', MultinomialNB()),
              ])
nb.fit(X_train, y_train)

y_pred = nb.predict(X_val)

print('accuracy %s' % accuracy_score(y_pred, y_val))
print(classification_report(y_val, y_pred))
```

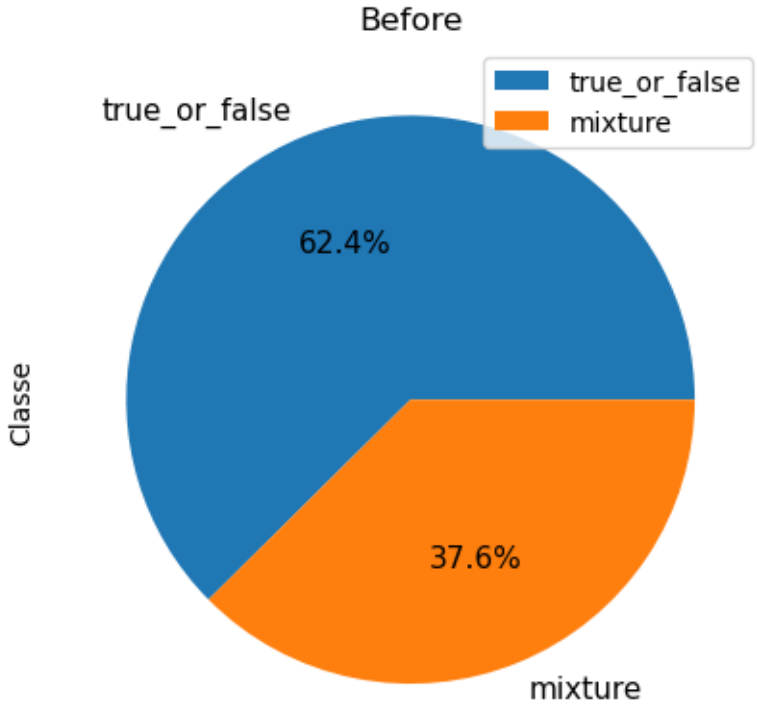
```
accuracy 0.6736842105263158
              precision    recall  f1-score   support

      mixture       0.63      0.80      0.70       139
true_or_false       0.74      0.55      0.64       146

     accuracy                           0.67       285
    macro avg       0.69      0.68      0.67       285
```

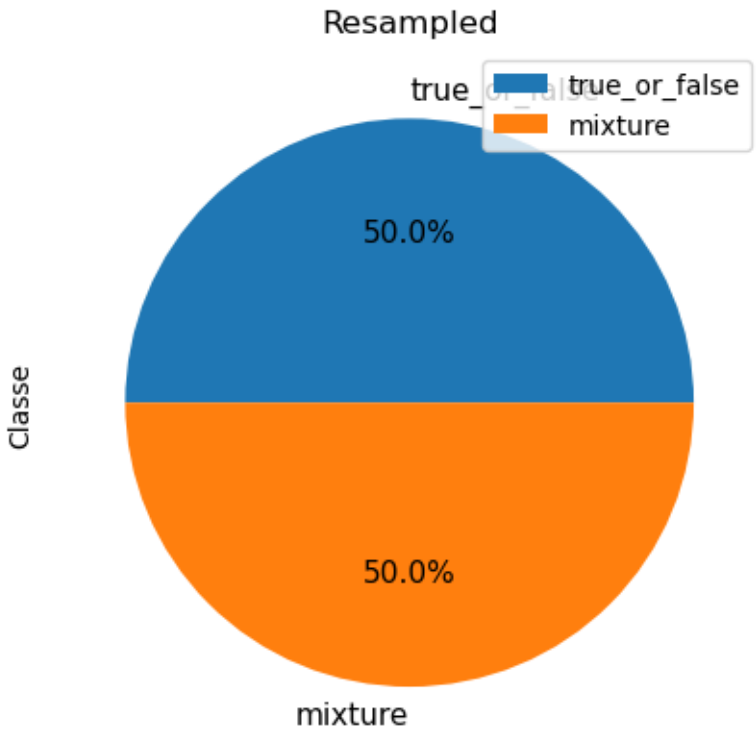
```
weighted avg       0.69      0.67      0.67       285
```

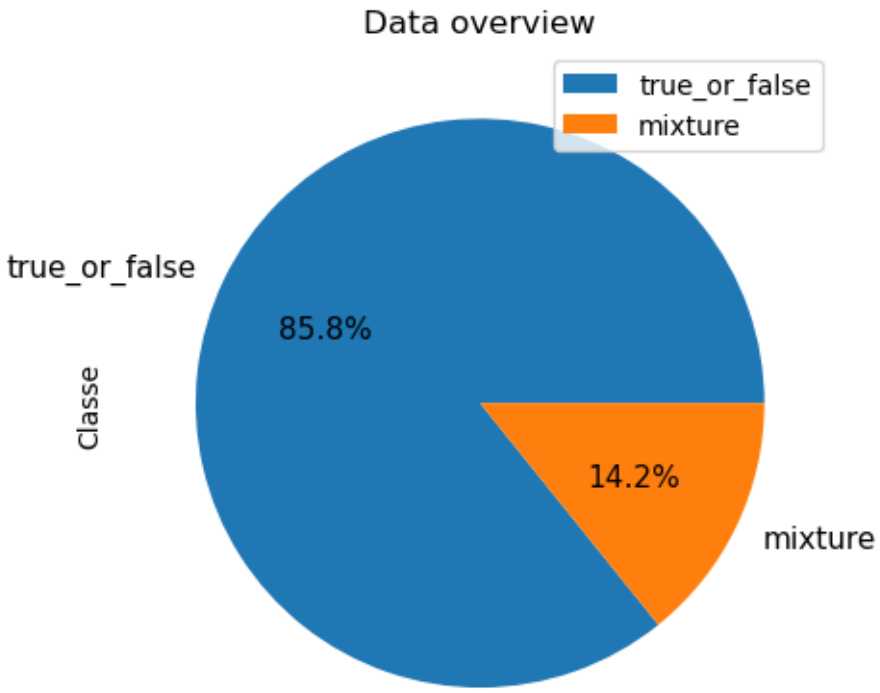
3.1.3 La Cross Validation

[49]:
```
train_df_init = pd.read_csv('HAI817_Project_data/HAI817_Projet_train.csv')
train_df_init = train_df_init.rename(columns={"our rating": "rating"})

test_df_init = pd.read_csv("HAI817_Project_data/HAI817_Projet_test.csv")
test_df_init = test_df_init.rename(columns={"our rating": "rating"})



train_df_init = train_df_init.fillna(' ')
test_df_init = test_df_init.fillna(' ')

train_df_init['text'] = train_df_init['title'] + " "+train_df_init['text']
test_df_init['text'] = test_df_init['title'] + " "+test_df_init['text']

df_train = Resampler(train_df_init, classif_type=1)
df_test = DataCleanerForClassification(test_df_init, classif_type=1)


X = df_train["text"]
y = df_train["rating"]
# found when testing different parameters in the preprocessing
text_normalizer=␣
 ↪TextNormalizer(removestopwords=False,lowercase=True,getstemmer=False,removedigit=False)
X=text_normalizer.fit_transform(X)

models = []
models.append(('LRegression', LogisticRegression()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('MultinomialNB', MultinomialNB()))
models.append(('DecisionTreeClassifier', DecisionTreeClassifier()))
models.append(('SVM', SVC()))
models.append(('SGDClassifier',SGDClassifier()))
models.append(('RandomForest', RandomForestClassifier()))
```


Data overview
false
mixture
true
other
45.7%
28.3%
16.7%
9.3%
Classe
false
mixture
true
other


Before
true_or_false
mixture
62.4%
37.6%
Classe
true_or_false
mixture


Resampled
true_or_false
mixture
50.0%
50.0%
Classe
mixture

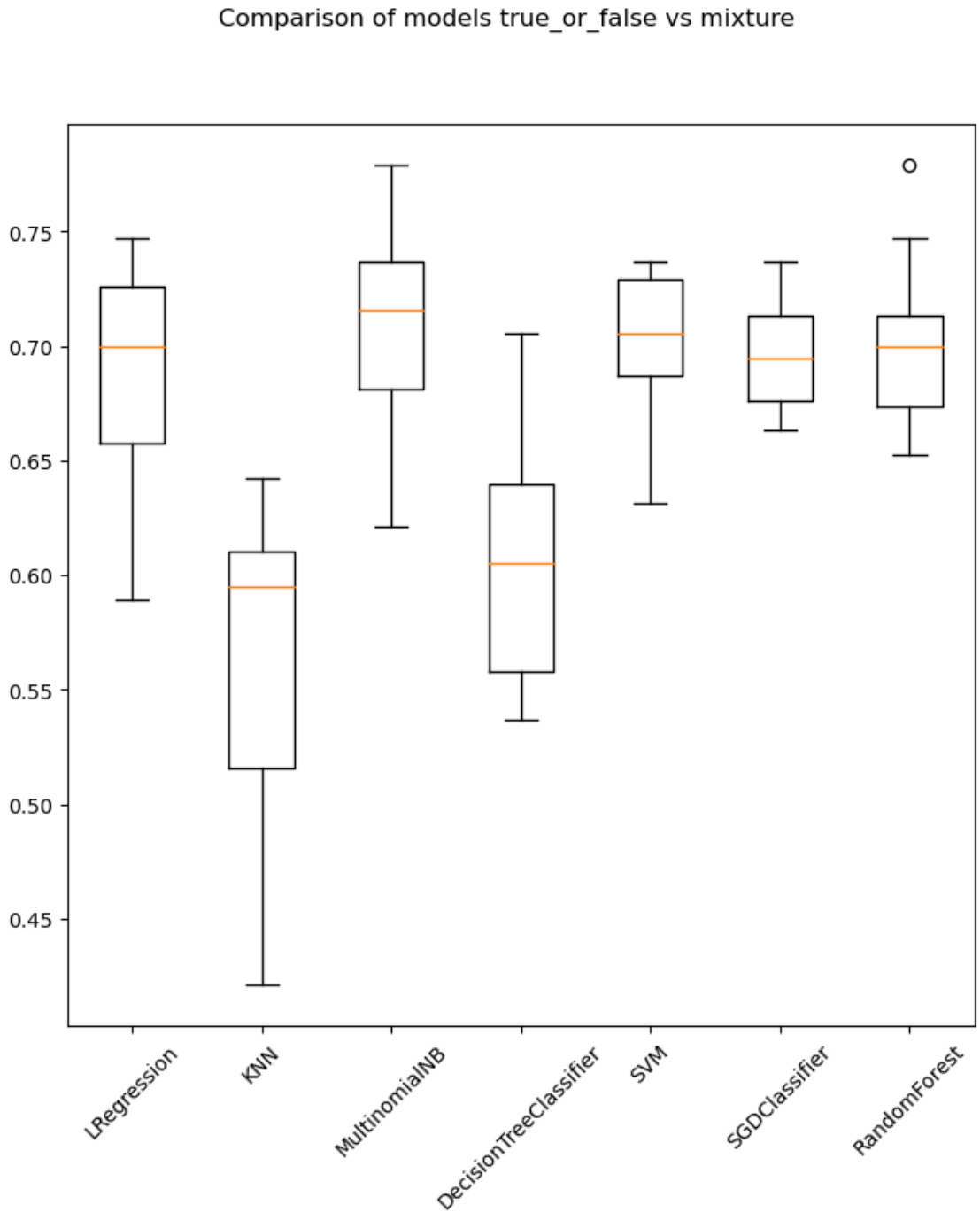


[50]:
```
results = []
names = []
scoring = 'accuracy'
for name, model in models:
  kfold = KFold(n_splits=10, shuffle=True)
  model = make_pipeline(TfidfVectorizer(), model)
  cv_results = cross_val_score(model, X, y, cv=kfold, scoring=scoring)
  results.append(cv_results)
  names.append(name)
  msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
  print(msg)
```

```
LRegression: 0.689474 (0.047834)
KNN: 0.564211 (0.072807)
MultinomialNB: 0.706316 (0.047880)
DecisionTreeClassifier: 0.602105 (0.052166)
SVM: 0.702105 (0.030526)
SGDClassifier: 0.695789 (0.024211)
RandomForest: 0.700000 (0.038316)
```

[51]:
```
classes = ["true_or_false", "mixture"]
fig = plt.figure(figsize=(8,8))
```

```
if len(classes)>2:
  fig.suptitle('Comparison of models {} vs {} vs {}'.
 ↪format(classes[0],classes[1],classes[2]))
else:
  fig.suptitle('Comparison of models {} vs {}'.format(classes[0],classes[1]))
ax = fig.add_subplot(111)
ax.boxplot(results)
ax.set_xticklabels(names,rotation = 45)
plt.show()
```


Comparison of models true_or_false vs mixture
0.75
0.70
0.65
0.60
0.55
0.50
0.45
LRegression
KNN
MultinomialNB
DecisionTreeClassifier
SVM
SGDClassifier
RandomForest

3.1.4 Hyperparameters finding process

3.1.5 Grid Search CV

3.1.6 Logistic Regression

[52]:
```
X = df_train["text"]
y = df_train["rating"]

X_s = X  # X.sample(300) # X if X.shape[0] <=700 else X.sample(700)
y_s = y.loc[X_s.index]
preprocess_selection("logistic_regression",LogisticRegression(
#     n_jobs=-1
),X_s,y_s)
```

```
Evaluation des différentes configurations :

/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
```

```
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```

```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
```

```
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
```

```
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
```

```
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
```

```
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
```

```
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(

Pipeline             Score
-----------------  -------
TFIDF_brut          0.7063
TFIDF_lowStop       0.7000
TFIDF_lowStopstem   0.7000
TFIDF_lowcase       0.6947
TFIDF_lowStopna     0.6863
CV_brut             0.6821
CV_lowcase          0.6821
```

```
CV_lowStop          0.6674
CV_lowStopstem      0.6674
CV_lowStopna        0.6674
```

[53]:
```
X = df_train["text"]
y = df_train["rating"]

# put text normalizer in the pipe is a bad idea because of the very long time␣
 ↪processing
text_normalizer =␣
 ↪TextNormalizer(removestopwords=False,lowercase=True,getstemmer=False,removedigit=False)
X = text_normalizer.fit_transform(X)

X_train,X_val,y_train,y_val = train_test_split(X,y,train_size=0.
 ↪7,random_state=42)

pipe = Pipeline([('vect', TfidfVectorizer()),
                 ('clf', LogisticRegression(solver='lbfgs')),
                 ])

grid = {"clf__penalty": ['l2' ,'none'],
        "clf__C":np.logspace(-2,2,4),
        "clf__max_iter":[100,1000]}

gd_srLR = GridSearchCV(pipe,
                       param_grid=grid,
                       scoring='accuracy',
                       cv=10,
#                        n_jobs=-1,
                      return_train_score=True)

gd_srLR.fit(X_train, y_train)
print('meilleur score ',
 gd_srLR.best_score_,'\n')
print('meilleurs paramètres',
 gd_srLR.best_params_,'\n')
print('meilleur estimateur',
gd_srLR.best_estimator_,'\n')
```

```
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
```

```
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
```

```
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
```

```
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
```

```
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
```

```
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
```

```
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
```

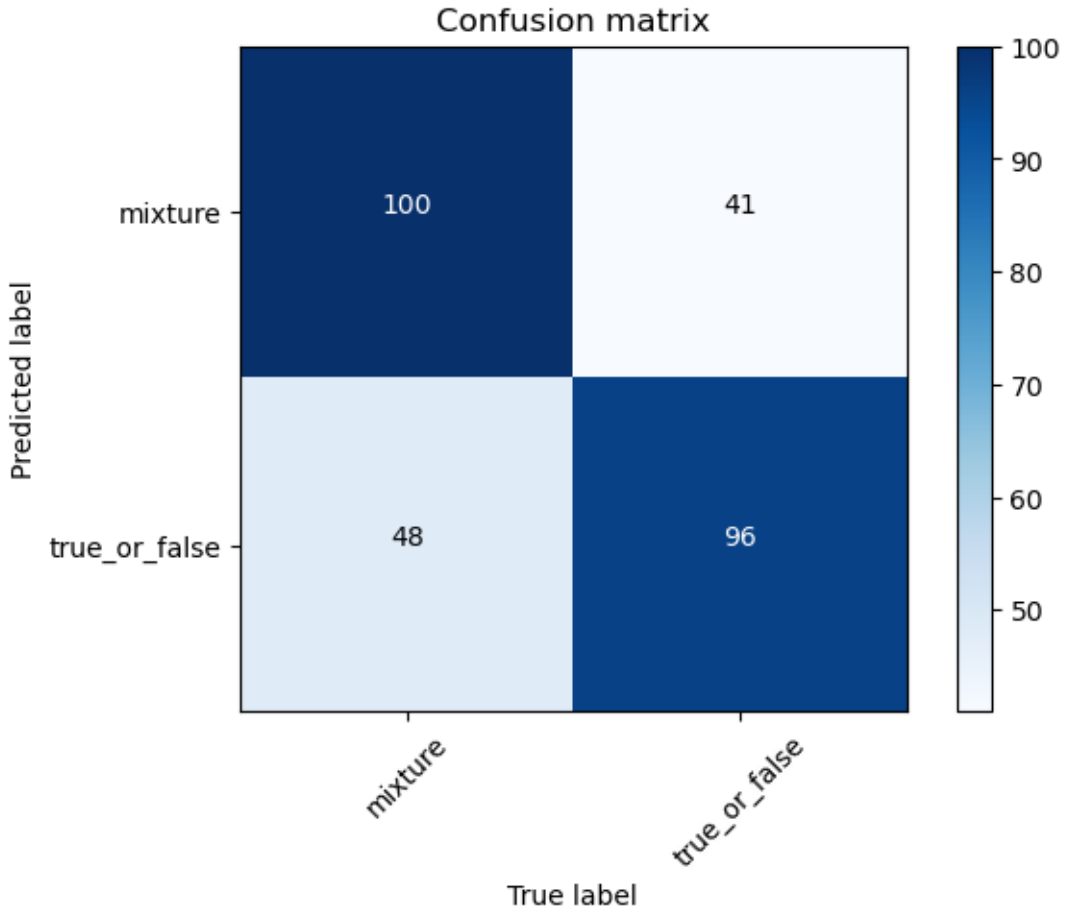
```
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
/home/richard/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:1483: UserWarning: Setting
penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
```

```
meilleur score  0.6811849841700589

meilleurs paramètres {'clf__C': 4.6415888336127775, 'clf__max_iter': 100,
'clf__penalty': 'l2'}

meilleur estimateur Pipeline(steps=[('vect', TfidfVectorizer()), ('tfidf',
TfidfTransformer()),
                ('clf', LogisticRegression(C=4.6415888336127775))])
```

[54]:
```python
# Creation d'une instance de l'algorithme en utilisant les meilleurs paramètres
lr = gd_srLR.best_estimator_

lr.fit(X_train, y_train)
y_pred = lr.predict(X_val)
print('\n accuracy: ', accuracy_score(y_pred, y_val),'\n')

print('\n',classification_report(y_val, y_pred))
plot_confusion_matrix(confusion_matrix(y_val, y_pred),classes = target_names)
```

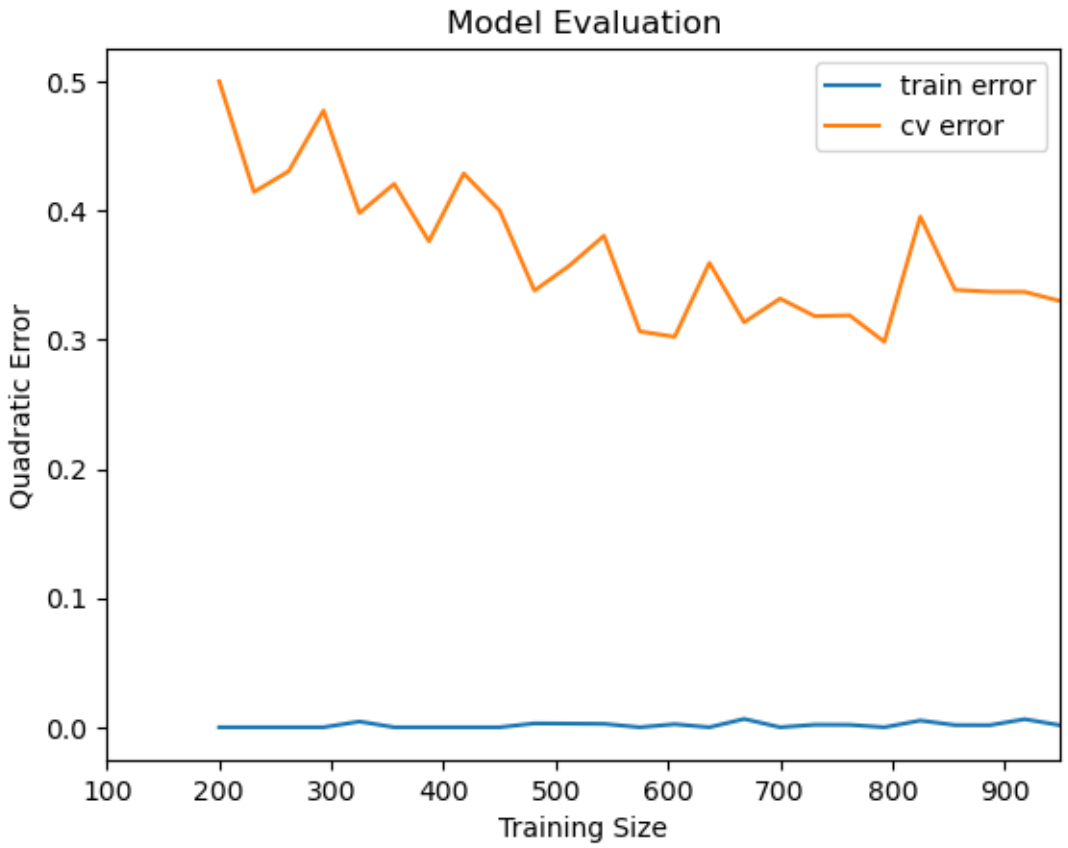
```
 accuracy:  0.6877192982456141
```

```
                precision    recall  f1-score   support

       mixture       0.68      0.71      0.69       141
 true_or_false       0.70      0.67      0.68       144

      accuracy                           0.69       285
     macro avg       0.69      0.69      0.69       285
  weighted avg       0.69      0.69      0.69       285

Confusion matrix, without normalization
```



[55]:
```
X = df_train["text"]
y = df_train["rating"]
text_normalizer =␣
  ↪TextNormalizer(removestopwords=True,lowercase=True,getstemmer=True,removedigit=False)
```

```
# print("From call:  X.index: ",X.index, "y.index", y.index)
X = X.reset_index(drop=True)
y = y.reset_index(drop=True)


X = text_normalizer.fit_transform(X)

# X = X.reset_index(drop=True)
# y = y.reset_index(drop=True)

X = pd.Series(X)

plot_quad_error(X,y,lr)
```

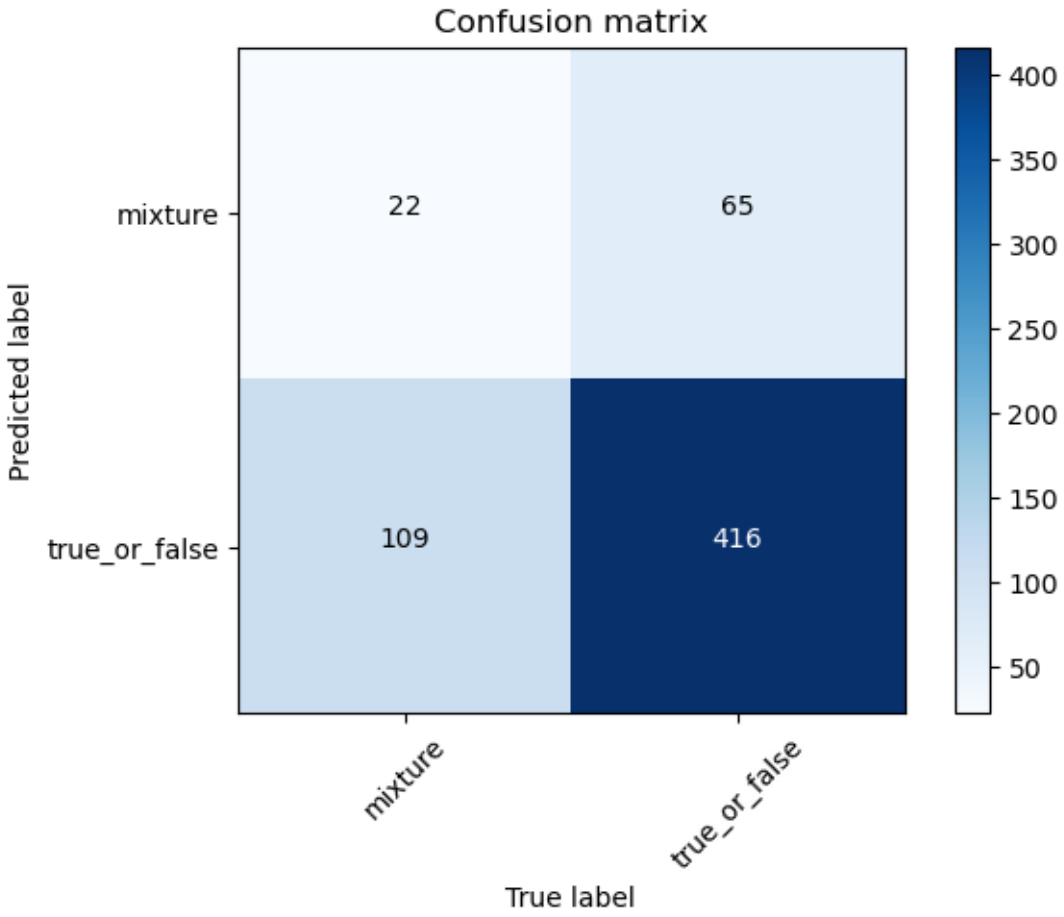


[56]:
```
X_test,y_test = df_test["text"],df_test["rating"]
y_pred = lr.predict(X_test)
print('\n accuracy: ', accuracy_score(y_pred, y_test),'\n')

print('\n',classification_report(y_test, y_pred)) # ,target_names=classes
plot_confusion_matrix(confusion_matrix(y_test, y_pred),classes = target_names)
```

```
accuracy:  0.7156862745098039
```

	precision	recall	f1-score	support
mixture	0.17	0.25	0.20	87
true_or_false	0.86	0.79	0.83	525
accuracy			0.72	612
macro avg	0.52	0.52	0.51	612
weighted avg	0.77	0.72	0.74	612

```
Confusion matrix, without normalization
```



3.1.7 Svc

[57]:
```
X = df_train["text"]
y = df_train["rating"]

X_s = X # X.sample(300) # X if X.shape[0] <=700 else X.sample(700)
y_s = y.loc[X_s.index]
preprocess_selection("SVC",SVC(),X_s,y_s)
```

```
Evaluation des différentes configurations :
Pipeline             Score
-----------------  -------
TFIDF_brut          0.7242
TFIDF_lowcase       0.7116
TFIDF_lowStop       0.7084
TFIDF_lowStopstem   0.7053
TFIDF_lowStopna     0.7000
CV_lowStop          0.6432
CV_lowStopstem      0.6337
CV_lowStopna        0.6316
CV_brut             0.5884
CV_lowcase          0.5842
```

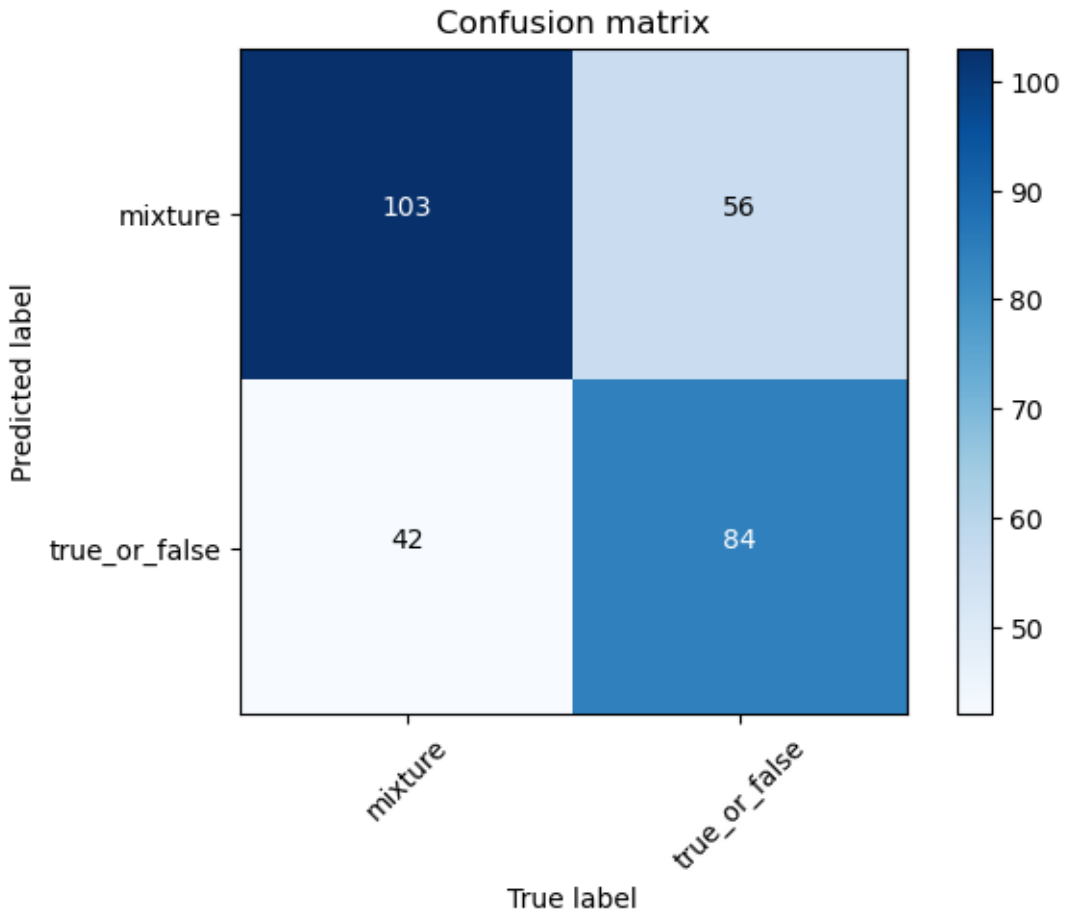
[58]:
```
X = df_train["text"]
y = df_train["rating"]
# put text normalizer in the pipe is a bad idea because of the very long time␣
 ↪processing
text_normalizer =␣
 ↪TextNormalizer(removestopwords=True,lowercase=True,getstemmer=True,removedigit=False)
X = text_normalizer.fit_transform(X)
X_train,X_val,y_train,y_val = train_test_split(X,y,train_size=0.
 ↪7,random_state=0)

pipe = Pipeline([('vect', TfidfVectorizer()),
                 ('clf', SVC()),
                ])

grid = {'clf__C': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
        'clf__kernel': ['linear'],
        'clf__gamma': [ 0.0001, 0.001, 0.01, 0.1, 1],
        'clf__kernel': ['linear','rbf','poly','sigmoid']}

gd_srSVC = GridSearchCV(pipe,
                        param_grid=grid,
                        scoring='accuracy',
                        cv=10,
#                         n_jobs=-1,
```

```
                    return_train_score=True)

gd_srSVC.fit(X_train, y_train)
print('meilleur score ',
 gd_srSVC.best_score_,'\n')
print('meilleurs paramètres',
 gd_srSVC.best_params_,'\n')
print('meilleur estimateur',
gd_srSVC.best_estimator_,'\n')
```

```
meilleur score  0.670872908186341

meilleurs paramètres {'clf__C': 100, 'clf__gamma': 0.1, 'clf__kernel': 'rbf'}

meilleur estimateur Pipeline(steps=[('vect', TfidfVectorizer()), ('tfidf',
TfidfTransformer()),
                ('clf', SVC(C=100, gamma=0.1))])
```

[59]:
```
svm = gd_srSVC.best_estimator_

svm.fit(X_train, y_train)
y_pred = svm.predict(X_val)
print('\n accuracy: ', accuracy_score(y_pred, y_val),'\n')

print('\n',classification_report(y_val, y_pred))
plot_confusion_matrix(confusion_matrix(y_val, y_pred),classes = target_names)
```

```
 accuracy:  0.656140350877193


              precision    recall  f1-score   support

     mixture       0.71      0.65      0.68       159
true_or_false       0.60      0.67      0.63       126

    accuracy                           0.66       285
   macro avg       0.66      0.66      0.65       285
weighted avg       0.66      0.66      0.66       285

Confusion matrix, without normalization
```



[60]:
```
X = df_train["text"]
y = df_train["rating"]
# put text normalizer in the pipe is a bad idea because of the very long time␣
 ↪processing
text_normalizer = TextNormalizer(removestopwords=True,
                                 lowercase=True,
                                 getstemmer=True,
                                 removedigit=False)
X = text_normalizer.fit_transform(X)
X_train,X_val,y_train,y_val = train_test_split(X,y,train_size=0.
 ↪7,random_state=42)

pipe = Pipeline([('vect', TfidfVectorizer()),
                 ('clf', SVC()),
                ])

grid = {
   'vect__stop_words':['english',None],
```

```
    'vect__lowercase': [True,False],
     'clf__C': [1, 10],
    'clf__gamma' : [1],
    'clf__kernel': ['rbf']
       }

gd_srSVC = GridSearchCV(pipe,
                     param_grid=grid,
                     scoring='accuracy',
                     cv=10,
#                      n_jobs=-1,
                    return_train_score=True)

gd_srSVC.fit(X_train, y_train)
print('meilleur score ',
 gd_srSVC.best_score_,'\n')
print('meilleurs paramètres',
 gd_srSVC.best_params_,'\n')
print('meilleur estimateur',
gd_srSVC.best_estimator_,'\n')
```

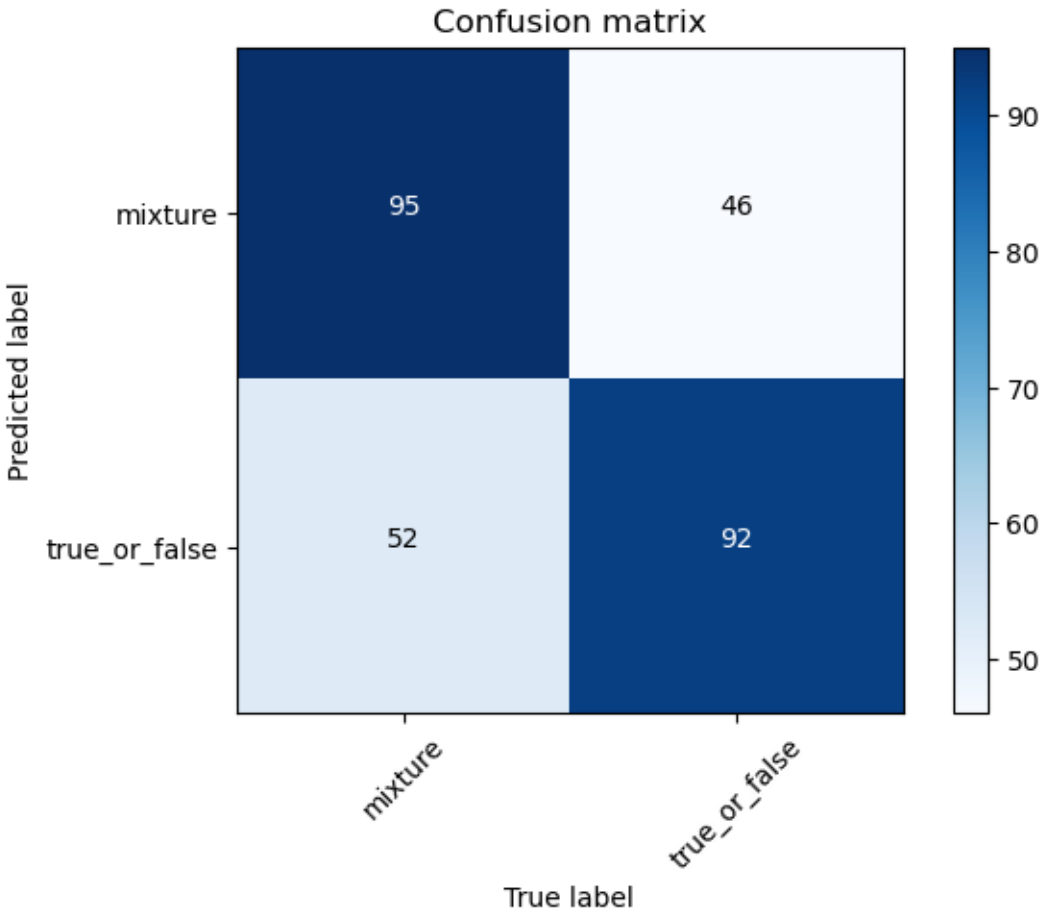
```
meilleur score  0.694843962008141

meilleurs paramètres {'clf__C': 10, 'clf__gamma': 1, 'clf__kernel': 'rbf',
'vect__lowercase': True, 'vect__stop_words': 'english'}

meilleur estimateur Pipeline(steps=[('vect',
TfidfVectorizer(stop_words='english')),
                ('tfidf', TfidfTransformer()), ('clf', SVC(C=10, gamma=1))])
```

[61]:
```
# Creation d'une instance de l'algorithme en utilisant les meilleurs paramètres
svm = gd_srSVC.best_estimator_

svm.fit(X_train, y_train)
y_pred = svm.predict(X_val)
print('\n accuracy: ', accuracy_score(y_pred, y_val),'\n')

print('\n',classification_report(y_val, y_pred))
plot_confusion_matrix(confusion_matrix(y_val, y_pred),classes = target_names)
```

```
 accuracy:  0.656140350877193

              precision    recall  f1-score   support

     mixture       0.65      0.67      0.66       141
```

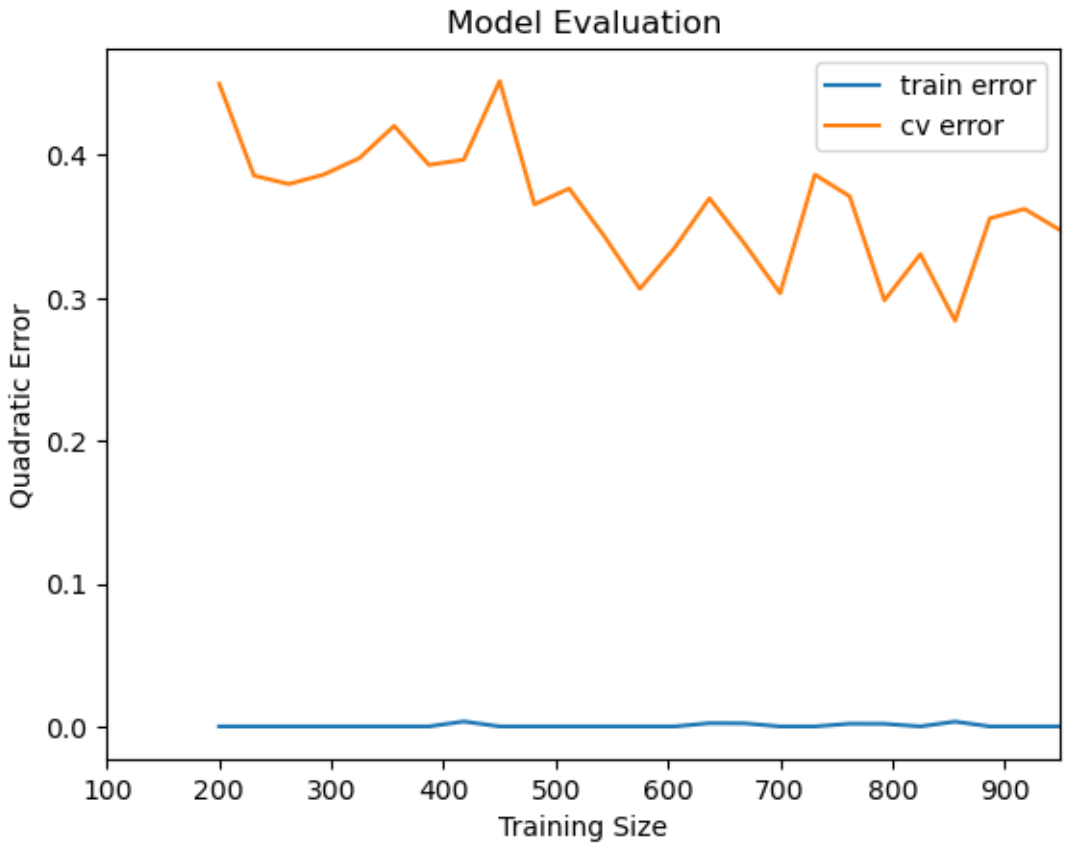
```
true_or_false       0.67      0.64      0.65       144

    accuracy                            0.66       285
   macro avg        0.66      0.66      0.66       285
weighted avg        0.66      0.66      0.66       285
```

```
Confusion matrix, without normalization
```



[62]:
```
X = df_train["text"]
y = df_train["rating"]
text_normalizer =␣
  ↪TextNormalizer(removestopwords=True,lowercase=True,getstemmer=True,removedigit=False)

X = X.reset_index(drop=True)
y = y.reset_index(drop=True)

X = text_normalizer.fit_transform(X)
X = pd.Series(X)
```

```
plot_quad_error(X,y,svm)
```

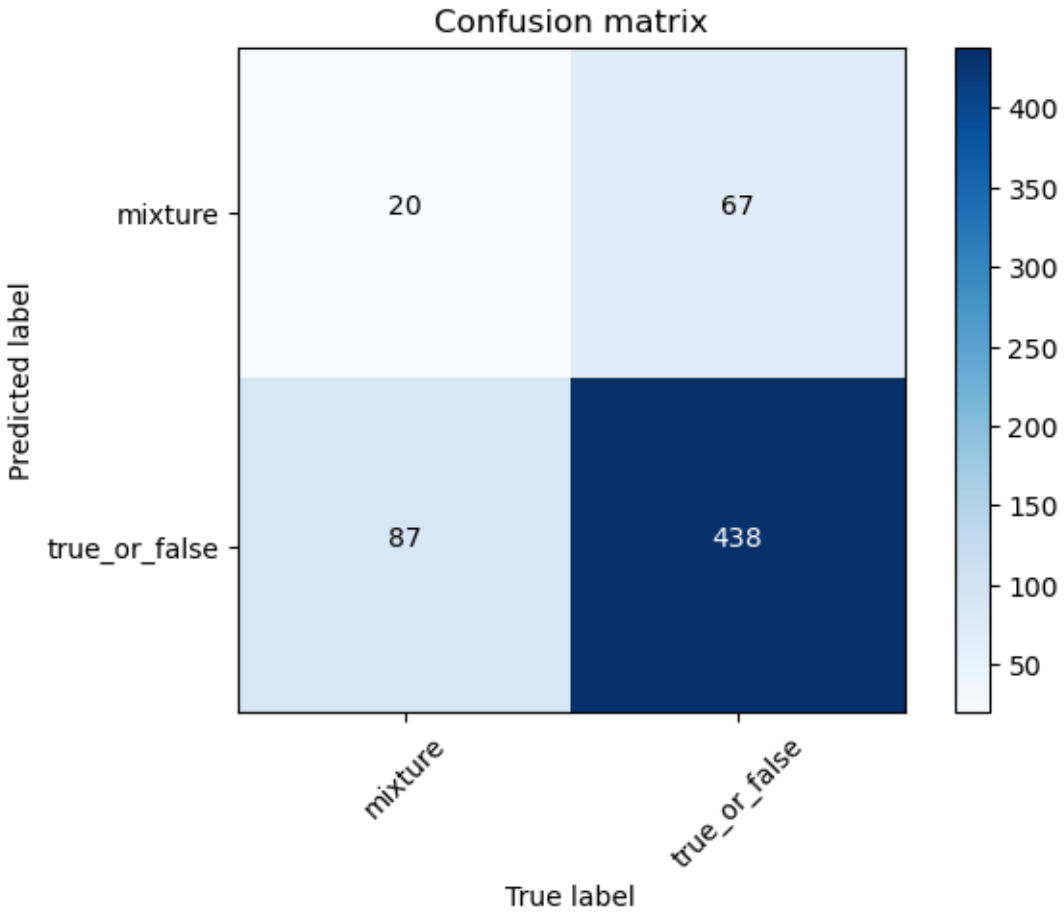


[63]:
```
X_test,y_test = df_test["text"],df_test["rating"]
y_pred = svm.predict(X_test)
print('\n accuracy: ', accuracy_score(y_pred, y_test),'\n')

print('\n',classification_report(y_test, y_pred))
plot_confusion_matrix(confusion_matrix(y_test, y_pred),classes = target_names)
```

```
 accuracy:  0.7483660130718954

                precision    recall  f1-score   support

       mixture       0.19      0.23      0.21        87
true_or_false       0.87      0.83      0.85       525

      accuracy                           0.75       612
     macro avg       0.53      0.53      0.53       612
```

```
 weighted avg       0.77      0.75      0.76       612
```

```
Confusion matrix, without normalization
```



3.1.8 MultinominalNB

[64]:
```
X = df_train["text"]
y = df_train["rating"]

X_s = X #  X.sample(300) # X if X.shape[0] <=700 else X.sample(700)
y_s = y.loc[X_s.index]
preprocess_selection("MuntinomialNB",MultinomialNB(),X_s,y_s)
```

```
Evaluation des différentes configurations :
Pipeline             Score
-----------------  -------
TFIDF_brut          0.7105
TFIDF_lowStop       0.7063
```

```
TFIDF_lowcase       0.7021
CV_brut             0.6968
TFIDF_lowStopstem   0.6947
TFIDF_lowStopna     0.6937
CV_lowcase          0.6884
CV_lowStop          0.6863
CV_lowStopstem      0.6832
CV_lowStopna        0.6821
```

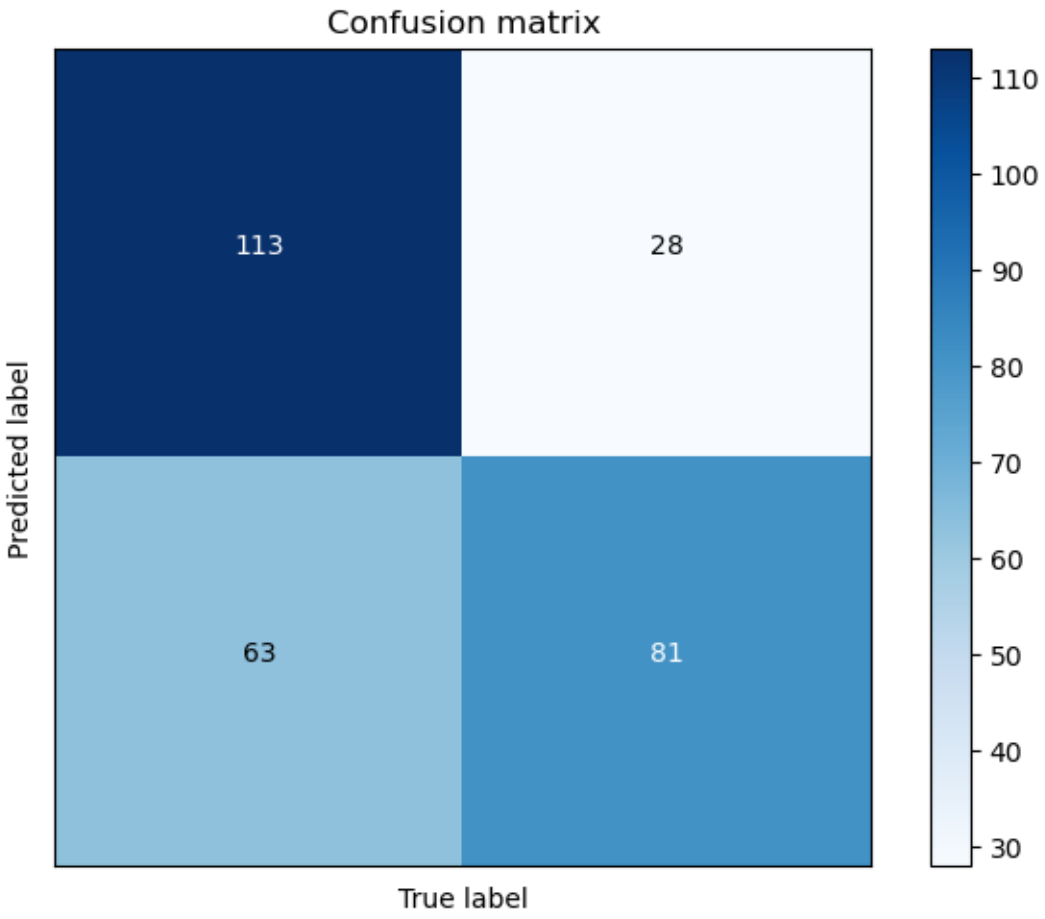
[65]:
```
X = df_train["text"]
y = df_train["rating"]
# put text normalizer in the pipe is a bad idea because of the very long time␣
 ↪processing
text_normalizer =␣
 ↪TextNormalizer(removestopwords=True,lowercase=True,getstemmer=True,removedigit=False)
X = text_normalizer.fit_transform(X)
X_train,X_val,y_train,y_val = train_test_split(X,y,train_size=0.
 ↪7,random_state=42)

pipe = Pipeline([('vect', TfidfVectorizer()),
                 ('clf', MultinomialNB()),
                ])

grid = {'clf__alpha': np.linspace(0.5, 1.5, 6),
        'clf__fit_prior': [True, False],}

gd_srMNB = GridSearchCV(pipe,
                        param_grid=grid,
                        scoring='accuracy',
                        cv=10,
#                         n_jobs=-1,
                       return_train_score=True)

gd_srMNB.fit(X_train, y_train)
print('meilleur score ',
 gd_srMNB.best_score_,'\n')
print('meilleurs paramètres',
 gd_srMNB.best_params_,'\n')
print('meilleur estimateur',
gd_srMNB.best_estimator_,'\n')
```

```
meilleur score  0.6797829036635006

meilleurs paramètres {'clf__alpha': 0.5, 'clf__fit_prior': False}

meilleur estimateur Pipeline(steps=[('vect', TfidfVectorizer()), ('tfidf',
TfidfTransformer()),
```

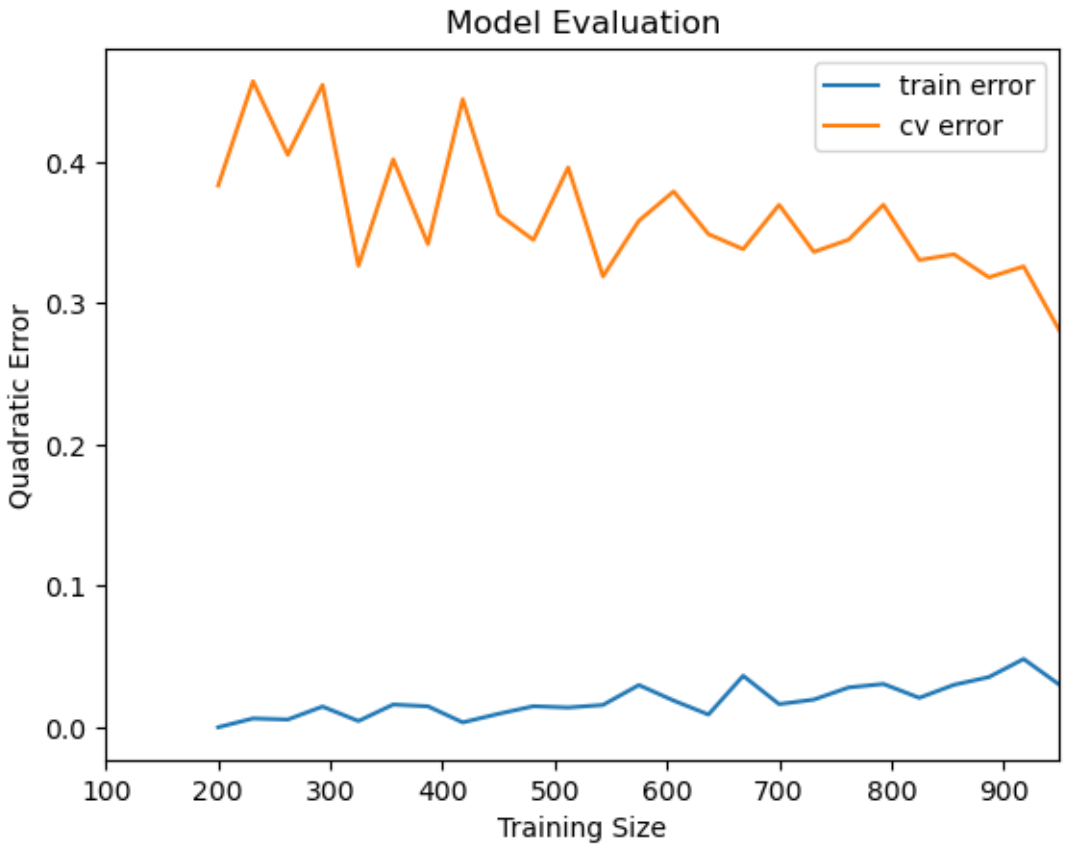
```
('clf', MultinomialNB(alpha=0.5, fit_prior=False))])
```

[66]:
```
#Creation d'une instance de l'algorithme en utilisant les meilleurs paramètres
mNB = gd_srMNB.best_estimator_

mNB.fit(X_train, y_train)
y_pred = mNB.predict(X_val)
print('\n accuracy: ', accuracy_score(y_pred, y_val),'\n')

print('\n',classification_report(y_val, y_pred))
plot_confusion_matrix(confusion_matrix(y_val, y_pred))
```

```
 accuracy:  0.6807017543859649


               precision    recall  f1-score   support

      mixture       0.64      0.80      0.71       141
true_or_false       0.74      0.56      0.64       144

     accuracy                           0.68       285
    macro avg       0.69      0.68      0.68       285
 weighted avg       0.69      0.68      0.68       285

Confusion matrix, without normalization
```



[67]:
```
X = df_train["text"]
y = df_train["rating"]
text_normalizer =␣
 ↪TextNormalizer(removestopwords=True,lowercase=True,getstemmer=True,removedigit=False)

X = X.reset_index(drop=True)
y = y.reset_index(drop=True)

X = text_normalizer.fit_transform(X)
X = pd.Series(X)
plot_quad_error(X,y,mNB)
```

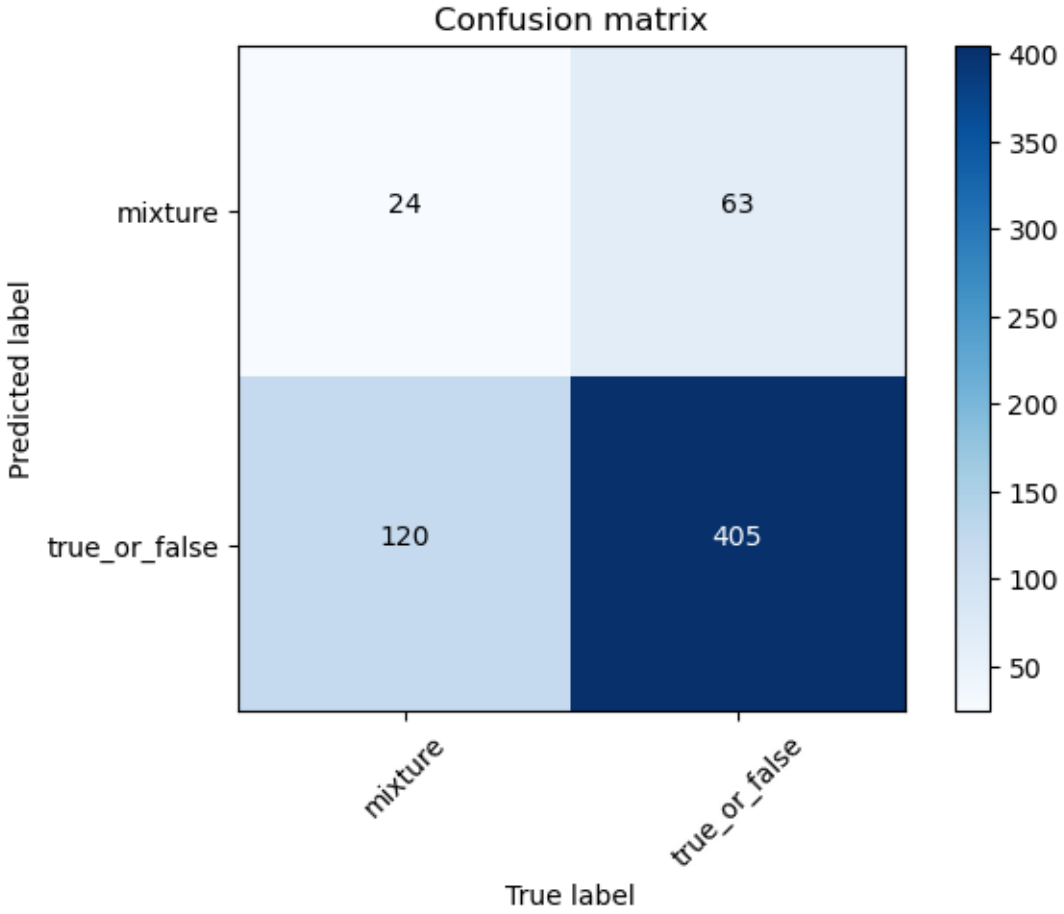


[68]:
```
X_test,y_test = df_test["text"],df_test["rating"]
y_pred = mNB.predict(X_test)
print('\n accuracy: ', accuracy_score(y_pred, y_test),'\n')

print('\n',classification_report(y_test, y_pred,target_names=classes))
plot_confusion_matrix(confusion_matrix(y_test, y_pred),classes = target_names)
```

```
 accuracy:  0.7009803921568627
```

```
                precision    recall  f1-score   support

 true_or_false       0.17      0.28      0.21        87
       mixture       0.87      0.77      0.82       525

      accuracy                           0.70       612
     macro avg       0.52      0.52      0.51       612
  weighted avg       0.77      0.70      0.73       612
```

```
Confusion matrix, without normalization
```



3.1.9 RandomForest

[69]:
```
X = df_train["text"]
y = df_train["rating"]

X_s = X # X.sample(300) # X if X.shape[0] <=700 else X.sample(700)
y_s = y.loc[X_s.index]
preprocess_selection("RandomForest",RandomForestClassifier(random_state=42),X_s,y_s)
```

```
Evaluation des différentes configurations :
Pipeline             Score
-----------------  -------
CV_lowStop          0.7084
TFIDF_lowcase       0.7084
CV_lowStopstem      0.7063
TFIDF_lowStopstem   0.6926
```

```
TFIDF_lowStop       0.6884
CV_lowStopna        0.6811
TFIDF_brut          0.6789
CV_lowcase          0.6747
CV_brut             0.6747
TFIDF_lowStopna     0.6705
```

[70]:
```
X = df_train["text"]
y = df_train["rating"]
# put text normalizer in the pipe is a bad idea because of the very long time␣
 ↪processing
text_normalizer =␣
 ↪TextNormalizer(removestopwords=True,lowercase=True,getstemmer=True,removedigit=False)
X = text_normalizer.fit_transform(X)
X_train,X_val,y_train,y_val = train_test_split(X,y,train_size=0.
 ↪7,random_state=42)

pipe = Pipeline([('vect', TfidfVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('clf', RandomForestClassifier()),
                ])



parameters = {
    'clf__n_estimators': list(range(50,125,25)),   #[500, 1200],
    'clf__max_depth': list(range(60,81,2)),    # [25, 30],
    'clf__min_samples_split': [5, 10, 15],
    'clf__min_samples_leaf' : [1, 2],
    }

gd_srMNB = GridSearchCV(pipe,
                        param_grid=parameters,
                        scoring='accuracy',
                        cv=10,
#                         n_jobs=-1,
                       return_train_score=True)

gd_srMNB.fit(X_train, y_train)
print('meilleur score ',
 gd_srMNB.best_score_,'\n')
print('meilleurs paramètres',
 gd_srMNB.best_params_,'\n')
print('meilleur estimateur',
gd_srMNB.best_estimator_,'\n')
```

```
meilleur score  0.6916553595658075
```

```
meilleurs paramètres {'clf__max_depth': 66, 'clf__min_samples_leaf': 2,
'clf__min_samples_split': 5, 'clf__n_estimators': 75}

meilleur estimateur Pipeline(steps=[('vect', TfidfVectorizer()), ('tfidf',
TfidfTransformer()),
                ('clf',
                 RandomForestClassifier(max_depth=66, min_samples_leaf=2,
                                        min_samples_split=5,
                                        n_estimators=75))])
```

[71]:
```
rfc = gd_srMNB.best_estimator_

rfc.fit(X_train, y_train)
y_pred = rfc.predict(X_val)
print('\n accuracy: ', accuracy_score(y_pred, y_val),'\n')

print('\n',classification_report(y_val, y_pred))
plot_confusion_matrix(confusion_matrix(y_val, y_pred,classes = target_names))
```

```
 accuracy:  0.6421052631578947


              precision    recall  f1-score   support

     mixture       0.63      0.65      0.64       141
true_or_false       0.65      0.63      0.64       144

    accuracy                           0.64       285
   macro avg       0.64      0.64      0.64       285
weighted avg       0.64      0.64      0.64       285
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_316857/2114611614.py in <module>
      6
      7 print('\n',classification_report(y_val, y_pred))
----> 8 plot_confusion_matrix(confusion_matrix(y_val, y_pred,classes =␣
 ↪target_names))

TypeError: confusion_matrix() got an unexpected keyword argument 'classes'
```

[ ]:
```
X = df_train["text"]
y = df_train["rating"]
text_normalizer =␣
 ↪TextNormalizer(removestopwords=True,lowercase=True,getstemmer=True,removedigit=False)

X = X.reset_index(drop=True)
y = y.reset_index(drop=True)

X = text_normalizer.fit_transform(X)
X = pd.Series(X)
plot_quad_error(X,y,rfc)
```

[ ]:
```
X_test,y_test = df_test["text"],df_test["rating"]
y_pred = rfc.predict(X_test)
print('\n accuracy: ', accuracy_score(y_pred, y_test),'\n')

print('\n',classification_report(y_test, y_pred))
plot_confusion_matrix(confusion_matrix(y_test, y_pred),classes = target_names)
```

[ ]:
```
# ('RandomForest', RandomForestClassifier())
```

[ ]:
```
models = pd.DataFrame({

    "Models": ["Logistic Regression" , "SVM", "Multinominal Naive␣
 ↪Bayes","Random Forest Classifier"],
    "Score":[lr.score(X_test,y_test), svm.score(X_test,y_test), mNB.
 ↪score(X_test,y_test), rfc.score(X_test,y_test)]

})
models.sort_values(by="Score" , ascending=False)
```

[ ]:
```

```