Tâche de classification {VRAI} vs {FAUX}

Etudiants : Cazeres Mathieu (22200082), Martin-Chantereau Etienne (21909526), Moreaux Victor (22200010), Poiret Valentin (21609227)

Chargement des librairies et des fonctions

```
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)

# fonctions utilities (affichage, confusion, etc.)
from Fonction.MyNLPUtilities import *

# fonctions (fonction de clean, import etc etc)
from Fonction.myFonction import *
from Fonction.AllModels import *
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\victo\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\victo\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\victo\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\victo\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\victo\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\victo\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Chargement des données équilibrés

```
#Importation du jeu de donné traité obtenu avec Traitement_data.ipynb
mySample = pd.read_csv('./Data_equilibre/balanced_data_VF.csv')

print(mySample['our rating'].value_counts())
X_train = mySample['text']
y_train = mySample['our rating']
```

```
True     500
False    500
Name: our rating, dtype: int64
```

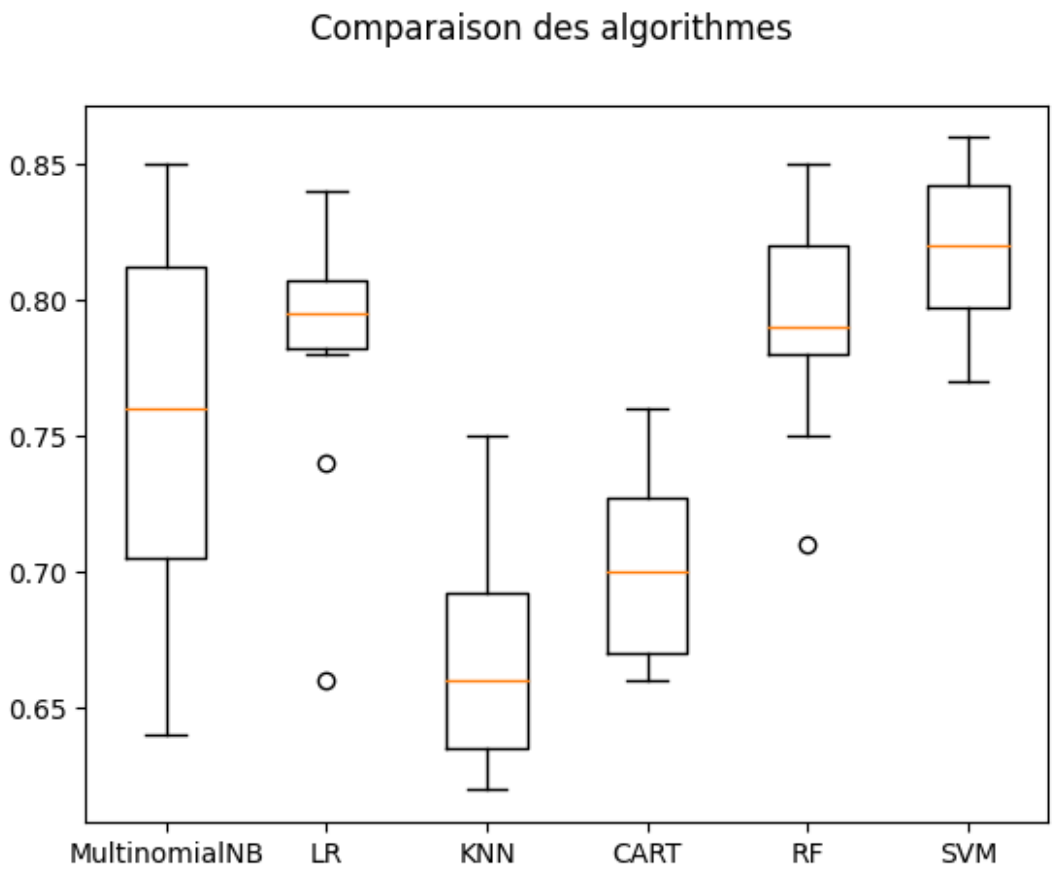
Test de tout les modèles

On teste tout les modèles de base pour voir lesquels sont le splus performants :

```
testAllModel(X_train,y_train,3)
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
```

```
Evaluation de  MultinomialNB
MultinomialNB : 0.758 (0.065) in 2.227 s
Evaluation de  LR
LR : 0.784 (0.049) in 21.474 s
Evaluation de  KNN
KNN : 0.668 (0.041) in 5.114 s
Evaluation de  CART
CART : 0.702 (0.035) in 35.351 s
Evaluation de  RF
RF : 0.790 (0.039) in 38.816 s
Evaluation de  SVM
SVM : 0.820 (0.029) in 249.748 s

Le meilleur resultat :
Classifier :  SVM  accuracy : 0.820  (0.029)  en 249.748   s

Tous les résultats :

Classifier :  SVM  accuracy : 0.820  (0.029)  en 249.748   s
Classifier :  RF  accuracy : 0.790  (0.039)  en 38.816   s
Classifier :  LR  accuracy : 0.784  (0.049)  en 21.474   s
Classifier :  MultinomialNB  accuracy : 0.758  (0.065)  en 2.227   s
Classifier :  CART  accuracy : 0.702  (0.035)  en 35.351   s
Classifier :  KNN  accuracy : 0.668  (0.041)  en 5.114   s
```



Les modèles SVM, LR, RF et MNB sont les plus performants pour la tâche de classification {VRAI} vs {FAUX}. Pour chacun de ces modèles, nous allons chercher les meilleurs paramètrages de prétraitement et les meilleurs paramètrages du modèle.

Modèle SVM (Support Vector Machine)

Recherche du meilleur paramètrage et sauvegarde des résultats de tout les paramètrages dans le fichier data_svc_VF:

```
testSVC(X_train,y_train,3,'data_svc_VF')
```

```
Application de gridsearch ...
pipeline : ['cleaner', 'tfidf', 'svm']
parameters :
{'cleaner__removedigit': [True, False], 'cleaner__getlemmatisation':
[True, False], 'tfidf__stop_words': ['english', None],
'tfidf__lowercase': [True, False], 'svm__C': [0.001, 0.01, 0.1, 1,
10], 'svm__gamma': [0.001, 0.01, 0.1, 1], 'svm__kernel': ['linear',
'rbf', 'poly', 'sigmoid']}

[nltk_data] Downloading package omw-1.4 to /root/nltk_data...

Fitting 3 folds for each of 1280 candidates, totalling 3840 fits
réalisé en  1384.504 s
```

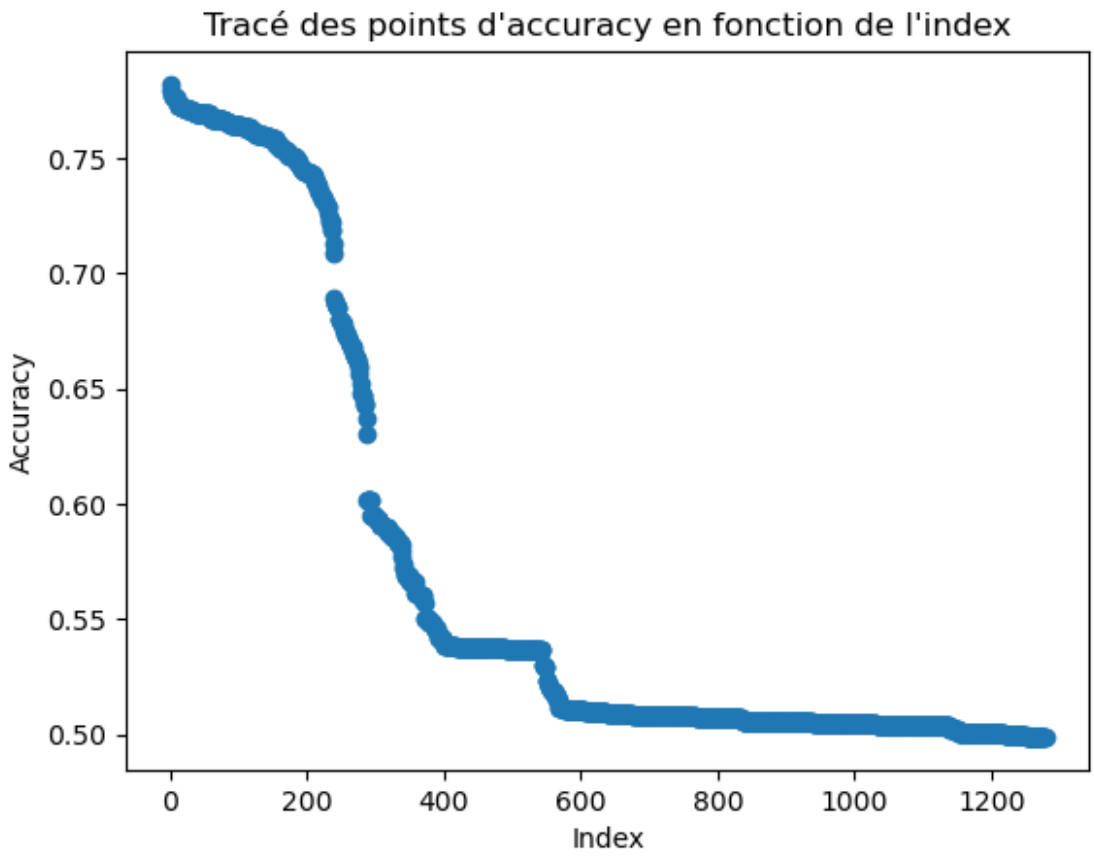
```
Meilleur résultat : 0.782
Ensemble des meilleurs paramètres :
      cleaner__getlemmatisation: False
      cleaner__removedigit: True
      svm__C: 10
      svm__gamma: 0.1
      svm__kernel: 'rbf'
      tfidf__lowercase: True
      tfidf__stop_words: None

Les premiers résultats :
       cleaner__getlemmatisation  cleaner__removedigit  svm__C
svm__gamma  \
933                        False                  True    10.0
0.10
1253                       False                 False    10.0
0.10
945                        False                  True    10.0
1.00
913                        False                  True    10.0
0.01
929                        False                  True    10.0
0.10

     svm__kernel  tfidf__lowercase tfidf__stop_words  accuracy
933          rbf              True              None  0.782064
1253         rbf              True              None  0.779069
945       linear              True              None  0.777076
913       linear              True              None  0.777076
929       linear              True              None  0.777076
```

Affichage de l'accuracy en fonction de l'index des différents paramètrages :

```python
import matplotlib.pyplot as plt
import pandas as pd
# Chargement des données
data = pd.read_csv('./Data_parametrage/data_svc_VF.csv')


display(data.head())

df = data
# Récupérer les valeurs de x et de y
x = df.index.values
y = df['accuracy']

# Tracer les points
plt.scatter(x, y)

# Ajouter un titre et des labels d'axe
```

```
plt.title("Tracé des points d'accuracy en fonction de l'index")
plt.xlabel('Index')
plt.ylabel('Accuracy')

# Afficher le plot
plt.show()
```

```
   cleaner__getlemmatisation  cleaner__removedigit  svm__C  svm__gamma
\
0                      False                  True    10.0        0.10

1                      False                 False    10.0        0.10

2                      False                  True    10.0        1.00

3                      False                  True    10.0        0.01

4                      False                  True    10.0        0.10


  svm__kernel  tfidf__lowercase tfidf__stop_words  accuracy
0         rbf              True               NaN  0.782064
1         rbf              True               NaN  0.779069
2      linear              True               NaN  0.777076
3      linear              True               NaN  0.777076
4      linear              True               NaN  0.777076
```



Affichage des points de rupture sur la courbe et de la proportion de certains paramètres pour chaque segment :

```
from numpy import NaN
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
!pip install ruptures
import ruptures as rpt

# Charger les données
data = pd.read_csv('./Data_parametrage/data_svc_VF.csv')

data.fillna('vide', inplace=True)
display(data.head())
# Sélectionner les colonnes à analyser
y_colonne = 'accuracy'

signal = data[y_colonne].values
model = "l2"
algo = rpt.Window(width=55, model=model, jump=1).fit(signal)
result = algo.predict(n_bkps=3)
```

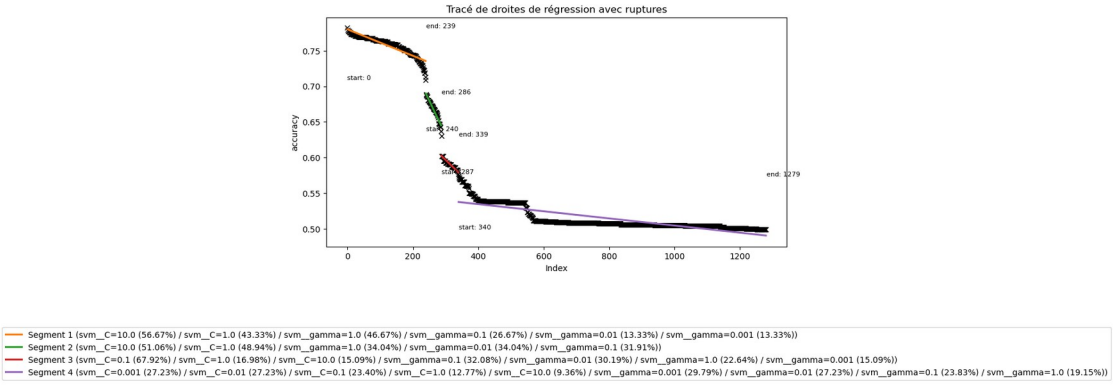
```
fig, ax = plt.subplots(figsize=(10, 5))
# Define a list of parameters to show in the legend
params = ['svm__C','svm__gamma']

# Plot the data points
ax.plot(data.index, data[y_colonne], 'x', color='black')
my_row=[]
for i, (start, end) in enumerate(zip([0] + result, result +
[len(signal)])):
    segment = data.iloc[start:end]

    # Calculate the proportion of each unique value in the selected
columns for this segment
    param_props = []
    for param in params:
        param_value_counts =
segment[param].value_counts(normalize=True)
        param_value_props = [f"{count*100:.2f}%" for count in
param_value_counts]
        param_value_legend = " / ".join([f"{param}={param_value}
({param_value_props[j]})"
                                         for j, param_value in
enumerate(param_value_counts.index)])
        param_props.append(param_value_legend)

    # Join the legends for each parameter into one legend for the
segment
    segment_legend = " / ".join(param_props)

    # Plot the regression line for this segment with the corresponding
color and legend
    sns.regplot(x=segment.index, y=y_colonne, data=segment, ax=ax,
color=f'C{i+1}',
                label=f'Segment {i+1} ({segment_legend})',
scatter=False)

    # Add text to show the start and end of each segment
    if(start != len(data[y_colonne])):
      ax.text(start, segment[y_colonne].min(), f'start: {start}',
fontsize=8)
      if start not in my_row:
        my_row.append(start)
    if(end-1 != len(data[y_colonne])):
      ax.text(end, segment[y_colonne].max(), f'end: {end-1}',
fontsize=8)
      if end-1 not in my_row:
        my_row.append(end-1)


# Set the axis labels and title
```

```
ax.set_xlabel('Index')
ax.set_ylabel(y_colonne)
ax.set_title('Tracé de droites de régression avec ruptures')

# Hide the current legend
ax.legend(loc='lower center', bbox_to_anchor=(0.5, -0.6), ncol=1)


plt.show()

fig.savefig('nom_du_fichier.png', dpi=300, bbox_inches='tight')
```

```
Collecting ruptures
  Downloading ruptures-1.1.7-cp39-cp39-win_amd64.whl (383 kB)
     ------------------------------------ 383.3/383.3 kB 2.2 MB/s
eta 0:00:00
Requirement already satisfied: scipy in c:\users\victo\anaconda3\lib\
site-packages (from ruptures) (1.9.1)
Requirement already satisfied: numpy in c:\users\victo\anaconda3\lib\
site-packages (from ruptures) (1.21.5)
Installing collected packages: ruptures
Successfully installed ruptures-1.1.7
```

	cleaner__getlemmatisation	cleaner__removedigit	svm__C	svm__gamma	svm__kernel	tfidf__lowercase	tfidf__stop_words	accuracy
0	False	True	10.0	0.10	rbf	True	vide	0.782064
1	False	False	10.0	0.10	rbf	True	vide	0.779069
2	False	True	10.0	1.00	linear	True	vide	0.777076
3	False	True	10.0	0.01	linear	True	vide	0.777076
4	False	True	10.0	0.10	linear	True	vide	0.777076

```
posx and posy should be finite values
posx and posy should be finite values
```



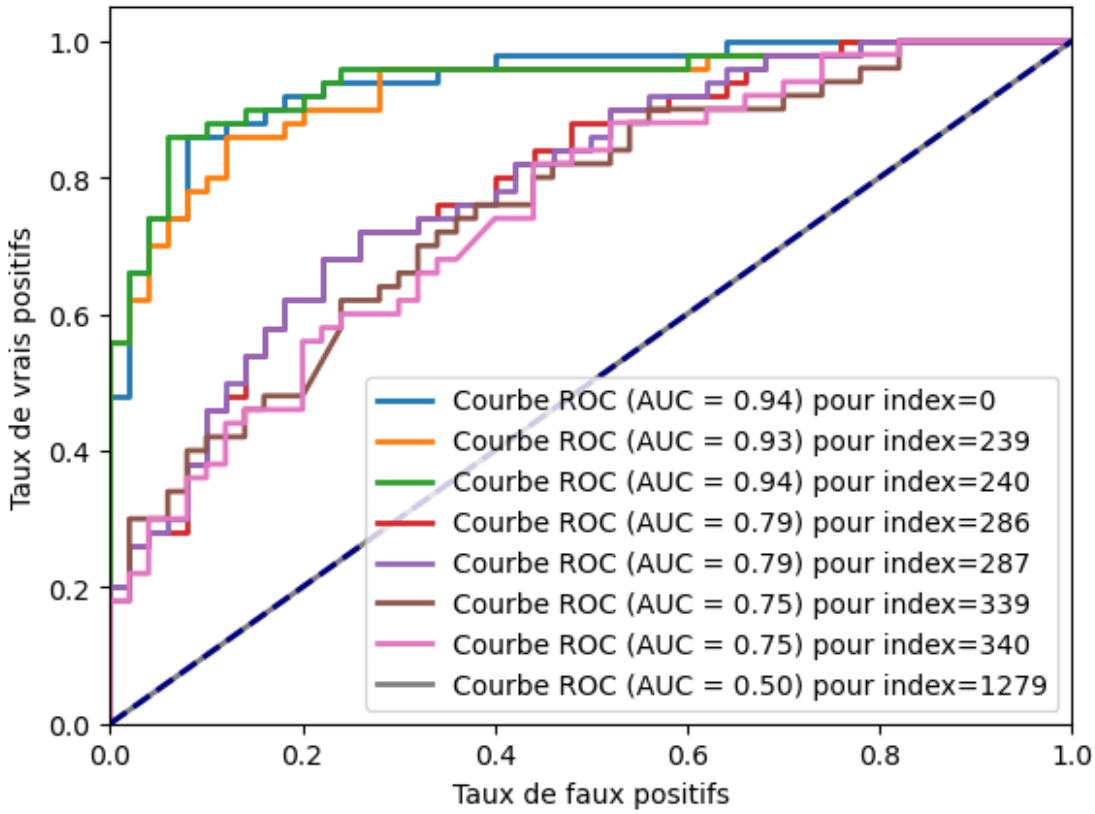

```
posx and posy should be finite values
posx and posy should be finite values
```

Affichage des courbes de ROC pour les extrèmes des différents segments :

```
# Charger les données
data = pd.read_csv('./Data_parametrage/data_svc_VF.csv')

data.fillna('vide', inplace=True)
display(data.head())

df_selection = data.loc[my_row]


print(df_selection)

import pickle
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Création d'un jeu d'apprentissage et de test
trainsize=0.9 # 90% pour le jeu d'apprentissage, il reste 30% du jeu
de données pour
testsize= 0.1
seed=30
train_title,test_title,train_note,test_note=train_test_split(X_train,y
_train,
train_size=trainsize,random_state=seed,test_size=testsize,stratify=y_t
rain)
# Initialiser une liste pour stocker les résultats de la prédiction
pour chaque pipeline
y_pred_probas = []
plt.figure()

for index, row in df_selection.iterrows():
```

```
  stopswords = 'english' if row.cleaner__getlemmatisation is None else
None

  pipeline=Pipeline([
      ("cleaner", TextNormalizer(removedigit=row.cleaner__removedigit,
getlemmatisation=row.cleaner__getlemmatisation)),
      ("tfidf", TfidfVectorizer(lowercase=row.tfidf__lowercase,
stop_words=stopswords)),
        ('svm', SVC(C=row.svm__C, gamma=row.svm__gamma,
kernel=row.svm__kernel,probability=True)) ####################
  ])
  # Entraîner le modèle avec le jeu d'apprentissage
  pipeline.fit(train_title, train_note)

  # Test avec les données qu'il a apprises
  y_pred = pipeline.predict(test_title)

  # Calcul de la courbe ROC
  y_pred_proba = pipeline.predict_proba(test_title)[:,1]
  fpr, tpr, thresholds = roc_curve(test_note, y_pred_proba)
  roc_auc = auc(fpr, tpr)

  # Tracé de la courbe ROC

  plt.plot(fpr, tpr, lw=2, label='Courbe ROC (AUC = %0.2f) pour index=
%d' % (roc_auc, index))


plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Taux de faux positifs')
plt.ylabel('Taux de vrais positifs')
# plt.title('Courbe ROC')
plt.legend(loc="lower right")
plt.show()


   cleaner__getlemmatisation  cleaner__removedigit  svm__C  svm__gamma
\
0                      False                  True    10.0        0.10

1                      False                 False    10.0        0.10

2                      False                  True    10.0        1.00

3                      False                  True    10.0        0.01

4                      False                  True    10.0        0.10
```

```
  svm__kernel  tfidf__lowercase tfidf__stop_words  accuracy
0         rbf              True              vide  0.782064
1         rbf              True              vide  0.779069
2      linear              True              vide  0.777076
3      linear              True              vide  0.777076
4      linear              True              vide  0.777076

      cleaner__getlemmatisation  cleaner__removedigit  svm__C
svm__gamma  \
0                         False                  True  10.000
0.1
239                        True                 False   1.000
1.0
240                       False                  True  10.000
1.0
286                        True                 False   1.000
0.1
287                        True                  True   1.000
0.1
339                       False                 False   0.100
1.0
340                        True                  True   0.100
1.0
1279                       True                 False   0.001
1.0

     svm__kernel  tfidf__lowercase tfidf__stop_words  accuracy
0            rbf              True              vide  0.782064
239         poly             False              vide  0.709077
240         poly              True           english  0.689063
286          rbf             False           english  0.637107
287          rbf             False           english  0.630100
339      sigmoid              True              vide  0.577044
340      sigmoid             False              vide  0.574029
1279      linear              True           english  0.498999
```



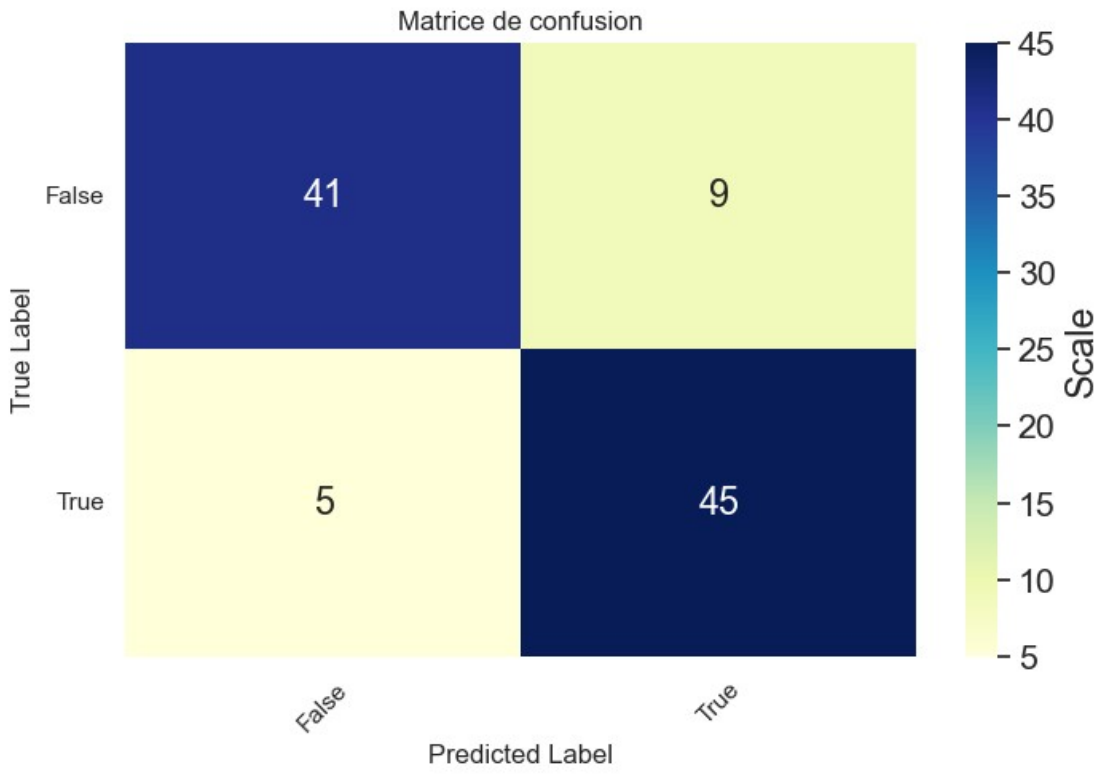
Enregistrement du modele SVC avec les meilleurs paramètres déterminés et affichage de la matrice de confusion et de la courbe ROC correspondante :

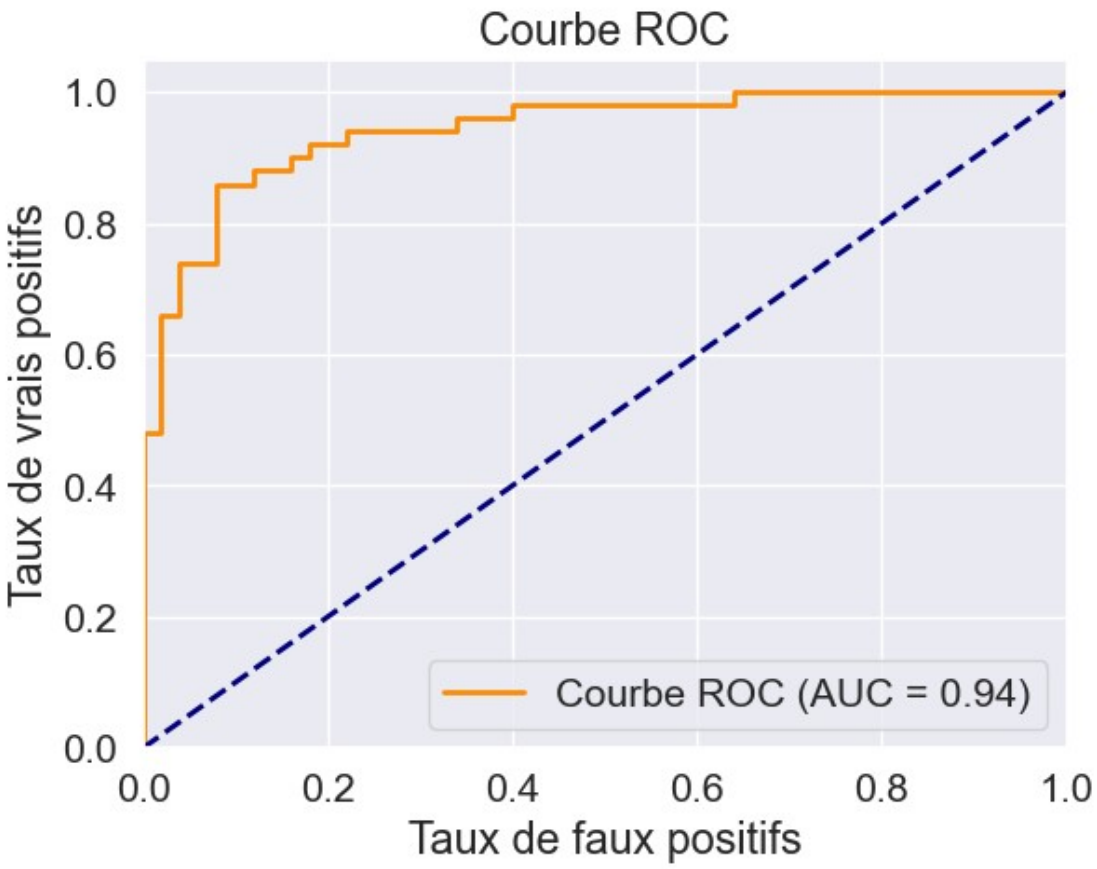
```
from sklearn.model_selection import train_test_split
import pickle
# Création d'un jeu d'apprentissage et de test
trainsize=0.9 # 70% pour le jeu d'apprentissage, il reste 30% du jeu
de données pour
testsize= 0.1
seed=30

train_title,test_title,train_note,test_note=train_test_split(X_train,y
_train,
train_size=trainsize,random_state=seed,test_size=testsize,stratify=y_t
rain)

pipeline=Pipeline([
    ("cleaner", TextNormalizer(removedigit=True,
getlemmatisation=False)),
    ("tfidf", TfidfVectorizer(lowercase=True, stop_words=None)),
    ('svm', SVC(C=10, gamma=0.1, kernel='rbf',probability=True))
])
pipeline.fit(train_title,train_note)
filename='./Modele/VF_svm.pkl'
print("Sauvegarde du modèle dans ", filename)
```

```
pickle.dump(pipeline, open(filename, "wb"))


print ("Chargement du modèle \n")
# le chargement se fait via la fonction load
clf_loaded = pickle.load(open(filename, 'rb'))
# affichage du modèle sauvegardé
print (clf_loaded)


# test  avec les données qu'il a apprise c'est parfait woahhha c'est
beau
y_pred = clf_loaded.predict(test_title)
# autres mesures et matrice de confusion
MyshowAllScores(test_note,y_pred)

# Calcul de la courbe ROC
y_pred_proba = clf_loaded.predict_proba(test_title)[:,1]
fpr, tpr, thresholds = roc_curve(test_note, y_pred_proba)
roc_auc = auc(fpr, tpr)

# Tracé de la courbe ROC
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='Courbe ROC (AUC =
%0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Taux de faux positifs')
plt.ylabel('Taux de vrais positifs')
plt.title('Courbe ROC')
plt.legend(loc="lower right")
plt.show()
```

```
Sauvegarde du modèle dans  ./Modele/VF_svm.pkl
Chargement du modèle

Pipeline(steps=[('cleaner', TextNormalizer(removedigit=True)),
                ('tfidf', TfidfVectorizer()),
                ('svm', SVC(C=10, gamma=0.1, probability=True))])
Accuracy : 0.860
Classification Report
              precision    recall  f1-score   support

       False    0.89130   0.82000   0.85417        50
        True    0.83333   0.90000   0.86538        50

    accuracy                        0.86000       100
   macro avg    0.86232   0.86000   0.85978       100
```

```
weighted avg    0.86232   0.86000   0.85978       100
```





Modèle LR (LogisticRegression)

Recherche du meilleur paramètrage et sauvegarde des résultats de tout les paramètrages dans le fichier data_lr_VF:

```
testLR(X_train,y_train,3,'data_lr_VF')
```

```
Application de gridsearch ...
pipeline : ['cleaner', 'tfidf', 'lr']
parameters :
{'cleaner__removedigit': [True, False], 'cleaner__getlemmatisation':
[True, False], 'tfidf__stop_words': ['english', None],
'tfidf__lowercase': [True, False], 'lr__solver': ['newton-cg',
'lbfgs', 'liblinear'], 'lr__penalty': ['l2'], 'lr__C': [100, 10, 1.0,
0.1, 0.01]}
Fitting 3 folds for each of 240 candidates, totalling 720 fits
réalisé en  222.677 s
Meilleur résultat : 0.778
Ensemble des meilleurs paramètres :
      cleaner__getlemmatisation: False
      cleaner__removedigit: False
      lr__C: 100
      lr__penalty: 'l2'
      lr__solver: 'newton-cg'
```

```
     tfidf__lowercase: True
     tfidf__stop_words: None

Les premiers résultats :
      cleaner__getlemmatisation  cleaner__removedigit  lr__C
lr__penalty  \
181                       False                 False  100.0
l2
185                       False                 False  100.0
l2
189                       False                 False  100.0
l2
73                         True                 False   10.0
l2
77                         True                 False   10.0
l2

    lr__solver  tfidf__lowercase tfidf__stop_words  accuracy
181  newton-cg              True              None  0.778065
185      lbfgs              True              None  0.778065
189  liblinear              True              None  0.778065
73   newton-cg              True              None  0.778056
77       lbfgs              True              None  0.778056
```

Affichage des points de rupture sur la courbe et de la proportion du paramètre C du modèle LR pour chaque segment :

```
from matplotlib import patches
from numpy import NaN
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
!pip install ruptures
import ruptures as rpt

# Charger les données
data = pd.read_csv('./Data_parametrage/data_lr_VF.csv')

data.fillna('vide', inplace=True)
display(data.head())
# Sélectionner les colonnes à analyser
y_colonne = 'accuracy'

signal = data[y_colonne].values
model = "l2"
algo = rpt.Window(width=40, model=model, jump=1).fit(signal)
result = algo.predict(n_bkps=2)

fig, ax = plt.subplots(figsize=(10, 5))
```

```
#Tu choisis la liste des param ici a visualiser je te conseil de le
faire en deux fois


params =
['lr__C']#'cleaner__removedigit','cleaner__getlemmatisation','tfidf__s
top_words','tfidf__lowercase','lr__solver'] #

l = []
for param in params:
  p = data[param].value_counts(normalize=True)
  for i in range(len(p)):
    x = p.index.tolist()[i]
    l.append(str(param)+"="+str(x))


mydf = pd.DataFrame(columns=l,index=[0, 1, 2])
print(mydf)



# Plot the data points
ax.plot(data.index, data[y_colonne], 'x', color='black')
my_row=[]
for i, (start, end) in enumerate(zip([0] + result, result +
[len(signal)])):
    segment = data.iloc[start:end]

    # Calculate the proportion of each unique value in the selected
columns for this segment
    param_props = []
    for param in params:
        param_value_counts =
segment[param].value_counts(normalize=True)

        param_value_props = [f"{count*100:.2f}%" for count in
param_value_counts]

        param_value_legend = " / ".join([f"{param}={param_value}
({param_value_props[j]})"
                                         for j, param_value in
enumerate(param_value_counts.index)])

        param_props.append(param_value_legend)

        for j, param_value in enumerate(param_value_counts.index):
          k=str(param)+"="+str(param_value)
```

```
            # print(k)
            param_value_counts_df = param_value_counts.reset_index()
            param_value_counts_df =
param_value_counts_df.rename(columns={param: 'Parametre', 0:
'Pourcentage'})
            # print(param_value_counts_df.loc[j, 'Parametre'])
            mydf.loc[i][k]=param_value_counts_df.loc[j, 'Parametre']

    # Join the legends for each parameter into one legend for the
segment
    segment_legend = " / ".join(param_props)

    # Plot the regression line for this segment with the corresponding
color and legend
    sns.regplot(x=segment.index, y=y_colonne, data=segment, ax=ax,
color=f'C{i+1}',
                label=f'Segment {i+1}', scatter=False)

    # Add text to show the start and end of each segment
    if(start != len(data[y_colonne])):
        ax.text(start, segment[y_colonne].min(), f'start: {start}',
fontsize=8)
        if start not in my_row:
            my_row.append(start)
    if(end-1 != len(data[y_colonne])):
        ax.text(end, segment[y_colonne].max(), f'end: {end-1}',
fontsize=8)
        if end-1 not in my_row:
            my_row.append(end-1)

    # Set the values of the corresponding row in mydf to the
parameters in this segment




# d = pd.DataFrame(my_param, columns=nom_col)

# Set the axis labels and title
ax.set_xlabel('Index du paramètrage')
ax.set_ylabel(y_colonne)
# ax.set_title('Tracé de droites de régression avec ruptures des
paramètrage différents du modèle en fonction de l\'accuracy')

# Hide the current legend
ax.legend(loc='lower center', bbox_to_anchor=(0.5, -0.6), ncol=1)


plt.show()
```

```
fig.savefig('data_lr_1.png', dpi=300, bbox_inches='tight')
```

```
# créer le graphique
fig, ax = plt.subplots(figsize=(8, 6))
mydf.plot(kind='bar', ax=ax)

# ajouter des étiquettes
# ax.set_title('Proportion des paramètres sans impact pour le modèle
LogisticRegression pour chacun des segments')
ax.set_xlabel('Segment')
ax.set_ylabel('Proportion des paramètres')
legend = ax.legend(ncol=1)

# afficher le graphique
plt.show()
```

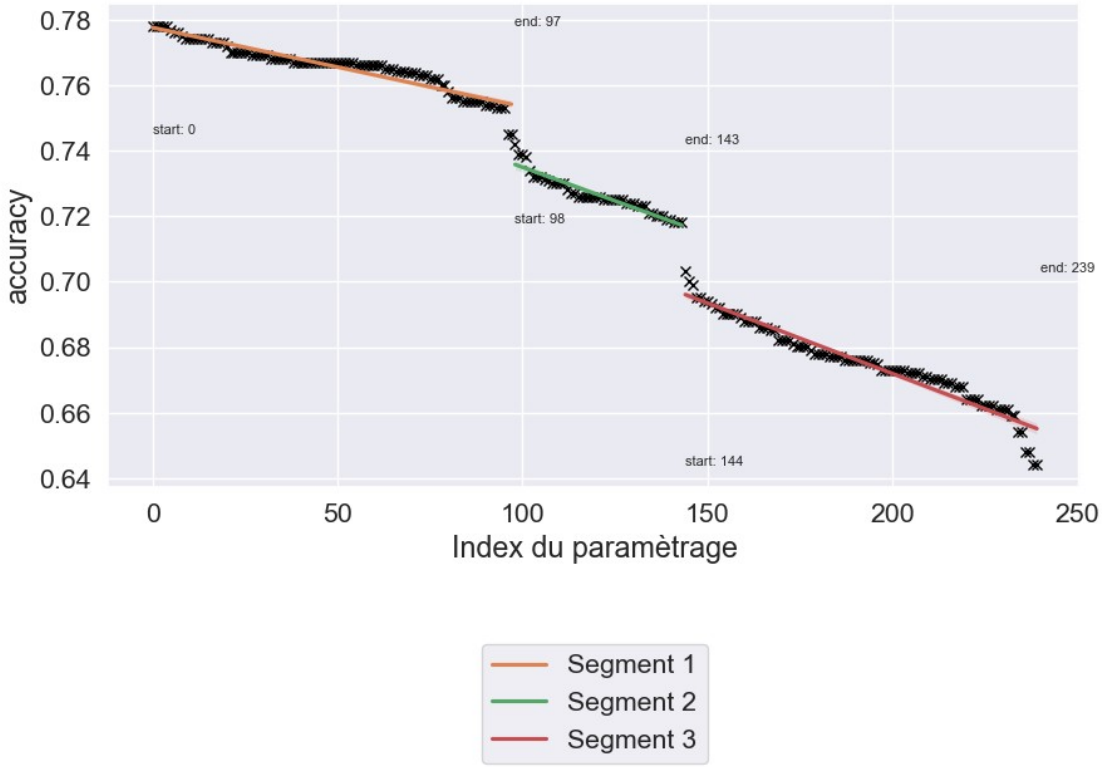
```
Requirement already satisfied: ruptures in c:\users\victo\anaconda3\
lib\site-packages (1.1.7)
Requirement already satisfied: numpy in c:\users\victo\anaconda3\lib\
site-packages (from ruptures) (1.21.5)
Requirement already satisfied: scipy in c:\users\victo\anaconda3\lib\
site-packages (from ruptures) (1.9.1)
```

	cleaner__getlemmatisation	cleaner__removedigit	lr__C	lr__penalty
0	False	False	100.0	l2
1	False	False	100.0	l2
2	False	False	100.0	l2
3	True	False	10.0	l2
4	True	False	10.0	l2

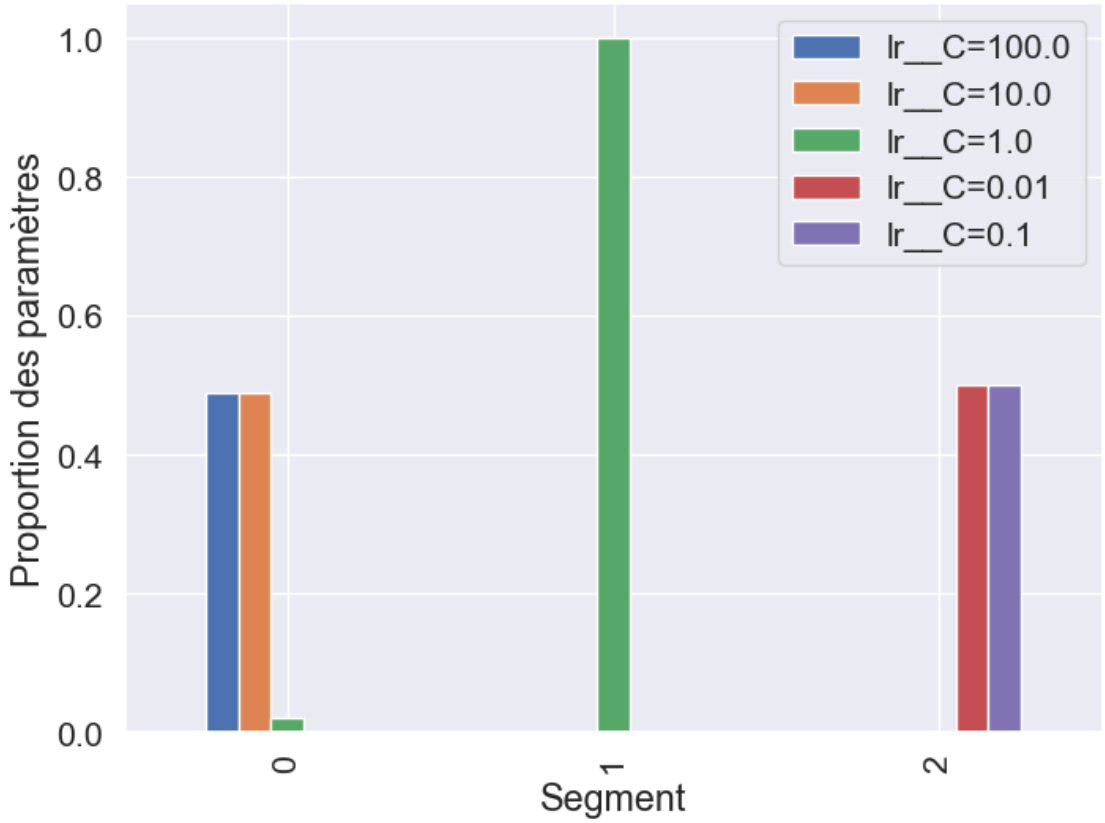
	lr__solver	tfidf__lowercase	tfidf__stop_words	accuracy
0	newton-cg	True	vide	0.778065
1	lbfgs	True	vide	0.778065
2	liblinear	True	vide	0.778065
3	newton-cg	True	vide	0.778056
4	lbfgs	True	vide	0.778056

	lr__C=100.0	lr__C=10.0	lr__C=1.0	lr__C=0.01	lr__C=0.1
0	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN

```
posx and posy should be finite values
posx and posy should be finite values
```




```
posx and posy should be finite values
posx and posy should be finite values
```

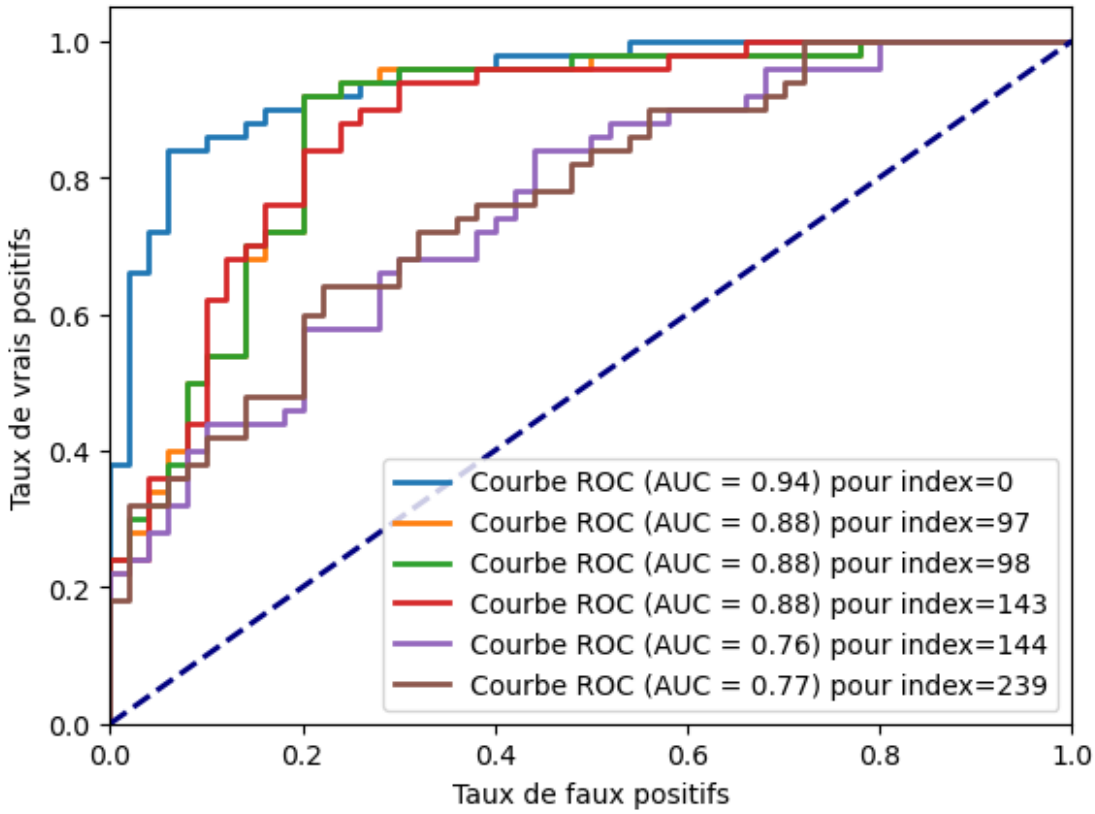


Affichage des courbes de ROC pour les points extrèmes des différents segments :

```
# Charger les données
data = pd.read_csv('./Data_parametrage/data_lr_VF.csv')

data.fillna('vide', inplace=True)
display(data.head())

df_selection = data.loc[my_row]


print(df_selection)

import pickle
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Création d'un jeu d'apprentissage et de test
trainsize=0.9 # 70% pour le jeu d'apprentissage, il reste 30% du jeu
de données pour
testsize= 0.1
```

```
seed=30
train_title,test_title,train_note,test_note=train_test_split(X_train,y
_train,
train_size=trainsize,random_state=seed,test_size=testsize,stratify=y_t
rain)
# Initialiser une liste pour stocker les résultats de la prédiction
pour chaque pipeline
y_pred_probas = []
plt.figure()

for index, row in df_selection.iterrows():

  stopswords = 'english' if row.cleaner__getlemmatisation is None else
None

  pipeline=Pipeline([
      ("cleaner", TextNormalizer(removedigit=row.cleaner__removedigit,
getlemmatisation=row.cleaner__getlemmatisation)),
      ("tfidf", TfidfVectorizer(lowercase=row.tfidf__lowercase,
stop_words=stopswords)),
      ('lr', LogisticRegression(C=row.lr__C,penalty='l2',
solver=row.lr__solver))
  ])
  # Entraîner le modèle avec le jeu d'apprentissage
  pipeline.fit(train_title, train_note)

  # Test avec les données qu'il a apprises
  y_pred = pipeline.predict(test_title)

  # Calcul de la courbe ROC
  y_pred_proba = pipeline.predict_proba(test_title)[:,1]
  fpr, tpr, thresholds = roc_curve(test_note, y_pred_proba)
  roc_auc = auc(fpr, tpr)

  # Tracé de la courbe ROC

  plt.plot(fpr, tpr, lw=2, label='Courbe ROC (AUC = %0.2f) pour index=
%d' % (roc_auc, index))


plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Taux de faux positifs')
plt.ylabel('Taux de vrais positifs')
# plt.title('Courbe ROC')
plt.legend(loc="lower right")
plt.show()
```

```
   cleaner__getlemmatisation  cleaner__removedigit  lr__C lr__penalty
\
0                      False                 False  100.0          l2

1                      False                 False  100.0          l2

2                      False                 False  100.0          l2

3                       True                 False   10.0          l2

4                       True                 False   10.0          l2


  lr__solver  tfidf__lowercase tfidf__stop_words  accuracy
0  newton-cg              True              vide  0.778065
1      lbfgs              True              vide  0.778065
2  liblinear              True              vide  0.778065
3  newton-cg              True              vide  0.778056
4      lbfgs              True              vide  0.778056

     cleaner__getlemmatisation  cleaner__removedigit   lr__C
lr__penalty  \
0                       False                 False  100.00
l2
97                      False                  True    1.00
l2
98                      False                  True    1.00
l2
143                      True                 False    1.00
l2
144                     False                  True    0.01
l2
239                      True                 False    0.01
l2

    lr__solver  tfidf__lowercase tfidf__stop_words  accuracy
0    newton-cg              True              vide  0.778065
97       lbfgs             False           english  0.745074
98   liblinear             False           english  0.742074
143  liblinear              True              vide  0.718068
144  liblinear             False           english  0.703074
239  newton-cg              True           english  0.644093
```



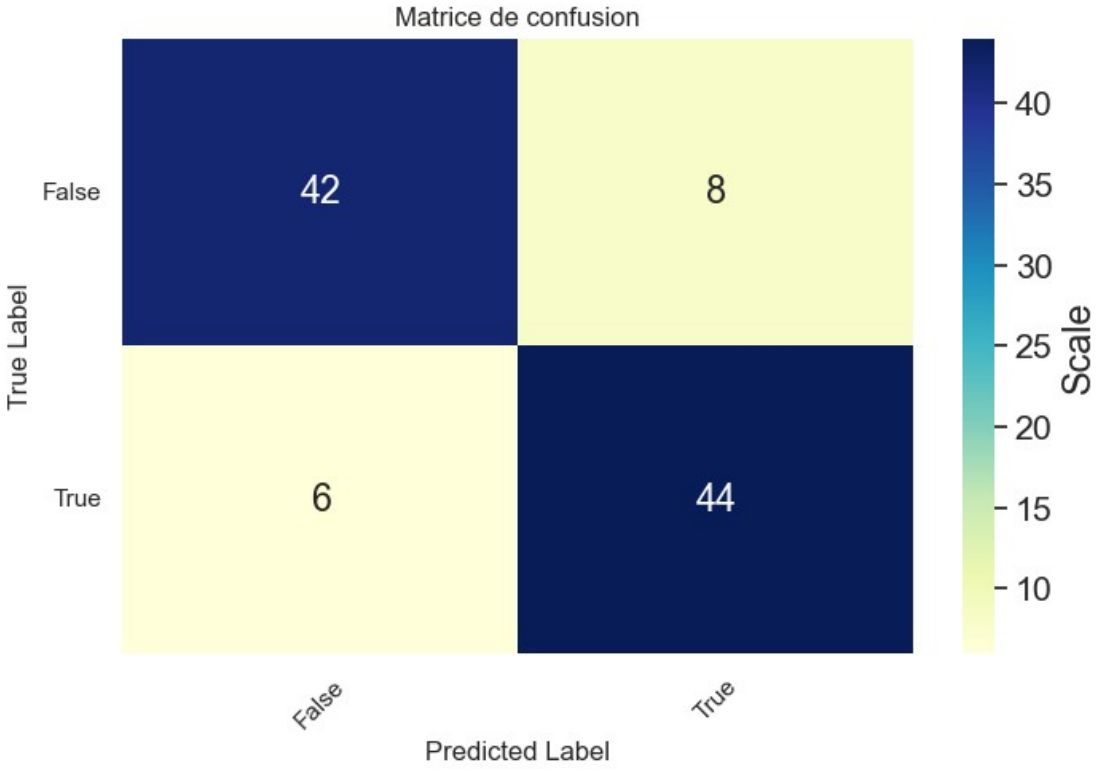
Enregistrement du modele LR avec les meilleurs paramètres déterminés et affichage de la matrice de confusion et de la courbe ROC correspondante : Affichage en nuage de point des texte utilisé pour le test du classifieur LR dans le but de visualiser une eventuelle tendance des articles qui ont été mal labélisés par le modèle (peu concluant)

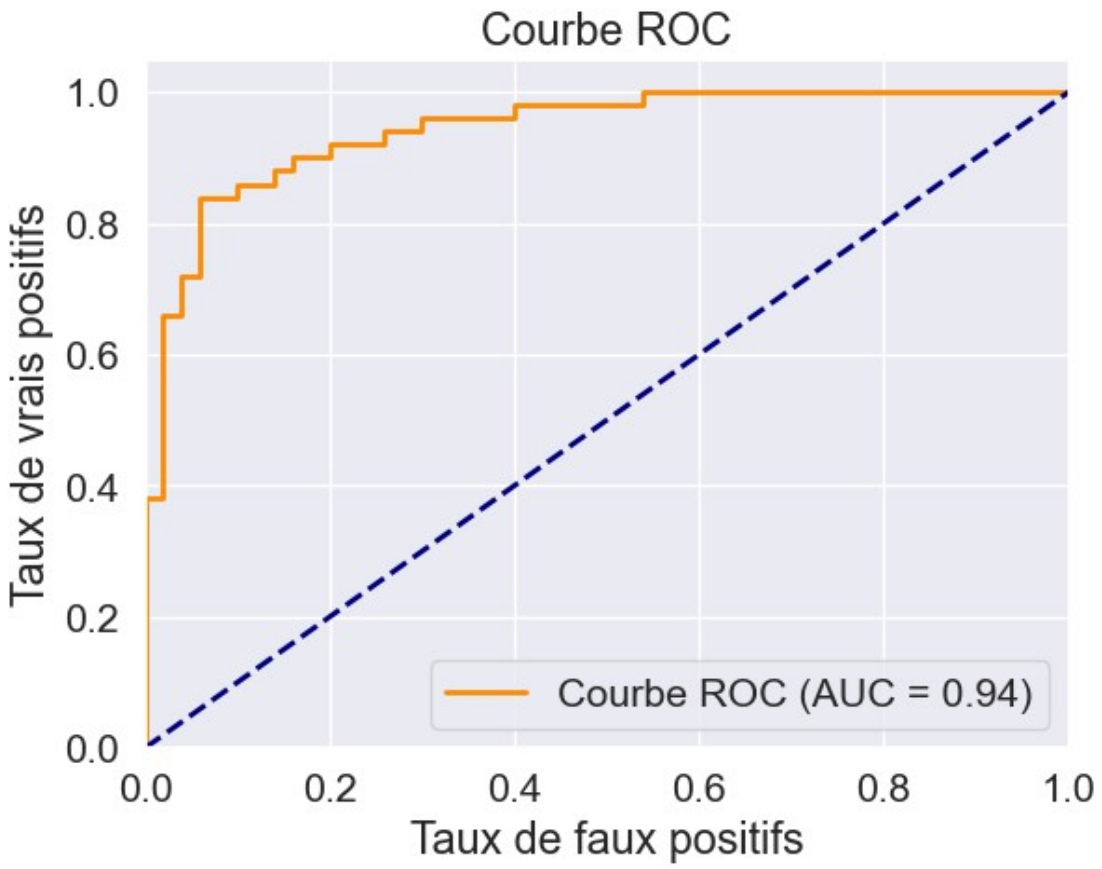
```
import pickle
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Création d'un jeu d'apprentissage et de test
trainsize=0.9 # 90% pour le jeu d'apprentissage, il reste 30% du jeu
de données pour
testsize= 0.1
seed=30
train_title,test_title,train_note,test_note=train_test_split(X_train,y
_train,
train_size=trainsize,random_state=seed,test_size=testsize,stratify=y_t
rain)

pipeline=Pipeline([
    ("cleaner", TextNormalizer(removedigit=False,
getlemmatisation=False)),
    ("tfidf", TfidfVectorizer(lowercase=True, stop_words=None)),
```

```
    ('lr', LogisticRegression(C=100,penalty='l2', solver='newton-cg'))
])
pipeline.fit(train_title,train_note)
filename='./Modele/VF_LR.pkl'
print("Sauvegarde du modèle dans ", filename)
pickle.dump(pipeline, open(filename, "wb"))

print ("Chargement du modèle \n")
# le chargement se fait via la fonction load
clf_loaded = pickle.load(open(filename, 'rb'))
# affichage du modèle sauvegardé
print (clf_loaded)

# Test avec les données qu'il a apprises
y_pred = clf_loaded.predict(test_title)
# autres mesures et matrice de confusion
MyshowAllScores(test_note,y_pred)

# Calcul de la courbe ROC
y_pred_proba = clf_loaded.predict_proba(test_title)[:,1]
fpr, tpr, thresholds = roc_curve(test_note, y_pred_proba)
roc_auc = auc(fpr, tpr)

# Tracé de la courbe ROC
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='Courbe ROC (AUC =
%0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Taux de faux positifs')
plt.ylabel('Taux de vrais positifs')
plt.title('Courbe ROC')
plt.legend(loc="lower right")
plt.show()


new_class_data = pd.DataFrame(np.column_stack((test_title,
y_pred,test_note)), columns=['title', 'predictions','note'])


# # concatenate the two dataframes
# new_class_data = pd.concat([test_title, predictions_df],ignore_index
= True)

new_class_data =
new_class_data.loc[~new_class_data['predictions'].isin(["false","true"
])]
# print(new_class_data)
```

```python
import pandas as pd
import numpy as np
from sklearn.manifold import TSNE
import plotly.express as px

text_normalizer = TextNormalizer(getlemmatisation=True,
removedigit=True, removestopwords=True)
cleaned_text = text_normalizer.fit_transform(new_class_data["title"])
tfidf = TfidfVectorizer(lowercase=False)
vector_tfidf = tfidf.fit_transform(cleaned_text)
tsne = TSNE(n_components=2, random_state=42)
projections = tsne.fit_transform(vector_tfidf.toarray())

# Ajoutez une colonne pour indiquer si la prédiction du modèle est
correcte ou non
new_class_data["correct"] = (new_class_data["predictions"] ==
new_class_data["note"])

# Tracez un graphique en utilisant Plotly pour représenter les
projections obtenues avec des couleurs différentes pour les
prédictions correctes et incorrectes
fig = px.scatter(x=projections[:,0], y=projections[:,1],
color=new_class_data["correct"],symbol=new_class_data["note"])
fig.show()
```

```
Sauvegarde du modèle dans  ./Modele/VF_LR.pkl
Chargement du modèle

Pipeline(steps=[('cleaner', TextNormalizer()), ('tfidf',
TfidfVectorizer()),
                ('lr', LogisticRegression(C=100, solver='newton-
cg'))])
Accuracy : 0.860
Classification Report
              precision    recall  f1-score   support

       False    0.87500   0.84000   0.85714        50
        True    0.84615   0.88000   0.86275        50

    accuracy                        0.86000       100
   macro avg    0.86058   0.86000   0.85994       100
weighted avg    0.86058   0.86000   0.85994       100
```


Matrice de confusion
42
8
6
44
False
True
True Label
False
True
Predicted Label
Scale
40
35
30
25
20
15
10


Courbe ROC
Taux de vrais positifs
Taux de faux positifs
Courbe ROC (AUC = 0.94)
0.0
0.2
0.4
0.6
0.8
1.0

{"config":{"plotlyServerURL":"https://plot.ly"},"data":
[{"hovertemplate":"color=True
symbol=True
x=%{x}
y=%
{y}<extra></extra>","legendgroup":"True, True","marker":
{"color":"#636efa","symbol":"circle"},"mode":"markers","name":"True,
True","orientation":"v","showlegend":true,"type":"scatter","x":
[18.65099334716797,6.379286766052246,-17.610458374023438,-
12.99390697479248,-23.791200637817383,-3.739166498184204,-
19.97856903076172,62.201080322265625,1.682145595550537,-
33.828182220458984,-
13.812026023864746,3.269655704498291,40.570064544677734,14.360267639160156,6.0622453689575195,-10.537105560302734,16.186107635498047,-
57.1298828125,71.30026245117188,56.976165771484375,35.946006774902344,
-6.068255424499512,-56.075782775878906,49.66289520263672,-
18.331926345825195,46.43513488769531,12.864662170410156,-
32.62465286254883,52.39387512207031,7.204028129577637,61.765132904052734,29.809162139892578,39.708152770996094,13.27916145324707,-
40.23502731323242,-26.094755172729492,-7.546300888061523,-
23.068588256835938,21.73410987854004,64.82970428466797,-
35.32258987426758,-
14.95065975189209,24.66248893737793,4.4791364669799805],"xaxis":"x","y
":[36.70009231567383,-49.38399887084961,51.9490966796875,-
8.2394380569458,-16.72513198852539,-10.01863956451416,-
34.74696350097656,9.216795921325684,26.241098403930664,-
18.072999954223633,-
46.3700065612793,3.5367913246154785,17.43186378479004,13.166376113891602,-17.794029235839844,-62.3141975402832,-21.426591873168945,-
44.70981979370117,25.75661277770996,30.255977630615234,47.27922821044922,-47.244869232177734,-
36.0462760925293,16.638357162475586,10.735478401184082,10.613355636596
68,-5.602511405944824,-49.14399337768555,22.54631996154785,-
7.268466949462891,14.805536270141602,-6.934234142303467,-
39.045169830322266,-61.43376922607422,14.031494140625,-
5.717067718505859,-70.7194595336914,-
26.23095703125,22.78176498413086,22.181591033935547,-
68.7920150756836,24.69699478149414,30.91002655029297,-
54.65536117553711],"yaxis":"y"},
{"hovertemplate":"color=True
symbol=False
x=%{x}
y=%
{y}<extra></extra>","legendgroup":"True, False","marker":
{"color":"#636efa","symbol":"diamond"},"mode":"markers","name":"True,
False","orientation":"v","showlegend":true,"type":"scatter","x":[-
30.43089485168457,-8.126972198486328,-
8.714815139770508,24.121322631835938,-
35.17521286010742,49.91487121582031,-11.939254760742188,-
13.739880561828613,-40.531986236572266,36.05718994140625,-
1.8099108934402466,-47.639949798583984,1.4770019054412842,-
55.690948486328125,36.83745574951172,51.09287643432617,-
11.211864471435547,-4.524508476257324,-39.09783172607422,-
54.314178466796875,48.89133834838867,39.324806213378906,-
60.78513717651367,-1.1909270286560059,-
51.343135833740234,18.87948989868164,-18.355937957763672,-

60.29901123046875,-11.915267944335938,9.365164756774902,-29.093040466308594,39.7319450378418,29.53276252746582,2.4538848400115967,-30.284015655517578,22.97905158996582,-46.61501693725586,-50.23202896118164,1.7725086212158203,3.1346330642700195,-32.646549224853516,-16.834928512573242],"xaxis":"x","y":[17.169437408447266,2.3548808097839355,66.61836242675781,-33.68879699707031,-29.60721778869629,-14.22297477722168,8.336541175842285,-20.377607345581055,5.551934242248535,-24.127046585083008,-1.8487650156021118,-26.00308609008789,-32.22806930541992,-24.86608123779297,-8.784356117248535,-31.62596321105957,35.62009811401367,14.021341323852539,-6.142336845397949,10.771904945373535,3.9183342456817627,-32.126220703125,-16.316204071044922,42.35692596435547,-43.02511215209961,-16.612348556518555,-60.3431396484375,-10.223228454589844,17.588638305664062,-36.478492736816406,28.7120418548584,4.743452548980713,-21.065969467163086,-42.61800765991211,5.688126087188721,-51.937950134277344,-6.271275997161865,32.7244987487793,13.01318645477295,21.126150131225586,38.40319061279297,1.5605391263961792],"yaxis":"y"},{"hovertemplate":"color=False
symbol=True
x=%{x}
y=%{y}<extra></extra>","legendgroup":"False, True","marker":{"color":"#EF553B","symbol":"circle"},"mode":"markers","name":"False, True","orientation":"v","showlegend":true,"type":"scatter","x":[26.64445686340332,-4.045104503631592,-9.236483573913574,-6.959585189819336,-5.006954669952393,-44.936161041259766],"xaxis":"x","y":[9.396905899047852,-22.12198257446289,-27.585182189941406,38.936466217041016,-35.95514678955078,19.08760643005371],"yaxis":"y"},{"hovertemplate":"color=False
symbol=False
x=%{x}
y=%{y}<extra></extra>","legendgroup":"False, False","marker":{"color":"#EF553B","symbol":"diamond"},"mode":"markers","name":"False, False","orientation":"v","showlegend":true,"type":"scatter","x":[-26.77147102355957,14.987166404724121,46.88753890991211,25.75765609741211,12.920439720153809,20.825727462768555,-31.331966400146484,15.51702880859375],"xaxis":"x","y":[-22.380613327026367,-43.05506896972656,28.395458221435547,14.510045051574707,4.417087078094482,-4.331374168395996e-2,-42.27190399169922,40.77279281616211],"yaxis":"y"}],"layout":{"legend":{"title":{"text":"color, symbol"},"tracegroupgap":0},"margin":{"t":60},"template":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":{"color":"#2a3f5f"},"marker":{"line":{"color":"#E5ECF6","width":0.5},"pattern":{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpolar":[{"marker":{"line":{"color":"#E5ECF6","width":0.5},"pattern":{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"carpet":[{"aaxis":{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min

```
orgridcolor":"white","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"type":"carpet"}],"ch
oropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
```

```
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"#E5ECF6","showlake
s":true,"showland":true,"subunitcolor":"white"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"#E5ECF6","po
lar":{"angularaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","radialaxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
```

lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"yaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"zaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"baxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","caxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2}}},"xaxis":
{"anchor":"y","domain":[0,1],"title":{"text":"x"}},"yaxis":
{"anchor":"x","domain":[0,1],"title":{"text":"y"}}}}

Modèle RFC (RandomForestClassifier)

Recherche du meilleur paramètrage et sauvegarde des résultats de tout les paramètrages dans le fichier data_rfc_VF:

```
testRFC(X_train,y_train,3,'data_rfc_VF')
```

```
Application de gridsearch ...
pipeline : ['cleaner', 'tfidf', 'rfc']
parameters :
{'cleaner__removedigit': [True, False], 'cleaner__getlemmatisation':
[True, False], 'tfidf__stop_words': ['english', None],
'tfidf__lowercase': [True, False], 'rfc__n_estimators': [500, 1200],
'rfc__max_depth': [25, 30], 'rfc__min_samples_split': [5, 10, 15],
'rfc__min_samples_leaf': [1, 2]}
Fitting 3 folds for each of 384 candidates, totalling 1152 fits
réalisé en  624.752 s
Meilleur résultat : 0.785
Ensemble des meilleurs paramètres :
     cleaner__getlemmatisation: True
     cleaner__removedigit: True
     rfc__max_depth: 30
     rfc__min_samples_leaf: 1
     rfc__min_samples_split: 5
     rfc__n_estimators: 1200
     tfidf__lowercase: False
     tfidf__stop_words: 'english'

Les premiers résultats :
```

```
      cleaner__getlemmatisation  cleaner__removedigit  rfc__max_depth
\
54                         True                  True              30

7                          True                  True              25

246                       False                  True              30

151                        True                 False              30

13                         True                  True              25


     rfc__min_samples_leaf  rfc__min_samples_split  rfc__n_estimators
\
54                       1                       5               1200

7                        1                       5               1200

246                      1                       5               1200

151                      1                       5               1200

13                       1                      10               1200


     tfidf__lowercase tfidf__stop_words  accuracy
54              False           english  0.785084
7               False              None  0.785064
246             False           english  0.783062
151             False              None  0.782066
13               True              None  0.781077
```

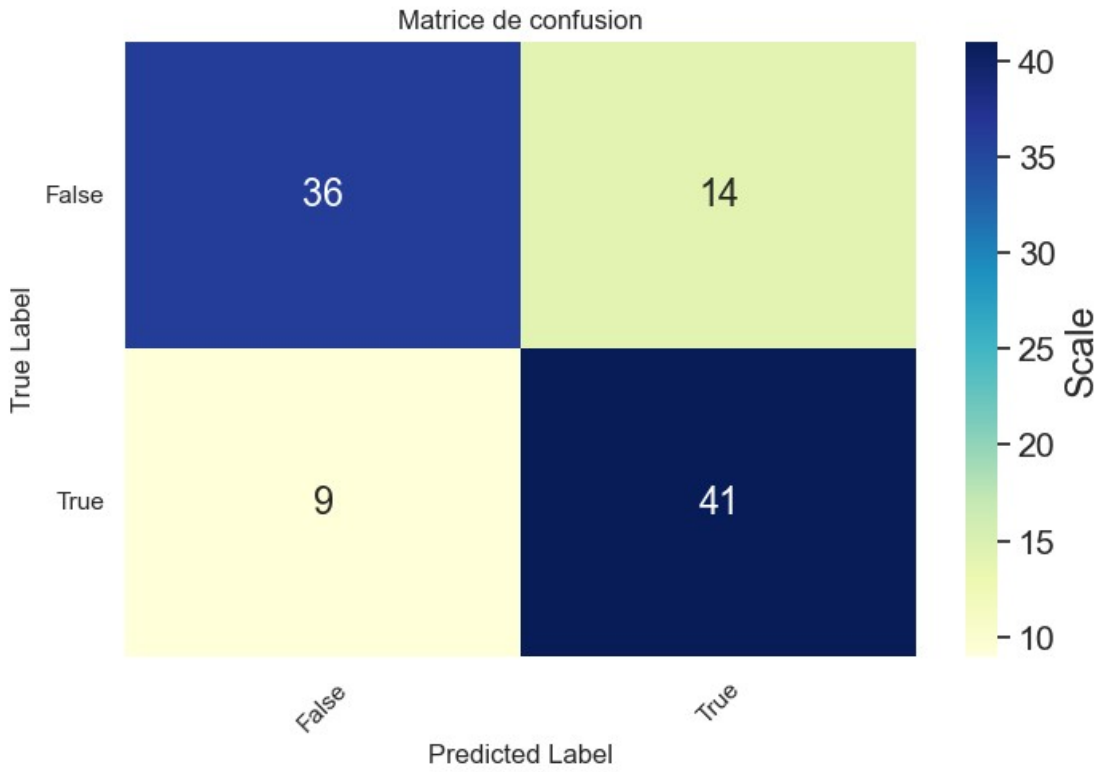
Enregistrement du modele RFC avec les meilleurs paramètres déterminés et affichage de la matrice de confusion et de la courbe ROC correspondante :

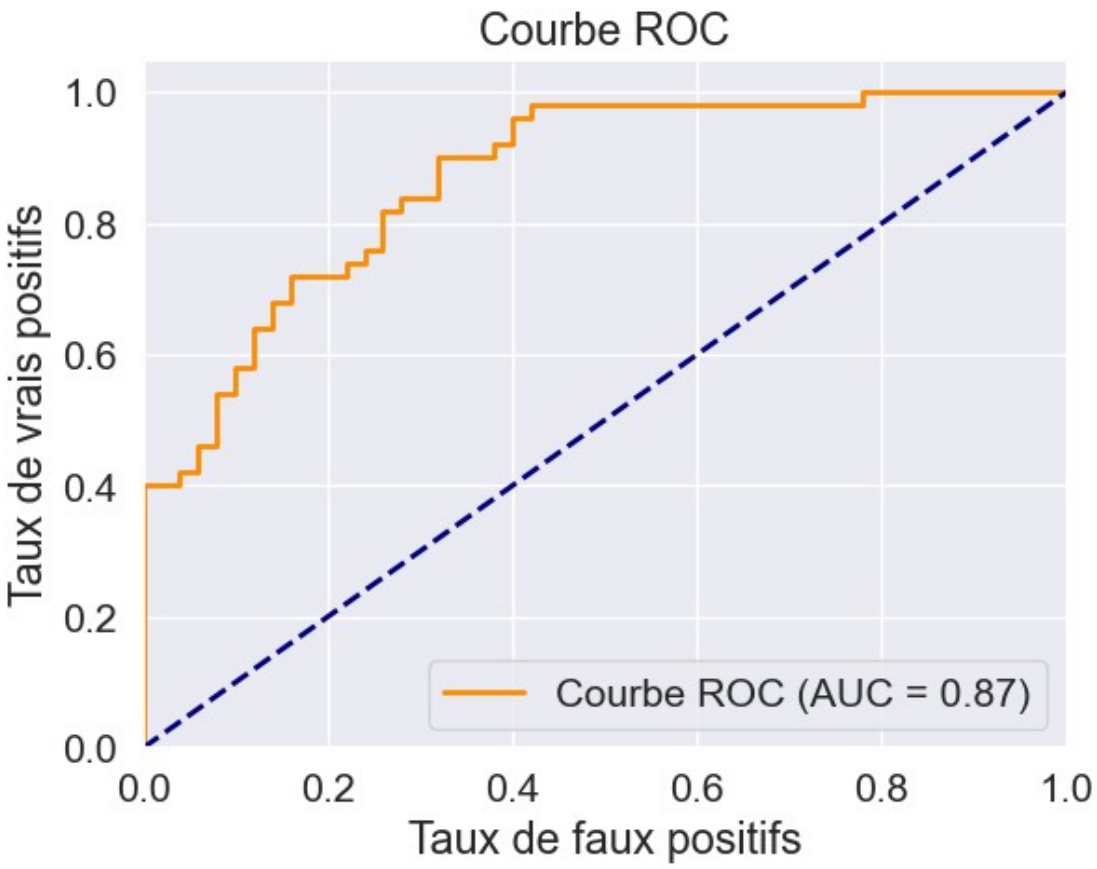
```
import pickle
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
# Création d'un jeu d'apprentissage et de test
trainsize=0.9 # 70% pour le jeu d'apprentissage, il reste 30% du jeu
de données pour
testsize= 0.1
seed=30
train_title,test_title,train_note,test_note=train_test_split(X_train,y
_train,
train_size=trainsize,random_state=seed,test_size=testsize,stratify=y_t
rain)
```

```
pipeline=Pipeline([
    ("cleaner", TextNormalizer(removedigit=True,
getlemmatisation=True)),
    ("tfidf", TfidfVectorizer(lowercase=True, stop_words='english')),
    ('rfc', RandomForestClassifier(max_depth=30,min_samples_leaf=1,
min_samples_split=5, n_estimators=1200))
])
pipeline.fit(train_title,train_note)
filename='./Modele/VF_rfc.pkl'
print("Sauvegarde du modèle dans ", filename)
pickle.dump(pipeline, open(filename, "wb"))



print ("Chargement du modèle \n")
# le chargement se fait via la fonction load
clf_loaded = pickle.load(open(filename, 'rb'))
# affichage du modèle sauvegardé
print (clf_loaded)


# test  avec les données qu'il a apprise c'est parfait woahhha c'est
beau
y_pred = clf_loaded.predict(test_title)
# autres mesures et matrice de confusion
MyshowAllScores(test_note,y_pred)



# Calcul de la courbe ROC
y_pred_proba = clf_loaded.predict_proba(test_title)[:,1]
fpr, tpr, thresholds = roc_curve(test_note, y_pred_proba)
roc_auc = auc(fpr, tpr)

# Tracé de la courbe ROC
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='Courbe ROC (AUC =
%0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Taux de faux positifs')
plt.ylabel('Taux de vrais positifs')
plt.title('Courbe ROC')
plt.legend(loc="lower right")
plt.show()
```

```
Sauvegarde du modèle dans  ./Modele/VF_rfc.pkl
Chargement du modèle
```

```
Pipeline(steps=[('cleaner',
                 TextNormalizer(getlemmatisation=True,
removedigit=True)),
                ('tfidf', TfidfVectorizer(stop_words='english')),
                ('rfc',
                 RandomForestClassifier(max_depth=30,
min_samples_split=5,
                                        n_estimators=1200))])
Accuracy : 0.770
Classification Report
              precision    recall  f1-score   support

       False    0.80000   0.72000   0.75789        50
        True    0.74545   0.82000   0.78095        50

    accuracy                        0.77000       100
   macro avg    0.77273   0.77000   0.76942       100
weighted avg    0.77273   0.77000   0.76942       100
```





Modèle MNB (MultinomialNB)

Recherche du meilleur paramètrage et sauvegarde des résultats de tout les paramètrages dans le fichier data_mnb_VF:

```
testMNB(X_train,y_train,3,'data_mnb_VF')
```

```
Application de gridsearch ...
pipeline : ['cleaner', 'tfidf', 'mnb']
parameters :
{'cleaner__removedigit': [True, False], 'cleaner__getlemmatisation':
[True, False], 'tfidf__stop_words': ['english', None],
'tfidf__lowercase': [True, False], 'mnb__alpha': array([0.5, 0.7, 0.9,
1.1, 1.3, 1.5]), 'mnb__fit_prior': [True, False], 'mnb__force_alpha':
[True, False]}
Fitting 3 folds for each of 384 candidates, totalling 1152 fits
réalisé en  351.543 s
Meilleur résultat : 0.749
Ensemble des meilleurs paramètres :
      cleaner__getlemmatisation: False
      cleaner__removedigit: True
      mnb__alpha: 0.5
      mnb__fit_prior: True
      mnb__force_alpha: True
```

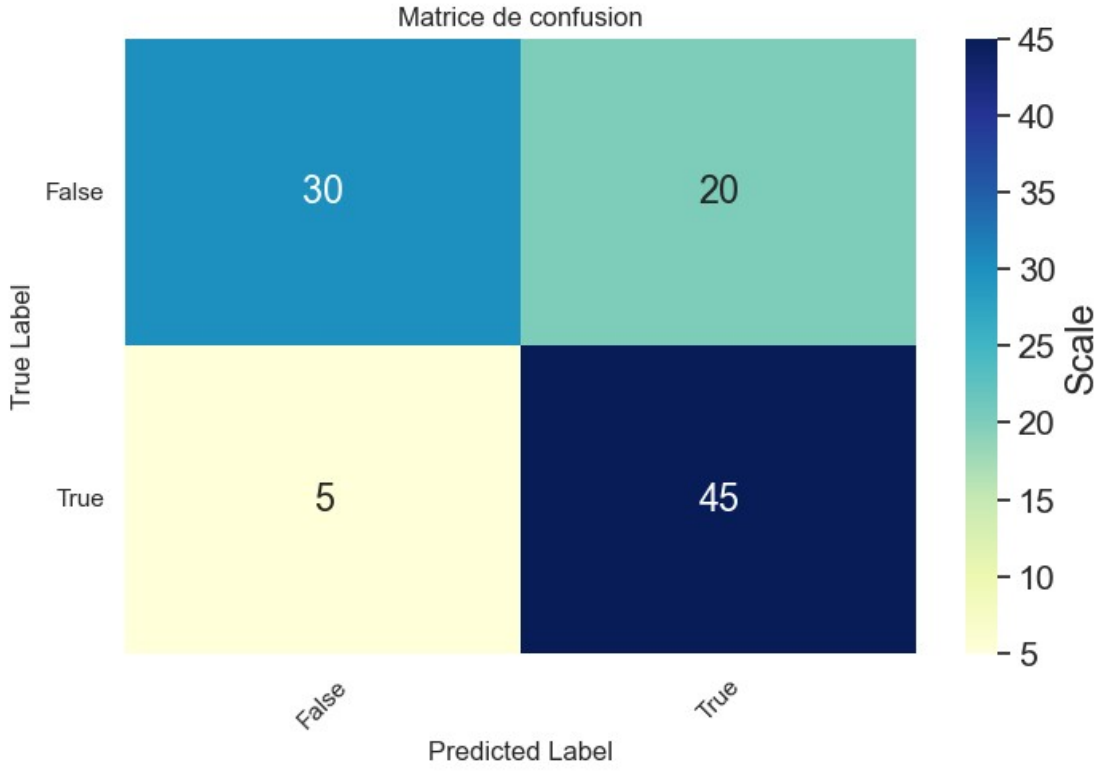
```
     tfidf__lowercase: False
     tfidf__stop_words: None

Les premiers résultats :
      cleaner__getlemmatisation  cleaner__removedigit  mnb__alpha  \
207                       False                  True         0.5
203                       False                  True         0.5
199                       False                  True         0.5
195                       False                  True         0.5
3                          True                  True         0.5

     mnb__fit_prior  mnb__force_alpha  tfidf__lowercase
tfidf__stop_words  \
207           False             False             False
None
203           False              True             False
None
199            True             False             False
None
195            True              True             False
None
3              True              True             False
None

     accuracy
207  0.749051
203  0.749051
199  0.749051
195  0.749051
3    0.748041
```

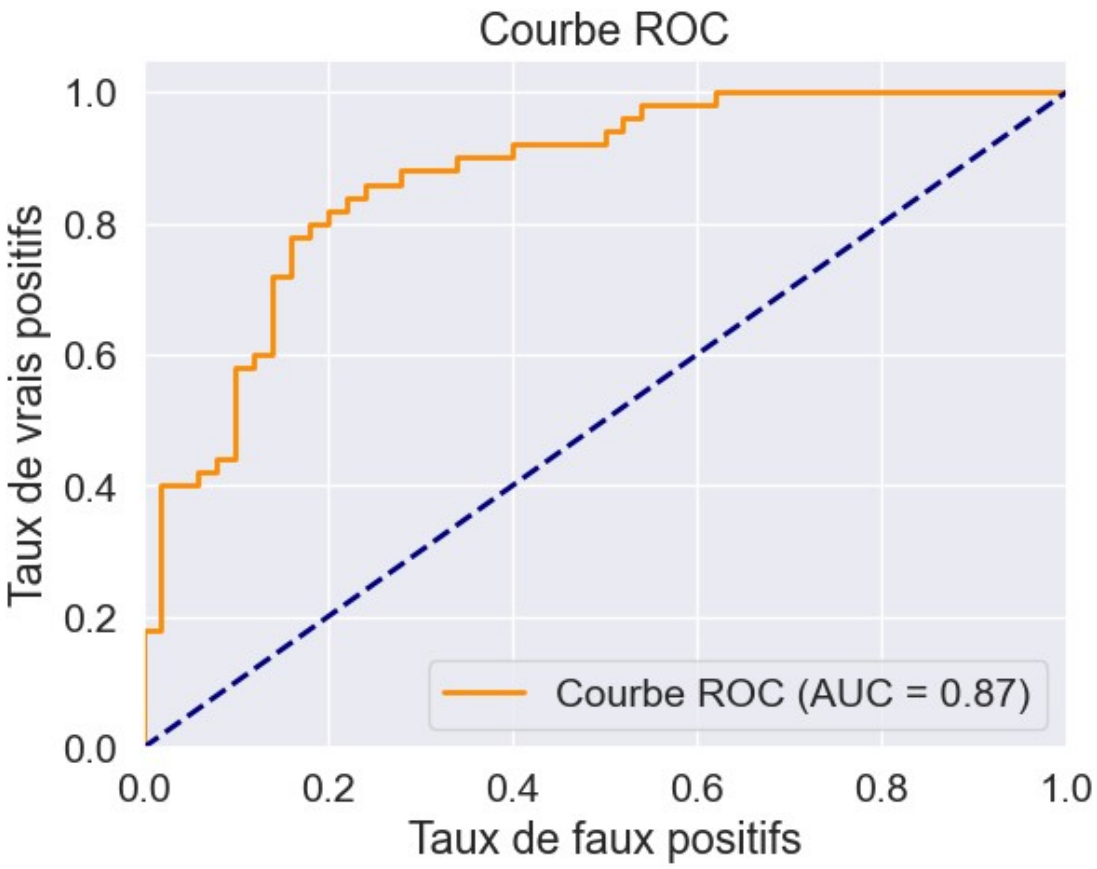
Enregistrement du modele MNB avec les meilleurs paramètres déterminés et affichage de la matrice de confusion et de la courbe ROC correspondante :

```
import pickle
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
# Création d'un jeu d'apprentissage et de test
trainsize=0.9 # 70% pour le jeu d'apprentissage, il reste 30% du jeu
de données pour
testsize= 0.1
seed=30
train_title,test_title,train_note,test_note=train_test_split(X_train,y
_train,
train_size=trainsize,random_state=seed,test_size=testsize,stratify=y_t
rain)

pipeline=Pipeline([
    ("cleaner", TextNormalizer(removedigit=True,
getlemmatisation=False)),
```

```
    ("tfidf", TfidfVectorizer(lowercase=False, stop_words=None)),
    ('mnb', MultinomialNB(alpha=0.5, fit_prior=True))
#force_alpha=True
])
pipeline.fit(train_title,train_note)
filename='./Modele/MNB_VF.pkl'
print("Sauvegarde du modèle dans ", filename)
pickle.dump(pipeline, open(filename, "wb"))



print ("Chargement du modèle \n")
# le chargement se fait via la fonction load
clf_loaded = pickle.load(open(filename, 'rb'))
# affichage du modèle sauvegardé
print (clf_loaded)


# test  avec les données qu'il a apprise c'est parfait woahhha c'est
beau
y_pred = clf_loaded.predict(test_title)
# autres mesures et matrice de confusion
MyshowAllScores(test_note,y_pred)


# Calcul de la courbe ROC
y_pred_proba = clf_loaded.predict_proba(test_title)[:,1]
fpr, tpr, thresholds = roc_curve(test_note, y_pred_proba)
roc_auc = auc(fpr, tpr)

# Tracé de la courbe ROC
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='Courbe ROC (AUC =
%0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Taux de faux positifs')
plt.ylabel('Taux de vrais positifs')
plt.title('Courbe ROC')
plt.legend(loc="lower right")
plt.show()

Sauvegarde du modèle dans  ./Modele/MNB_VF.pkl
Chargement du modèle

Pipeline(steps=[('cleaner', TextNormalizer(removedigit=True)),
                ('tfidf', TfidfVectorizer(lowercase=False)),
                ('mnb', MultinomialNB(alpha=0.5))])
```

```
Accuracy : 0.750
Classification Report
              precision    recall  f1-score   support

       False    0.85714   0.60000   0.70588        50
        True    0.69231   0.90000   0.78261        50

    accuracy                        0.75000       100
   macro avg    0.77473   0.75000   0.74425       100
weighted avg    0.77473   0.75000   0.74425       100
```






```
from Fonction.visualisation import *

myTSNE_2d_3d(X_train,y_train,True,True)
```


```
{"config":{"plotlyServerURL":"https://plot.ly"},"data":
[{"hovertemplate":"our
rating=True<br>0=%{x}<br>1=%{y}<extra></extra>","legendgroup":"True","
marker":
{"color":"#00cc96","symbol":"circle"},"mode":"markers","name":"True","
orientation":"v","showlegend":true,"type":"scatter","x":[-
13.49732780456543,37.65546417236328,-
21.000965118408203,5.053759574890137,-21.57198143005371,-
12.632574081420898,3.7592031955718994,-23.572479248046875,-
5.768886089324951,-23.563005447387695,-21.04134750366211,-
5.175833702087402,4.7573370933532715,-
9.68233871459961,25.57622528076172,-
35.97625732421875,23.747323989868164,9.951845169067383,14.885514259338
379,0.8298690319061279,-7.356023788452148,-0.7844404578208923,-
31.132009506225586,13.421232223510742,1.470735788345337,19.56011199951
172,19.342113494873047,-10.273326873779297,13.570314407348633,-
3.754551649093628,-38.85669708251953,8.955025672912598,-
21.390810012817383,4.4366278648376465,8.113669395446777,-
23.09417152404785,-8.556098937988281,3.499666690826416,-
21.574018478393555,-8.08711051940918,-
```

31.175704956054688,24.97936248779297,5.525749206542969,35.233837127685
55,4.970816135406494,3.4405603408813477,-
12.202720642089844,13.48295783996582,7.485034942626953,-
30.368553161621094,-5.9230523109436035,-22.981067657470703,-
26.709375381469727,-7.708553791046143,24.97647476196289,-
6.934449672698975,13.359627723693848,-27.075397491455078,-
23.509775161743164,-16.4791316986084,-46.607872009277344,-
6.929936408996582,-7.699748516082764,-9.365589141845703,-
21.352663040161133,5.574280738830566,-
9.8624906539917,37.18330764770508,-
26.115663528442383,12.935914993286133,-48.27564239501953,-
13.533129692077637,-12.122841835021973,-
27.495132446289062,30.690532684326172,-
12.684538841247559,7.485033988952637,-3.696798086166382,-
3.719604253768921,-12.957438468933105,-
7.425813674926758,16.463239669799805,-
7.525758266448975,7.646337032318115,-5.9230523109436035,-
17.904253005981445,-8.109659194946289,-10.452091217041016,-
15.1478853225708,-30.454626083374023,-11.0310697555542,-
27.29846954345703,-17.29240608215332,-22.798051834106445,-
3.815797805786133,0.4699862003326416,-
10.509685516357422,20.2286434173584,-34.94149398803711,-
15.141474723815918,27.75666618347168,-29.750225067138672,-
15.796660423278809,-10.37199592590332,-22.34637451171875,-
6.775447368621826,14.976058959960938,-32.563716888427734,-
22.636764526367188,-29.067153930664062,-16.655988693237305,-
17.633647918701172,12.723085403442383,-
27.132505416870117,9.718100547790527,19.13299560546875,20.118301391601
562,-4.097131729125977,-46.88346481323242,-
9.057703018188477,5.911511421203613,0.9942273497581482,-
27.295658111572266,9.295899391174316,-54.591346740722656,-
9.39322566986084,4.097621440887451,-19.158248901367188,-
5.021819591522217,24.27324867248535,-
3.4121809005737305,19.496097564697266,3.496894121170044,8.160269737243
652,25.169845581054688,4.842351913452148,-18.817106246948242,-
13.851713180541992,13.297530174255371,9.232161521911621,-
19.400880813598633,-15.902008056640625,-
0.7298086881637573,1.8162024021148682,8.678815841674805,-
6.519716739654541,-13.901976585388184,-
12.630331039428711,2.5400373935699463,-23.55551528930664,-
11.2274169921875,-9.298779487609863,-26.675277709960938,-
17.90017318725586,34.75748825073242,-4.283478736877441,-
17.41388702392578,3.866182565689087,-49.91183090209961,-
57.43940353393555,-58.95823287963867,-44.63936996459961,-
50.38475036621094,-4.620532989501953,-45.125213623046875,-
44.006507873535156,-50.480960845947266,-54.3802375793457,-
45.38954544067383,-45.130245208740234,-14.309466361999512,-
45.898223876953125,-49.14582443237305,-60.96739959716797,-
44.07666015625,-54.94377899169922,-49.842384338378906,-
38.65034484863281,-46.71169662475586,-52.18476867675781,-

54.850582122802734,-40.193870544433594,-40.8669548034668,-
52.60235595703125,-12.634383201599121,-53.55983352661133,-
40.607948303222656,-43.26973342895508,-57.72742462158203,-
54.98456954956055,-46.73826599121094,-50.1702995300293,-
49.2396354675293,-50.610328674316406,-51.42401885986328,-
56.0101203918457,-54.70973587036133,-26.675281524658203,-
17.19478988647461,-17.559537887573242,-
18.921669006347656,5.8904942125082016e-2,-53.86137008666992,-
43.249691009521484,9.769248008728027,-
26.62914276123047,25.323698043823242,-0.9626662135124207,-
17.194580078125,-21.87839126586914,-8.77829360961914,-
17.018898010253906,-
3.799820899963379,19.51763153076172,35.85707092285156,-
47.32033157348633,-53.86739730834961,-48.4058837890625,-
36.7534065246582,-44.29676055908203,-53.54307556152344,-
53.8861198425293,-51.74896240234375,-59.51741027832031,-
46.7255859375,-51.785396575927734,-0.7898244857788086,-
28.19394302368164,-
5.308983325958252,33.81643295288086,0.8641958832740784,-
4.327870845794678,-6.64127779006958,33.37486267089844,-
4.596056938171387,-
43.08892059326172,28.58709144592285,17.194578170776367,0.1748380810022
354,-40.637351989746094,-47.84236145019531,42.48688507080078,-
51.126556396484375,-
3.485776662826538,1.5236525535583496,13.520590782165527,-
29.565685272216797,-11.390740394592285,-13.82898998260498,-
6.0594964027404785,-
28.944223403930664,23.20334815979004,22.3176212310791,-
13.803367614746094,22.4243106842041,20.87899398803711,-
5.118067264556885,21.356138229370117,27.763307571411133,21.22135543823
2422,6.367389678955078,26.681482315063477,-6.598047256469727,-
21.396587371826172,-
7.940995216369629,8.45479965209961,15.417720794677734,9.73471736907959
,0.14693447947502136,11.581130981445312,-17.982131958007812,-
20.0915584564209,18.49303436279297,-27.948871612548828,-
9.243659019470215,18.933643341064453,4.9493937492370605,13.85296058654
7852,15.167376518249512,3.519667863845825,16.827178955078125,11.966011
047363281,2.117356777191162,1.301563024520874,20.130897521972656,17.18
1747436523438,14.144882202148438,8.91158390045166,11.938304901123047,0
.3518356680870056,-1.2107737064361572,-
6.121053218841553,8.05039119720459,-
52.267459869384766,9.609241485595703,5.037355422973633,-
23.72144317626953,18.178258895874023,5.906611919403076,-
5.71207857131958,18.357418060302734,-
21.216222763061523,20.474048614501953,25.127321243286133,-
18.80628776550293,-0.916593611240387,-
11.993144989013672,28.567480087280273,-
8.04434871673584,3.831773519515991,39.75318908691406,30.70346832275390
6,12.84427261352539,8.756397247314453,14.80611515045166,11.65678310394
2871,34.01011276245117,25.37737464904785,-1.0222628116607666,-

7.095990180969238,-10.297048568725586,19.241830825805664,-
45.64555740356445,-53.984737396240234,-47.49996566772461,-
59.80451583862305,-59.800628662109375,-0.7571900486946106,-
27.542020797729492,-4.096981525421143,-
3.5167295932769775,17.64035987854004,0.28168487548828125,-
40.6912956237793,-47.84427261352539,-14.924782752990723,-
50.3726692199707,-
13.638020515441895,18.81598472595215,26.06008529663086,21.754686355590
82,22.518247604370117,-1.427655816078186,-
12.723645210266113,18.277585983276367,23.257360458374023,29.8444099426
26953,7.0694499015808105,24.79953956604004,13.227231979370117,17.45565
9866333008,2.360694169998169,21.511356353759766,-
17.462053298950195,18.346765518188477,12.356151580810547,-
11.505593299865723,2.1998724937438965,9.093647003173828,6.159515380859
375,5.0804972648620605,-
6.519046306610107,18.292783737182617,9.850359916687012,7.1966576576232
91,-9.475214958190918,-
21.545724868774414,34.49148178100586,13.098124504089355,19.01467704772
9492,6.044259548187256,25.22153091430664,-0.4386969208717346,-
10.275794982910156,-4.759128093719482,17.58895492553711,-
11.785367012023926,25.422260284423828,20.941102981567383,-
12.779563903808594,22.90985107421875,32.69638442993164,8.4408884048461
91,14.619180679321289,-
33.84299087524414,18.078996658325195,11.902835845947266,28.12018013000
4883,2.132559061050415,6.951803684234619,0.993321418762207,4.975409507
751465,-
57.288780212402344,27.282737731933594,0.7661352753639221,20.3140277862
54883,12.539610862731934,-
11.3467378616333,24.238008499145508,22.242828369140625,17.377431869506
836,40.19562530517578,22.810165405273438,-
61.45222091674805,10.878864288330078,10.245171546936035,11.85569858551
0254,18.8658447265625,4.221321105957031,4.180819988250732,3.7050623893
737793,3.8318774700164795,-7.940820693969727,-
28.944047927856445,22.242691040039062,-7.699317455291748,-
49.24003601074219,9.73471736907959,11.901406288146973,-
59.51533126831055,-17.95682716369629,-18.817106246948242,-
9.243368148803711,-52.601966857910156,37.18330764770508,-
8.086729049682617,20.313928604125977,35.856441497802734,23.74732589721
6797,-
17.19403648376465,25.576412200927734,39.75319290161133,34.491569519042
97,-49.842384338378906,-4.759167194366455,-29.783950805664062,-
46.60868453979492,8.441643714904785,33.374847412109375,-
29.21331214904785,28.570026397705078,5.395775318145752,-
11.784666061401367,1.8162542581558228,10.45128345489502,0.864015817642
2119,8.051005363464355,-5.118067741394043,-
27.075397491455078,19.560110092163086,9.715217590332031,-
18.104848861694336,-12.20301628112793,-
20.09180450439453,5.053759574890137,-56.010963439941406,-
59.80451965332031,-45.645633697509766,-
12.684538841247559,13.098123550415039,-

44.07666015625,42.48677062988281,-
8.554352760314941,10.880597114562988,-
12.634202003479004,14.885767936706543,-21.390663146972656,-
46.88346481323242,-13.901759147644043,40.195865631103516,-
7.596484184265137,-44.63959884643555,14.145095825195312,-
26.59881019592285,24.237844467163086,16.463367462158203,21.356462478637695,4.097801685333252,-17.14119529724121,-1.4283382892608643,-
45.125579833984375,-27.505769729614258,30.69053077697754,-
45.389549255371094,-
18.368144989013672,19.133220672607422,23.258037567138672,23.203596115112305,15.167376518249512,-31.1757755279541,-
40.193931579589844,13.359627723693848,18.933380126953125,-
21.57206916809082,19.51763153076172,16.826908111572266,-
61.4528694152832,5.907098770141602,6.044267654418945,32.69639205932617],"xaxis":"x","y":[-7.548803329467773,-
33.08891677856445,38.523868560791016,0.6774696707725525,17.249313354492188,28.241945266723633,39.610809326171875,-
13.46804428100586,12.233420372009277,-
11.967257499694824,25.456594467163086,37.18924331665039,-
3.5274295806884766,28.83824348449707,-
51.462158203125,4.754337310791016,28.663902282714844,-
12.24035358428955,-17.90875244140625,-21.193382263183594,-
23.636812210083008,-9.138484954833984,-17.692178726196289,-
2.413696527481079,60.6119499206543,-
7.662350177764893,24.794809341430664,16.919145584106445,-
2.203754425048828,-1.1628975868225098,14.143095970153809,-
11.240787506103516,-8.29916763305664,13.286306381225586,-
11.942647933959961,-3.0251169204711914,-6.97458028793335,-
13.094873428344727,27.9488525390625,24.457895278930664,21.696666717529297,-9.682164192199707,-11.5315523147583,-
13.012308120727539,6.6262688636779785,-6.465312480926514,-
7.273325443267822,25.05400276184082,-7.503669738769531,-
16.364126205444336,3.1418023109436035,4.579113483428955,-
40.359107971191406,-2.6226131916046143,43.31229019165039,-
12.170516014099121,9.757097244262695,-
7.885310173034668,21.426469802856445,-14.056251525878906,-
11.764065742492676,-21.533620834350586,-
17.354537963867188,1.3159170150756836,2.9195637702941895,-
17.099258422851562,-25.29965591430664,-6.161646842956543,-
40.14572525024414,-
17.584781646728516,7.023688793182373,54.00027084350586,28.788206100463867,17.42215919494629,2.9397764205932617,2.9915621280670166,-
7.503669738769531,8.952502250671387,-49.33814239501953,-
11.297673225402832,-22.717208862304688,-24.69363784790039,-
10.911431312561035,-
12.876131057739258,3.1418023109436035,6.9010210037231445,-
11.538684844970703,30.733213424682617,6.331033229827881,-
19.03584098815918,-14.74782943725586,9.383867263793945,-
3.8890771865844727,16.48944664001465,-
14.144521713256836,61.23795700073242,53.393436431884766,8.274025917053

223,4.513555526733398,22.095096588134766,5.319763660430908,-
15.9494047164917,27.524248123168945,-
21.520099639892578,3.527259349822998,-11.269088745117188,-
10.172467231750488,-
13.191439628601074,9.105207443237305,15.94874382019043,-
2.828784704208374,6.937492847442627,-
0.8360748291015625,17.6127986907959,6.536721229553223,-
11.449152946472168,-
17.37115478515625,30.019920349121094,17.38463592529297,27.799970626831
055,-10.149300575256348,32.464664459228516,9.38602066040039,-
10.2717924118042,22.149185180664062,3.9277536869049072,32.524238586425
78,38.94124984741211,20.851110458374023,-8.096138954162598,-
9.465872764587402,-20.02573585510254,-
13.093096733093262,1.1600934267044067,21.051828384399414,6.01554870605
46875,21.612640380859375,-16.403940200805664,-26.568893432617188,-
5.170233249664307,38.753509521484375,-
19.084566116333008,0.44448551535606384,59.41584777832031,-
6.199769973754883,-
3.497248411178589,13.775447845458984,13.605955123901367,-
6.415098667144775,-12.434469223022461,-
1.7029505968093872,0.5527940988540649,-25.51587677001953,-
30.451236724853516,11.9994478225708,0.43882057070732117,-
36.73457717895508,-
17.297222137451172,18.55976104736328,20.396745681762695,20.48944473266
6016,11.240670204162598,10.540372848510742,-
13.742079734802246,6.317347526550293,5.257743835449219,15.475869178771
973,18.91583251953125,14.736721992492676,16.749494552612305,7.06465435
0280762,13.177847862243652,15.752278327941895,9.67597484588623,19.3216
97235107422,19.81378746032715,14.243765830993652,15.867315292358398,15
.18085765838623,14.423023223876953,20.50261116027832,19.56015777587890
6,14.818355560302734,21.09267234802246,7.637979030609131,19.2266025543
2129,15.262128829956055,5.389639854431152,21.879974365234375,14.542345
04699707,21.249496459960938,10.34911060333252,22.574790954589844,17.95
1244354248047,18.046539306640625,17.82415771484375,15.978880882263184,
-25.515886306762695,-29.61859893798828,-37.59000015258789,-
36.48991012573242,23.243755340576172,25.30998420715332,9.098876953125,
33.33466720581055,42.92552185058594,-
0.2897699475288391,7.040021896362305,20.123231887817383,10.48399353027
3438,16.52898406982422,-
8.536974906921387,13.642477989196777,1.5480554103851318,-
42.800758361816406,20.61129379272461,25.31676483154297,8.1775646209716
8,4.888214111328125,23.280094146728516,16.811643600463867,10.971627235
412598,4.817071914672852,9.713616371154785,11.081597328186035,14.04823
4939575195,-16.290481567382812,3.6900501251220703,-
37.321868896484375,9.459155082702637,-2.81325101852417,-
23.69202423095703,-23.7829532623291,11.086910247802734,-
5.314286708831787,3.9485018253326416,-39.13470458984375,-
32.88227462768555,-
37.61408233642578,1.7767798900604248,27.474374771118164,-
10.477511405944824,-1.5429855585098267,-8.006644248962402,-

1.271297574043274,-46.158226013183594,-32.02268600463867,-
8.361162185668945,25.426265716552734,-7.43296480178833,-
30.869871139526367,-31.173301696777344,-46.14834976196289,-
21.35991668701172,-44.374542236328125,-32.765113830566406,-
27.060293197631836,-47.464988708496094,-37.04977035522461,-
23.8835391998291,-42.70844650268555,-
43.263084411621094,7.8618669509887695,31.43161964416504,-
36.73809051513672,-27.644739151000977,-
48.84121322631836,13.569978713989258,21.06744956970215,-
21.185800552368164,27.890676498413086,29.537845611572266,12.732013702392578,-1.9927589893341064,-26.386781692504883,-38.696048736572266,-29.711523056030273,4.71612024307251,-40.97171401977539,-27.522220611572266,-46.10257339477539,-40.17961883544922,-32.000953674316406,-44.945213317871094,-45.13787078857422,-28.687896728515625,-43.406280517578125,-34.315425872802734,-42.423099517822266,24.296035766601562,-39.352054595947266,52.85283279418945,55.41548538208008,4.691343307495117,37.562660217285156,22.98925018310547,-1.996599793434143,38.86507034301758,35.788063049316406,-44.431522369384766,30.587162017822266,30.10407257080078,-37.9559211730957,-38.86178207397461,-22.749557495117188,-36.708038330078125,50.76628112792969,-17.926427841186523,20.004060745239258,26.525943756103516,5.814277172088623,-2.2406506538391113,19.292478561401367,25.395231246948242,-16.377477645874023,-32.3336181640625,38.96856689453125,9.716848284006119e-2,7.683272838592529,16.119823455810547,19.152801513671875,2.684513568878174,22.710786819458008,9.490734100341797,10.509503364562988,14.060989379882812,11.401467323303223,-16.29425048828125,3.0894107818603516,-37.29315185546875,-4.139110088348389,-32.033485412597656,-37.97230529785156,1.750783085823059,27.571481704711914,53.82294845581055,-0.35768216848373413,-2.694735527038574,-34.33164596557617,-29.59703254699707,-34.77818298339844,-36.59111022949219,-26.99034881591797,-24.036020278930664,-35.41474533081055,-35.318416595458984,-38.801536560058594,-41.383934020996094,48.153076171875,-46.382205963134766,-43.22426986694336,1.752642273902893,-21.056434631347656,29.432884216308594,13.226130485534668,-28.720151901245117,-29.29279899597168,-25.697952270507812,-33.5035285949707,-29.419795989990234,23.044178009033203,-45.59630584716797,29.202110290527344,-44.04259490966797,34.368980407714844,20.21430778503418,-1.8687711954116821,37.87776184082031,18.150951385498047,-22.232206344604492,-57.23719024658203,-1.255075216293335,7.379551410675049,18.38212013244629,-6.576974868774414,-41.639671325683594,54.22743225097656,-30.081867218017578,-35.382022857666016,-21.849014282226562,-44.1434211730957,-33.55681228637695,-42.495548248291016,-44.85289764404297,13.787904739379883,14.171652793884277,-27.605667114257812,-42.52009582519531,-31.817800521850586,-

28.869230270385742,24.170143127441406,38.10965347290039,21.85683822631836,-29.71505355834961,-26.009849548339844,-41.582252502441406,-29.640201568603516,-29.116859436035156,10.301835060119629,-57.836387634277344,-41.64498519897461,23.578439712524414,-25.37384796142578,8.13248348236084,-18.388080596923828,-43.539878845214844,2.8196520805358887,18.875322341918945,-50.10724639892578,-49.31566619873047,-50.29607009887695,26.525236129760742,-36.73782730102539,-30.86988067626953,-57.835018157958984,-17.354167938232422,22.574724197387695,13.569978713989258,-27.605722427368164,9.716743469238281,-29.6254825592041,21.612640380859375,-26.38611602783203,21.092239379882812,-6.161649227142334,24.458045959472656,-41.58216857910156,-42.801029205322266,28.663902282714844,20.123151779174805,-51.46343994140625,5.814276218414307,37.87761688232422,14.243764877319336,-6.576962947845459,-16.995908737182617,-11.764001846313477,-42.494659423828125,11.086970329284668,-16.561769485473633,-17.928085327148438,6.208817958831787,54.22706985473633,59.41557693481445,-44.13330841064453,-2.8133246898651123,55.417362213134766,-27.06029510498047,-7.885317325592041,-7.662351131439209,6.539422988891602,-36.22917556762695,-7.274780750274658,29.53982162475586,0.6774696707725525,17.824380874633179,14.060989379882812,22.71157455444336,2.9915621280670166,18.150951385498047,19.321697235107422,-10.47750473022461,-6.973536491394043,-18.387357711791992,7.639132499694824,-17.908946990966797,-8.299102783203125,17.38463592529297,13.774303436279297,23.57832908630371,-12.32042121887207,11.240331649780273,-43.40678024291992,-39.73468780517578,10.301878929138184,-24.693939208984375,-47.46497344970703,32.52431106567383,-30.412492752075195,-26.98907470703125,6.3167219161987305,3.1423416137695312,2.939777135848999,14.736725807189941,-37.88399124145508,-11.449004173278809,-35.31959533691406,-31.173206329345703,-40.971710205078125,21.696531295776367,19.5603084564209,9.757097244262695,-38.69544219970703,17.249048233032227,1.5480554103851318,-46.10110092163086,8.132218360900879,35.78819274902344,-57.237552642822266,-33.55680847167969],"yaxis":"y"},
{"hovertemplate":"our rating=False
0=%{x}
1=%{y}<extra></extra>","legendgroup":"False","marker":
{"color":"#ab63fa","symbol":"circle"},"mode":"markers","name":"False",
"orientation":"v","showlegend":true,"type":"scatter","x":
[4.082810878753662,-4.75270938873291,32.55571746826172,-8.922905921936035,-2.0855259895324707,-0.7470874190330505,-1.6020687818527222,12.652103424072266,14.99388599395752,-57.686241149902344,12.786835670471191,-12.565816879272461,-17.333242416381836,24.327604293823242,2.2365894317626953,11.448771476745605,-13.758342742919922,-5.184327602386475,12.151864051818848,41.80461120605469,27.235403060913086,-

56.63517379760742,2.656942367553711,7.919570446014404,14.808503150939941,-3.478942632675171,-4.261532783508301,25.362367630004883,-5.232466697692871,11.948119163513184,22.8015193939209,18.599140167236328,21.990009307861328,8.908968925476074,18.819700241088867,8.671951293945312,13.714692115783691,15.373007774353027,21.47301483154297,36.690521240234375,3.003532886505127,28.00777244567871,45.383033752441406,0.14879232645034790,-5.336460590362549,1.9301422834396362,5.836128234863281,11.685982704162598,-24.75501251220703,21.467418670654297,-19.287429809570312,-2.9227206707000732,11.425775527954102,-12.112975120544434,4.083739757537842,38.88999557495117,9.741050720214844,-32.18291091918945,17.64573860168457,10.322528839111328,-18.356489181518555,-43.34465408325195,23.390209197998047,5.0568718910217285,-24.82634925842285,21.786766052246094,7.721749305725098,-11.99176025390625,21.731618881225586,17.869890213012695,2.4752540588378906,-15.821839332580566,22.16507339477539,22.845317840576172,-15.665884971618652,5.700368881225586,7.325790882110596,-12.64844036102295,6.097106456756592,-12.166223526000977,2.022610664367676,2.2875783443450928,-17.54670524597168,4.183107852935791,38.41957092285156,-33.50435256958008,14.88479232788086,-7.89588737487793,17.47355842590332,-12.462239265441895,37.1829948425293,10.809066772460938,20.788846969604492,-25.423185348510742,-31.93840980529785,-38.712459564208984,-13.035383224487305,18.312644958496094,-7.812148094177246,-8.3479585647583,-9.465653419494629,29.49526023864746,5.801630020141602,38.768936157226562,-8.646326065063477,21.73164939880371,-38.71246337890625,-1.1518477201461792,-17.031177520751953,-22.39336585998535,6.738389492034912,20.681915283203125,31.574844360351562,-32.48930740356445,-15.524714469909668,-11.677263259887695,14.869378089904785,-14.159625053405762,-8.536550521850586,22.670164108276367,18.289398193359375,11.832763671875,-4.4270734786987305,30.271560668945312,15.447497367858887,-7.945519924163818,-15.558145523071289,8.908968925476074,-2.7553279399871826,27.387712478637695,31.070613861083984,-8.685591697692871,-3.0051848888397217,-35.02896499633789,4.534326553344727,2.8208703994750977,6.753520488739014,-20.080976486206055,28.55294418334961,5.952474594116211,-15.536396980285645,10.381671905517578,22.14606475830078,-2.4808526039123535,29.195404052734375,19.512908935546875,24.636709213256836,2.1534066200256348,20.110387802124023,21.73737335205078,-43.676822662353516,9.041610717773438,-1.9153187274932861,29.831592559814453,4.163148880004883,1.6773617267608643,-12.655058860778809,2.7469427585601807,38.76889419555664,16.069395065307617,20.611474990844727,2.5345799922943115,-15.501313209533691,-1.5409871339797974,5.0522010773420334e-2,-27.53917121887207,31.666330337524414,0.48522597551345825,-

16.145536422729492,-16.833019256591797,1.4751588106155396,-
8.9787058532238e-
2,23.28968048095703,16.3142032623291,2.659588098526001,-
8.222908973693848,30.878864288330078,7.630901336669922,16.245870590209
96,-18.01312828063965,-11.797365188598633,15.422898292541504,-
15.347288131713867,8.596643447875977,2.1692826747894287,23.74468040466
3086,-43.11259841918945,13.614148139953613,7.72163200378418,-
21.97176742553711,9.933396339416504,10.508752822875977,45.453037261962
89,0.9693339467048645,4.593875408172607,6.534378528594971,39.312824249
26758,-
15.65881633758545,32.05296325683594,14.524995803833008,31.415159225463
867,8.670748710632324,20.825634002685547,13.350296020507812,8.59685134
8876953,12.886828422546387,-24.033933639526367,-28.293506622314453,-
25.441574096679688,-33.110260009765625,-
11.289008140563965,30.856966018676758,-
4.374871253967285,10.219346046447754,27.15615463256836,-
19.036863327026367,20.177722930908203,0.54838627576828,27.508787155151
367,5.343771934509277,26.383995056152344,28.554912567138672,-
0.9777052402496338,-
51.15950012207031,31.324243545532227,17.739871978759766,15.74660205841
0645,13.479501724243164,-
41.17018508911133,15.671561241149902,11.371956825256348,41.30046844482
422,10.815128326416016,20.615304946899414,-
28.804119110107422,3.441237688064575,0.6952899098396301,-
1.0689849853515625,0.7881516814231873,-12.523255348205566,-
26.144855499267578,-6.114182472229004,1.7029143571853638,-
45.65092849731445,5.379489898681641,-15.948461532592773,-
4.373720169067383,-23.832935333251953,-
23.42015838623047,5.723769187927246,23.024106979370117,-
18.8206787109375,7.403237342834473,-11.338323593139648,-
32.51227951049805,-
3.661097764968872,27.702707290649414,16.06863021850586,-
10.916000366210938,-3.408172845840454,34.24458694458008,-
15.160482406616211,30.882225036621094,-
9.560575485229492,31.882253646850586,20.551183700561523,9.908603668212
89,31.910015106201172,-10.655271530151367,-
5.606505870819092,30.538782119750977,7.939539909362793,15.740724563598
633,21.168682098388672,17.473892211914062,-18.927623748779297,-
12.090902328491211,-9.862757682800293,-9.551023691892624e-
2,16.252981185913086,-7.176453590393066,1.475158929824829,-
4.650761604309082,9.79626178741455,-
27.4984073638916,12.551713943481445,7.993078231811523,-
24.840253829956055,-
50.0485725402832,1.6380891799926758,30.847389221191406,41.321800231933
594,-3.718021869659424,-41.251426696777344,-
50.909156799316406,11.523916244506836,16.426166534423828,29.5207138061
52344,-2.133622407913208,-1.5085837841033936,-17.78567886352539,-
16.24725914001465,23.67351531982422,-
3.277291774749756,25.651140213012695,-
13.424060821533203,16.851285934448242,-41.62567901611328,-

35.52891159057617,-
17.14659881591797,35.23508071899414,32.54872512817383,2.90661406517028
8,6.616513252258301,8.921417236328125,-5.818427085876465,-
17.049936294555664,29.737394332885742,-5.236413478851318,-
9.226105690002441,13.628449440002441,19.195554733276367,13.54078292846
6797,2.9140071868896484,24.637292861938477,-
2.146489381790161,29.204059600830078,17.859317779541016,40.46944808959
961,0.2791052460670471,7.9899678230285645,-11.884629249572754,-
9.144113540649414,7.9931511878967285,23.643856048583984,-
38.35374069213867,-
11.741434097290039,16.870365142822266,41.85969543457031,25.53635406494
1406,12.912775993347168,44.906883239746094,-36.28288650512695,-
6.997118949890137,24.39333152770996,-15.160589218139648,-
23.4213924407959,-
12.999302864074707,27.961151123046875,31.071941375732422,-
3.2837369441986084,16.043474197387695,-1.315934419631958,-
2.449695110321045,-22.85835838317871,-
6.569479465484619,12.646530151367188,-
19.131120681762695,15.455907821655273,10.371918678283691,-
32.09613037109375,0.20895041525363922,0.13882094621658325,3.5822505950
927734,25.354957580566406,-
10.041036605834961,36.816158294677734,31.415361404418945,13.2040586471
55762,-4.087512969970703,9.611615180969238,-5.092852592468262,-
19.647151947021484,16.42384147644043,-26.46352767944336,-
28.614057540893555,22.16506576538086,26.384368896484375,38.40257263183
594,39.466583251953125,-
7.682057857513428,7.5176496505737305,31.97742462158203,23.483114242553
71,10.124105453491211,3.918644905090332,7.337258338928223,-
19.417232513427734,27.799001693725586,28.72583770751953,12.65044879913
33,-16.744848251342773,-53.907447814941406,-18.258758544921875,-
1.440812587738037,22.66547393798828,14.761962890625,5.787684440612793,
41.36003875732422,25.970252990722656,-
11.717674255371094,22.100317001342773,-
10.865594863891602,25.401491165161133,-17.597888946533203,-
28.705114364624023,21.14069175720215,31.140518188476562,22.80143547058
1055,-17.5908260345459,-28.470788955688477,-
13.694731712341309,31.268150329589844,23.590496063232422,-
8.661355972290039,0.12932567298412323,-28.132780075073242,-
17.923248291015625,13.479501724243164,6.79218053817749,12.786836624145
508,-0.23151670396327972,22.4790096282959,32.170013427734375,-
3.8610219955444336,18.718862533569336,40.72908020019531,20.57401275634
7656,-16.283954620361328,-36.277854919433594,-6.352786540985107,-
26.114038467407227,-33.4815788269043,9.034159660339355,-
3.5712897777557373,-
14.379493713378906,13.713708877563477,1.3774086236953735,-
41.62591552734375,1.2274484634399414,10.87936019897461,-
33.48408508300781,-4.23406982421875,8.368616104125977,-
9.540972709655762,-6.501297473907471,-
9.303312301635742,4.3383097648620605,25.128705978393555,37.18159866333
008,24.696792602539062,16.63553810119629,-

20.722562789916992,16.656200408935547,22.019018173217773,-47.50214385986328,-0.6521096229553223,8.081841468811035,-16.65036964416504,3.363356590270996,-12.32203483581543,2.3211331367492676,30.255001068115234,-3.565359354019165,10.258264541625977,3.664061799645424e-2,-2.6780219078063965,20.519145965576172,20.61262321472168,20.7040195465087 9,0.9980320334434509,18.739055633544922,-23.719711303710938,23.479171752929688,14.171542167663574,-14.336400032043457,-16.2547664642334,13.83600902557373,1.740287184715271,14.47214698791503 9,4.513634204864502,10.38492202758789,29.61670684814453,-14.710530281066895,20.70016860961914,-1.1703715324401855,13.129721641540527,21.133283615112305,21.926347732543945,20.081876754760742,-43.12291717529297,16.38965606689453,-3.362499952316284,13.141793251037598,44.905174255371094,-16.35648536682129,-4.394402027130127,12.67843246459961,25.302242279052734,16.158679962158203],"xaxis":"x","y":[17.96017837524414,34.93984603881836,-39.59892272949219,-3.2972865104675293,15.501816749572754,-45.715293884277344,28.896345138549805,42.102821350097656,-31.064416885375977,18.566307067871094,34.20499038696289,-1.8007943630218506,12.610255241394043,27.360933303833008,37.044071197509766,-36.96246337890625,57.37390899658203,10.043766975402832,-25.415151596069336,-19.383161544799805,33.364418029785156,19.72867202758789,6.951560020446777,12.11085319519043,10.308835983276367,-5.585514068603516,22.023597717285156,18.836257934570312,-3.9167962074279785,-35.309654235839844,-37.32197189331055,18.22239875793457,-29.704299926757812,43.939170837402344,28.22834587097168,39.236457824707 03,-21.779354095458984,-2.9614930152893066,-3.9293336868286133,-36.83189010620117,48.400028228759766,-30.73624610900879,10.875324249267578,42.89590835571289,40.764900207519 53,11.000060081481934,-11.23802661895752,-51.543067932128906,26.756790161132812,39.12525939941406,32.28239059448242,26.130054473876953,-38.422611236572266,17.614994049072266,17.960237503051758,-36.86361312866211,39.69031524658203,14.551164627075195,-36.48358154296875,50.69679641723633,-11.131375312805176,27.58999252319336,-24.4437255859375,-17.000469207763672,30.048480987548828,45.49082565307617,18.385841369628906,36.81428527832031,42.02455139160156,34.68988800048828,-51.27862548828125,32.840538024902344,-39.56353759765625,-17.664167404174805,-2.962026357650757,18.92339324951172,8.39847469329834,32.10248947143555,29.181316375732422,56.45646667480469,-43.858009338378906,17.028846740722656,-15.102421760559082,28.93564796447754,-25.186859130859375,-2.212951421737671,-38.52363204956055,12.648784637451172,-52.97527313232422,20.490304946899414,0.5981860756874084,26.24361038208

0078,-48.92891311645508,29.072093963623047,-18.8616943359375,-
9.404158592224121,20.86141014099121,3.2699992656707764,-
30.934467315673828,16.583309173583984,35.046024322509766,-
26.361345291137695,42.13005065917969,-
34.396549224853516,18.180042266845703,42.02455139160156,-
9.40415096282959,11.814615249633789,-11.060615539550781,-
18.24709129333496,-46.232730865478516,-44.61868667602539,-
20.475528717041016,14.69547176361084,-46.65713882446289,-
2.21498703956604,-28.6901912689209,29.431928634643555,-
1.337997555732727,37.224273681640625,17.102237701416016,10.284747123718262,-2.2494869232177734,-36.700897216796875,-
9.605195045471191,5.2219929695129395,45.07608413696289,43.93927764892578,-32.348716735839844,-46.737884521484375,-
49.88774490356445,44.94567108154297,25.892955780029297,-
1.886618733406067,-23.12152671813965,-17.268844604492188,-
25.730770111083984,33.343162536621094,32.12892532348633,39.67694854736328,-46.64085388183594,29.2918643951416,-19.34902572631836,-
32.35453796386719,-
14.889104843139648,37.653785705566406,47.8269157409668,-
24.440183639526367,8.425188064575195,-
33.50330352783203,12.630175590515137,21.959102630615234,-
45.34918975830078,-9.849931716918945,-25.271133422851562,-
38.75462341308594,-13.99271297454834,-25.002788543701172,-
34.396705627441406,-
36.39857864379883,32.47547149658203,26.85191535949707,-
22.61809730529785,34.81314468383789,15.093873977661133,49.001365661621094,-44.09185028076172,1.1982786655426025,-
24.723094940185547,21.058929443359375,12.638187408447266,-
12.350861549377441,-
26.5570125579834,7.729712009429932,47.10956954956055,32.33159255981445,30.740999221801758,28.860876083374023,35.83146286010742,34.15434646606445,39.710540771484375,22.508224487304688,45.963836669921875,27.359472274780273,46.40227127075195,19.43173599243164,15.182764053344727,29.364368438720703,18.385986328125,-2.051459550857544,-
17.011056900024414,-
9.118874549865723,11.242060661315918,30.88344383239746,10.360767364501953,16.91742706298828,-31.448400497436523,31.493642807006836,-
21.128507614135742,-34.62434387207031,-
3.210829734802246,39.235435485839844,-19.14025115966797,-
40.478939056396484,27.359375,43.16194152832031,34.753578186035156,50.343006134033203,26.711950302124023,5.126000881195068,-21.35512924194336,-
26.79048728942871,47.89125442504883,-
46.93648147583008,19.50650405883789,-
37.347957611083984,39.956329345703125,-24.12063217163086,-
46.86568069458008,-
34.115901947021484,37.64193344116211,32.12845230102539,20.38508415222168,-1.5961105823516846,19.41101837158203,-
15.760172843933105,37.51763153076172,-
11.573134422302246,16.086904525756836,-0.7131037712097168,-
24.83070945739746,-35.301361083984375,29.627588272094727,-

26.29079818725586,35.2877311706543,30.109268188476562,-
5.814266204833984,2.6420764923095703,4.052698135375977,-
19.870729446411133,27.672054290771484,-0.6438470482826233,-
5.9303507804870605,9.6302490234375,41.18695831298828,28.84300994873047
,47.890933990478516,-22.283729553222656,33.59844207763672,-
37.790122985839844,36.504058837890625,2.6329264640808105,1.09241235256
19507,39.18723678588867,13.901742935180664,10.946935653686523,5.370494
842529297,-36.39411926269531,-4.7511773109436035,-46.696617126464844,-
34.311622619628906,-12.68130111694336,-
40.81080627441406,54.4574089050293,-
19.93463897705078,3.114288568496704,5.111254692077637,19.3029098510742
2,39.25642013549805,41.08454132080078,-49.44233322143555,-
45.15171813964844,37.51335144042969,36.13432693481445,-
52.976318359375,31.329174041748047,-
4.3480224609375,7.956037998199463,-
12.270350456237793,7.781454563140869,-
4.17756986618042,12.638189315795898,36.47407150268555,-
36.04176330566406,10.810837745666504,29.52959632873535,-
22.272592544555664,30.056604385375977,4.086263656616211,28.92461395263
672,-26.796300888061523,32.0130500793457,-
46.536048889160156,10.792972564697266,6.638181209564209,-
37.020545959472656,47.08625411987305,-41.875850677490234,-
7.538939476013184,32.281227111816406,32.361698150634766,2.054546117782
5928,-41.75970458984375,35.524539947509766,-13.569899559020996,-
25.52904510498047,-38.63243103027344,-
20.11017608642578,30.2493839263916,10.709802627563477,-
13.026546478271484,-26.077829360961914,41.21839141845703,-
17.43625831604004,-
12.568398475646973,10.434287071228027,10.323535919189453,30.4598922729
4922,29.442909240722656,30.94186019897461,38.479156494140625,-
29.649330139160156,-38.47074508666992,-
8.1762113571167,18.436676025390625,44.794288635253906,-
43.414005279541016,34.693878173828125,-
18.9869384765625,39.31957244873047,30.73853874206543,10.14054012298584
,8.805091857910156,-22.272319793701172,-
41.80416488647461,36.04591369628906,30.64117431640625,0.21152719855308
533,-31.281997680664062,10.719627380371094,-
36.5127067565918,0.1600993573665619,24.153339385986328,8.7769613265991
21,-7.386509418487549,-
12.676909446716309,33.58473587036133,33.37546920776367,-
34.55582046508789,-
40.53688430786133,35.515106201171875,41.22168731689453,44.021583557128
906,-48.056339263916016,11.96704387664795,25.649635314941406,-
3.5519368648529053,1.0297402143478394,-31.823328018188477,-
50.43996047973633,-
2.266263484954834,0.9891204833984375,42.884761810302734,48.84190750122
07,-39.08878707885742,34.30604934692383,-43.633392333984375,-
3.210893392562866,-23.387638092041016,-32.340904235839844,-
38.37003707885742,-45.43046569824219,-
30.246065139770508,47.086360931396484,12.476410865783691,35.2690391540

52734,-39.56327438354492,37.64191436767578,-25.2115478515625,-
32.44633483886719,-
4.964913845062256,11.359044075012207,11.662354469299316,-
20.583667755126953,-41.23126220703125,44.02782440185547,-
21.35416030883789,2.630840539932251,-22.316476821899414,-
31.268247604370117,41.736907958984375,-
9.192102432250977,10.637471199035645,-
16.764461517333984,30.29497528076172,14.864323616027832,-
30.267953872680664,-
37.85855484008789,31.97761344909668,17.948497772216797,52.141834259033
2,-
48.90068054199219,31.32318115234375,16.270051956176758,30.832414627075
195,17.158920288085938,45.68405532836914,28.04524040222168,43.45132446
2890625,2.653118848800659,27.02269172668457,-
17.754182815551758,27.824892044067383,-
17.54579734802246,44.96552658081055,15.9440336227417,50.06993865966797
,38.867713928222656,-
11.573134422302246,26.35179901123047,34.20499038696289,27.735189437866
21,-
15.116386413574219,31.182525634765625,13.187914848327637,27.2871246337
89062,-
19.06816291809082,23.71693992614746,45.096622467041016,24.157186508178
71,6.727110862731934,41.61668395996094,-
7.353477478027344,21.94973373413086,-18.696178436279297,-
3.4813153743743896,-21.77787208557129,-22.321107864379883,-
20.112510681152344,20.1693115234375,28.98859214782715,-
7.383764743804932,17.746826171875,-
38.67615509033203,57.08049774169922,-
1.7518894672393799,10.300661087036133,40.12282943725586,-
35.93798828125,0.5972323417663574,-33.5928840637207,-
21.545305252075195,11.8006010055542,-27.577428817749023,-
25.227909088134766,12.558523178100586,12.370092391967773,19.9252967834
47266,30.493030548095703,39.006004333496094,-34.41472244262695,-
23.806720733642578,19.634437561035156,-18.776857376098633,-
50.36948013305664,31.063880920410156,-
2.2893083095550537,43.151920318603516,32.47162628173828,-
19.86985206604004,34.55134963989258,-18.761371612548828,-
2.837064504623413,17.8626708984375,23.558320999145508,-
13.183303833007812,45.816532135009766,13.942610740661621,50.0175666809
082,23.3538818359375,-3.684291362762451,51.018917083740234,-
20.1840877532959,-19.849178314208984,-
12.584696769714355,39.498409271240234,-
57.371124267578125,36.17275619506836,44.6878662109375,24.1179370880126
95,10.98865795135498,-33.695491790771484,4.345941066741943,-
57.41758346557617,0.1664922833442688,-
18.816194534301758,6.6619062423706055,-6.837522029876709,-
25.133731842041016,-19.27800178527832],"yaxis":"y"},{"marker":
{"size":10},"mode":"markers","name":"Centroid of
True","type":"scatter","x":[-6.865907764233649],"y":[-
2.944843927666545]},{"marker":

{"size":10},"mode":"markers","name":"Centroid of
False","type":"scatter","x":[3.7120113233029843],"y":
[3.981281605720 52]}],"layout":{"legend":{"title":{"text":"our
rating"},"tracegroupgap":0},"margin":{"t":60},"template":{"data":
{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"type":"carpet"}],"ch
oropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],

```
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
```

```
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"#E5ECF6","showlake
s":true,"showland":true,"subunitcolor":"white"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"#E5ECF6","po
lar":{"angularaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","radialaxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"yaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"zaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"baxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","caxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2}}},"xaxis":
{"anchor":"y","domain":[0,1],"title":{"text":"0"}},"yaxis":
{"anchor":"x","domain":[0,1],"title":{"text":"1"}}}}
```