Projet SMS Spam Filtering

Rushan Zamir Saeedullah & Berville Laurence

Introduction :

Analyse du besoin

> Le service de messages courts (SMS) est le service de communication textuelle des systèmes de communication téléphonique, qu'un utilisateur peut utiliser pour communiquer avec d'autres utilisateurs. L'inconvénient, c'est que les téléphones portables sont en train de devenir la dernière cible du courrier électronique indésirable, avec un nombre croissant d'annonceurs à utiliser les messages texte pour cibler leurs abonnés. Pour le consommateur, le spam est un message indésirable, parfois répété, qui vise généralement à le tromper et à lui soutirer de l'argent par le biais d'une communication payante.

> Nous allons présenter ici nos travaux qui consistent en l'exploitation d'une base de donnée SMS, puis par la construction d'une application de détection de spam. Voici les étapes :

- Exploration de la base de donnée SMS,
- Construction d'un pipeline de ML,
- Prétraitement des données,
- Entrainement, fine tuning,
- Et validation et sélection d'un modèle de classification.

Mise en place de l'environnement

- Acquisition des données et mise en forme

> 1- Une collection de 425 messages de spam SMS a été extraite manuellement du site web Grumbletext. Il s'agit d'un forum britannique dans lequel les utilisateurs de téléphones portables font des déclarations publiques sur les messages de spam SMS, la plupart du temps sans signaler l'incident.

> 2- Egalement inclus dans le dataset, un sous-ensemble de 3 375 SMS, choisis au hasard dans le corpus NUS SMS, qui est un ensemble de données d'environ 10 000 messages légitimes collectés pour la recherche au département d'informatique de l'université nationale de Singapour. Les messages proviennent en grande partie de Singapouriens et surtout d'étudiants de l'université. Ces messages ont été collectés auprès de volontaires qui ont été informés que leurs contributions allaient être rendues publiques.

> 3- Une liste de 450 SMS spam recueillis dans la thèse de doctorat de Caroline Tag, disponible à l'adresse http://etheses.bham.ac.uk/253/1/Tagg09PhD.pdf.

> 4- Enfin, nous avons intégré le corpus SMS Spam v.0.1 Big. Il contient 1 002 SMS ham et 322 messages de spam et est accessible au public à l'adresse suivante : http://www.esp.uem.es/jmgomez/smsspamcorpus/.

- Pour ce projet, nous allons utiliser les packages ci-dessous :

In [2]:
```
import matplotlib.pyplot as plt # Graphiques
import seaborn as sns # Graphiques

from sklearn.pipeline import Pipeline # pour faire un pipeline
from sklearn.model_selection import train_test_split # pour diviser le dataset en t
from sklearn.feature_extraction.text import TfidfVectorizer # pour vectoriser les m
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

import pandas as pd # Gestion des dataframes
import numpy as np # Gestion des datasframes

import nltk# Preprocessing pour enlever ponctuation - Natural Language Processing
nltk.download("punkt")
nltk.download('stopwords')
from nltk.corpus import stopwords # Preprocessing pour enlever les mots "courant"

import string

import warnings
warnings.filterwarnings('ignore') # enlever les warnings de python
```

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

- Récupération des données :

In [3]:
```
data = pd.read_csv('https://raw.githubusercontent.com/remijul/dataset/master/SMSSpa
        sep='\t', header=None)
```

- Renommer les colonnes :

Les données téléchargées consistent en 2 colonnes. Une avec la définition des messages comme "Spam" ou "Ham", renommée "Target". La seconde colonne contient le message, renommer "SMS".

In [4]:
```
data=data.rename(columns={0:"Target",1:"SMS"}) # Renommer les colonnes.
```

Exploration de la base de donnée

- Répartition des variables dans le data set.

> Dans les éléments suivants nous présentons quelques statistiques descriptives de l'ensemble de données.

> En résumé, le dataset est composé de 4 825 messages légitimes et de 747 messages de spam mobile, soit un total de 5 572 messages. Nous constatons une forte disproportion entre les deux variables.

In [5]:
```
Ratio=data['Target'].value_counts()
Ratio= pd.DataFrame(Ratio)
Ratio
```

Out[5]:

	Target
ham	4825
spam	747

In [6]:
```
DeepnoteChart(data, """{"layer":[{"layer":[{"mark":{"clip":true,"type":"bar","toolt
```

Out[6]:
```
<__main__.DeepnoteChart at 0x7fb9636bbb50>
```

- Les doublons :

> Nous créons deux sous tableaux avec les données Ham et les données Spams. Puis, nous observons les messages.

In [7]:
```
HamData = data[data['Target'] == "ham"]
print(HamData.head(5))
```

```
  Target                                                SMS
0    ham  Go until jurong point, crazy.. Available only ...
1    ham                      Ok lar... Joking wif u oni...
3    ham  U dun say so early hor... U c already then say...
4    ham  Nah I don't think he goes to usf, he lives aro...
6    ham  Even my brother is not like to speak with me. ...
```

In [8]:
```
SpamData = data[data['Target'] == "spam"]
print(SpamData.head(5))
```

```
   Target                                                SMS
2    spam  Free entry in 2 a wkly comp to win FA Cup fina...
5    spam  FreeMsg Hey there darling it's been 3 week's n...
8    spam  WINNER!! As a valued network customer you have...
9    spam  Had your mobile 11 months or more? U R entitle...
11   spam  SIX chances to win CASH! From 100 to 20,000 po...
```

> Pour compter les spams et les observer.

In [9]:
```
HamData_counts= pd.DataFrame(HamData['SMS'].value_counts())# il y a des duplicats
HamData_counts
```

Out[9]:

SMS	
Sorry, I'll call later	30
I cant pick the phone right now. Pls send a message	12
Ok...	10
Say this slowly.? GOD,I LOVE YOU & I NEED YOU,CLEAN MY HEART WITH YOUR BLOOD.Send this to Ten special people & u c miracle tomorrow, do it,pls,pls do it...	4
Ok.	4
...	...
Can... I'm free...	1
Ok thats cool. Its , just off either raglan rd or edward rd. Behind the cricket ground. Gimme ring when ur closeby see you tuesday.	1
... we r stayin here an extra week, back next wed. How did we do in the rugby this weekend? Hi to and and , c u soon "	1
This weekend is fine (an excuse not to do too much decorating)	1
How much she payed. Suganya.	1

4516 rows × 1 columns

In [10]:
```
SpamData_counts= pd.DataFrame(SpamData['SMS'].value_counts())# il y a des duplicats
SpamData_counts
```

Out[10]:

SMS	
Please call our customer service representative on FREEPHONE 0808 145 4742 between 9am-11pm as you have WON a guaranteed £1000 cash or £5000 prize!	4
I don't know u and u don't know me. Send CHAT to 86688 now and let's find each other! Only 150p/Msg rcvd. HG/Suite342/2Lands/Row/W1J6HL LDN. 18 years or over.	3
HMV BONUS SPECIAL 500 pounds of genuine HMV vouchers to be won. Just answer 4 easy questions. Play Now! Send HMV to 86688 More info:www.100percent-real.com	3
FREE for 1st week! No1 Nokia tone 4 ur mob every week just txt NOKIA to 8007 Get txting and tell ur mates www.getzed.co.uk POBox 36504 W45WQ norm150p/tone 16+	3
Loan for any purpose £500 - £75,000. Homeowners + Tenants welcome. Have you been previously refused? We can still help. Call Free 0800 1956669 or text back 'help'	3
...	...
it to 80488. Your 500 free text messages are valid until 31 December 2005.	1
URGENT! We are trying to contact U. Todays draw shows that you have won a £800 prize GUARANTEED. Call 09050003091 from land line. Claim C52. Valid12hrs only	1
Do you want a new video handset? 750 anytime any network mins? Half Price Line Rental? Camcorder? Reply or call 08000930705 for delivery tomorrow	1
Call Germany for only 1 pence per minute! Call from a fixed line via access number 0844 861 85 85. No prepayment. Direct access! www.telediscount.co.uk	1
December only! Had your mobile 11mths+? You are entitled to update to the latest colour camera mobile for Free! Call The Mobile Update Co FREE on 08002986906	1

653 rows × 1 columns

> Il y a par exemple 30 messages identiques dans les "Hams".

> Avec".describe", nous pouvons observer les statistiques descriptives :

In [11]:
```
HamData_counts.describe()
```

Out[11]:

	SMS
count	4516.000000
mean	1.068423
std	0.563539
min	1.000000
25%	1.000000
50%	1.000000
75%	1.000000
max	30.000000

In [12]:
```
SpamData_counts.describe()
```

Out[12]:

	SMS
count	653.000000
mean	1.143951
std	0.384653
min	1.000000
25%	1.000000
50%	1.000000
75%	1.000000
max	4.000000

> Nous pouvons émettre trois hypothèses, à propos de la présence des doublons.

> 1- Comme nous l'avons noté dans l'introduction, il y a plusieurs datasets qui ont été combinés pour créer un, et donc certains messages sont en double (ou plus).

> 2- Les messages spams ont été envoyés à plusieurs personnes et ensuite déclarés plusieurs fois par les utilisateurs.

> 3- Le type de message, comme "ok", et souvent utilisés pas les rédacteurs de sms.

> Dans les deux premiers cas, nous considérons que les doublons doivent être retirés, mais pas dans le 3e cas. Dans le pipeline nous allons donc utiliser deux datasets. Un avec les doublons et l'autre sans.

- Nous souhaitons ensuite compter le nombre moyen de mot dans un spam et un ham.

In [13]:
```
HamData["Number of Words"] = HamData["SMS"].apply(
    lambda n: len(n.split()))
mean_Ham = HamData["Number of Words"].mean()
```

In [14]:
```
SpamData["Number of Words"] = SpamData['SMS'].apply(
    lambda n: len(n.split()))
mean_Spam = SpamData["Number of Words"].mean()
```

Explorations graphiques

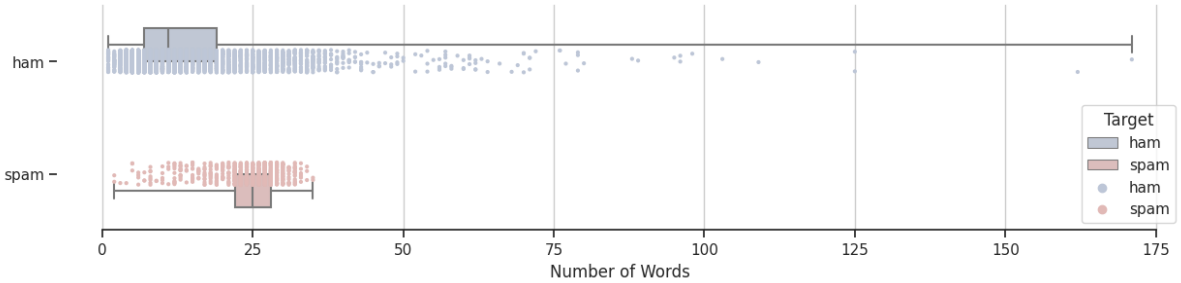
In [15]:
```
data["Number of Words"] = data["SMS"].apply(lambda n: len(n.split()))

sns.set_theme(style="ticks")# theme de la page.
f, ax = plt.subplots(figsize=(15, 3)) # taille de la page et initialisation de la f

# Plot the orbital period with horizontal boxes
sns.boxplot(
    data,
    x="Number of Words", #Variable x
     y="Target",  # variable y
     hue="Target",
    whis=[0, 100],
    width=.6, # taille des boxplot
    palette="vlag") # couleurs

# Rajouter les points de chaque observation.
sns.stripplot(data, x="Number of Words",
                y="Target",
                hue= "Target",
                size=3, # taille des points
                palette="vlag", # couleurs des points
                color=".5") # transparence

# Tweak the visual presentation
ax.xaxis.grid(True)# rajouter des grilles en background (x)
ax.set(ylabel="") # pas de titre pour l'axe y
sns.despine(trim=True, left=True)
```



> Nous notons que les hams ont des longueurs qui semblent plus variées, mais avec une moyenne plus faible que les spams.

Preprocessing

Enlever la ponctuation et les stopwords.

> En recherche d'information, un mot vide (ou stop word, en anglais) est un mot qui est tellement commun qu'il est inutile de l'indexer ou de l'utiliser dans une recherche. En d'autres termes, un mot qui apparaît avec une fréquence semblable dans chacun des textes de la collection n'est pas discriminant car il ne permet pas de distinguer les textes les uns par rapport aux autres. Ex : 'the', 'and', 'l', "a", "an", "the", et "of"

In [16]:
```
stop_words = list(stopwords.words('english'))
print(len(stop_words),stop_words)
```

```
179 ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're",
"you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him
', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its',
'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who',
'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'wer
e', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'do
ing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while
', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through
', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in',
'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 't
here', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',
'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', '
than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "s
hould've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'c
ouldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn',
"hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',
"mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "w
asn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

> Nous allons donc enlever 179 stopwords. Puis les ponctuations.

In [17]:
```
print(string.punctuation)
```

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

In [18]:
```
def text_process(text): # pour enlever les mots les plus courants et la ponctuation
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = [word for word in text.split() if word.lower() not in stopwords.words('e
    return " ".join(text)
```

In [19]:
```
df= pd.DataFrame(data['SMS'].apply(text_process))
```

In [20]:
```
label = pd.DataFrame(data['Target']) # Colonne avec que les Targets
df= pd.concat([label, df],axis=1)
```

Enlever les doublons

> Nous allons enlever les doublons. Puis, compter le nombre de mot dans les messages.

In [21]:
```
df = df.drop_duplicates()# enlever les doublons
```

In [22]:
```
df["Nb_Words"] = df["SMS"].apply(lambda n: len(n.split()))
df.head(5)
```

Out[22]:

	Target	SMS	Nb_Words
0	ham	Go jurong point crazy Available bugis n great ...	16
1	ham	Ok lar Joking wif u oni	6
2	spam	Free entry 2 wkly comp win FA Cup final tkts 2...	23
3	ham	U dun say early hor U c already say	9
4	ham	Nah dont think goes usf lives around though	8

Déclarer et encoder la target

In [23]:
```
from sklearn.preprocessing import LabelEncoder
lb_encod = LabelEncoder()
y = lb_encod.fit_transform(df['Target'])
```

In [24]:
```
y
```

Out[24]:
```
array([0, 0, 1, ..., 0, 0, 0])
```

Déclarer les features

In [26]:
```
X=pd.DataFrame(df['SMS'])
X.head(3)
```

Out[26]:

	SMS
0	Go jurong point crazy Available bugis n great ...
1	Ok lar Joking wif u oni
2	Free entry 2 wkly comp win FA Cup final tkts 2...

Conversion de mots en vecteurs : Feature extraction

> Nous pouvons convertir les mots en vecteurs en utilisant soit le vecteur de comptage, soit le vecteur TF-IDF.

> Le TF-IDF (de l'anglais term frequency-inverse document frequency) est meilleur que les vecteurs de comptage car il ne se concentre pas seulement sur la fréquence des mots présents dans le corpus, mais fournit également l'importance des mots. Nous pouvons alors supprimer les mots qui sont moins importants pour l'analyse, ce qui rend la construction du modèle moins complexe en réduisant les dimensions d'entrée.

In [27]:
```
# target preprocessing
tfidf_vectorizer  = TfidfVectorizer(use_idf=False,
                                    lowercase=True,
                                    strip_accents='ascii')
                                    #,stop_words=stop_words) # enlever les stop
```

Train test split

In [28]:
```
X_train, X_test, y_train, y_test = train_test_split(data['SMS'],
                                                    data['Target'],
                                                    test_size=0.20,# 20% test size
                                                    random_state=42,
                                                    stratify=data['Target']) #
```

In [29]:
```
X_train.shape
```

Out[29]:
```
(4457,)
```

In [31]:
```
y_train.shape
```

Out[31]:
```
(4457,)
```

In [32]:
```
y_test.shape
```

Out[32]:
```
(1115,)
```

In [30]:
```
X_test.shape
```

Out[30]: (1115,)

Classification à l'aide des classificateurs prédéfinis de Sklearn

> Dans cette étape, nous allons utiliser certains des classificateurs les plus populaires et comparer leurs résultats.

- Classification des spams à l'aide de la régression logistique
- Classification des spams à l'aide de SVM

In [33]:
```
from sklearn.linear_model import LogisticRegression
```

In [34]:
```
from sklearn.svm import SVC
```

- Classification des spams à l'aide de la méthode des bayes naïves
- Classification des spams à l'aide d'un arbre de décision

In [37]:
```
from sklearn.naive_bayes import MultinomialNB
```

In [38]:
```
from sklearn.tree import DecisionTreeClassifier
```

- Classification des spams à l'aide de K-Nearest Neighbor (KNN)
- Classification des spams à l'aide d'un Random Forest Classifier.

In [39]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [40]:
```
from sklearn.ensemble import RandomForestClassifier
```

> Nous allons tester les 6 modèles :

In [ ]:
```
svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier(n_neighbors=49)
mnb = MultinomialNB(alpha=0.2)
dtc = DecisionTreeClassifier(min_samples_split=7, random_state=111)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=31, random_state=111)
```

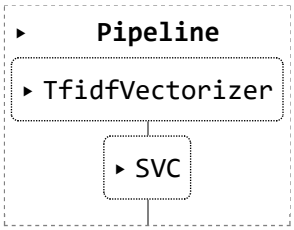
Pipelines

In [ ]:
```
SVC_vectorizer = Pipeline([
     ('vectorizer', tfidf_vectorizer),
     ('classifier', svc)
 ])
```

In [ ]:
```
KNC_vectorizer = Pipeline([
     ('vectorizer', tfidf_vectorizer),
     ('classifier',knc )
 ])
```

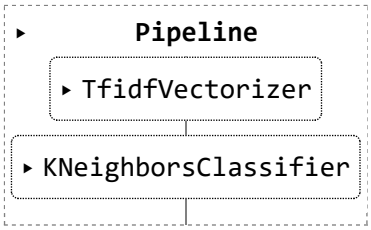
In [ ]:
```
SVC_vectorizer.fit(X_train, y_train)
```

Out[ ]:



In [ ]:
```
KNC_vectorizer.fit(X_train, y_train)
```

Out[ ]:



In [ ]:
```
y_train_pred_SVC = SVC_vectorizer.predict(X_train)
print(f"Train Accuracy using Count Vectorizer: {accuracy_score(y_train,y_train_pred
```

```
Train Accuracy using Count Vectorizer: 0.987
```

In [ ]:
```
y_train_pred_KNC = KNC_vectorizer.predict(X_train)
print(f"Train Accuracy using Count Vectorizer:{accuracy_score(y_train,y_train_pred_
```

```
Train Accuracy using Count Vectorizer:0.949
```

In [ ]:
```
MNB_vectorizer = Pipeline([
     ('vectorizer', tfidf_vectorizer),
     ('classifier', mnb)
 ]).fit(X_train, y_train)

y_train_pred_MNB = MNB_vectorizer.predict(X_train)
print(f"Train Accuracy using Count Vectorizer: {accuracy_score(y_train,y_train_pred
```

```
Train Accuracy using Count Vectorizer: 0.990
```

In [ ]:
```
DTC_vectorizer = Pipeline([
     ('vectorizer', tfidf_vectorizer),
     ('classifier', dtc)
 ]).fit(X_train, y_train)

y_train_pred_DTC = DTC_vectorizer.predict(X_train)
print(f"Train Accuracy using Count Vectorizer: {accuracy_score(y_train,y_train_pred
```

```
Train Accuracy using Count Vectorizer: 0.997
```

In [ ]:
```
LRC_vectorizer = Pipeline([
     ('vectorizer', tfidf_vectorizer),
     ('classifier', lrc)
 ]).fit(X_train, y_train)

y_train_pred_LRC = LRC_vectorizer.predict(X_train)
print(f"Train Accuracy using Count Vectorizer: {accuracy_score(y_train,y_train_pred
```

```
Train Accuracy using Count Vectorizer: 0.974
```

In [ ]:
```
RFC_vectorizer = Pipeline([
     ('vectorizer', tfidf_vectorizer),
     ('classifier', rfc)
 ]).fit(X_train, y_train)

y_train_pred_RFC = RFC_vectorizer.predict(X_train)
print(f"Train Accuracy using Count Vectorizer: {accuracy_score(y_train,y_train_pred
```

```
Train Accuracy using Count Vectorizer: 1.000
```

Rapports de la classification

In [ ]:
```
print(classification_report(y_train, y_train_pred_SVC))
```

```
              precision    recall  f1-score   support

         ham       0.99      1.00      0.99      3859
        spam       0.98      0.92      0.95       598

    accuracy                           0.99      4457
   macro avg       0.98      0.96      0.97      4457
weighted avg       0.99      0.99      0.99      4457
```

In [ ]:
```
print(classification_report(y_train, y_train_pred_KNC))
```

```
              precision    recall  f1-score   support

         ham       0.94      1.00      0.97      3859
        spam       0.99      0.62      0.76       598

    accuracy                           0.95      4457
   macro avg       0.97      0.81      0.87      4457
weighted avg       0.95      0.95      0.94      4457
```

In [ ]:
```
print(classification_report(y_train, y_train_pred_MNB))
```

```
              precision    recall  f1-score   support

         ham       0.99      1.00      0.99      3859
        spam       1.00      0.92      0.96       598

    accuracy                           0.99      4457
   macro avg       0.99      0.96      0.98      4457
weighted avg       0.99      0.99      0.99      4457
```

In [ ]:
```
print(classification_report(y_train, y_train_pred_DTC))
```

```
              precision    recall  f1-score   support

         ham       1.00      1.00      1.00      3859
        spam       0.99      0.98      0.99       598

    accuracy                           1.00      4457
   macro avg       1.00      0.99      0.99      4457
weighted avg       1.00      1.00      1.00      4457
```

In [ ]:
```
print(classification_report(y_train, y_train_pred_LRC))
```

```
              precision    recall  f1-score   support

         ham       0.98      0.99      0.99      3859
        spam       0.96      0.85      0.90       598

    accuracy                           0.97      4457
   macro avg       0.97      0.92      0.94      4457
weighted avg       0.97      0.97      0.97      4457
```

In [ ]:
```
print(classification_report(y_train, y_train_pred_RFC))
```

```
              precision    recall  f1-score   support

         ham       1.00      1.00      1.00      3859
        spam       1.00      1.00      1.00       598

    accuracy                           1.00      4457
   macro avg       1.00      1.00      1.00      4457
weighted avg       1.00      1.00      1.00      4457
```

Matrices de confusion

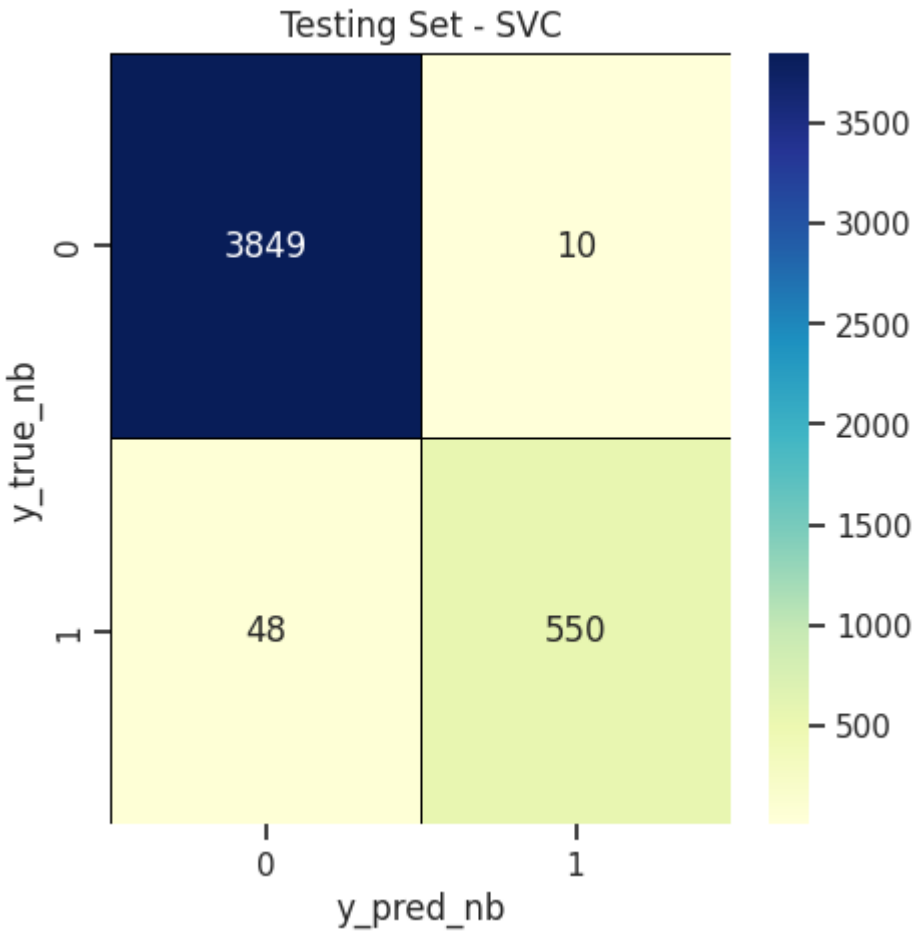
Les nombres en diagonale sont liés aux prédictions correctes, tandis que les nombres hors diagonale sont liés aux prédictions incorrectes (mauvaises classifications). Nous connaissons maintenant les quatre types de prédictions correctes et erronées :

le coin supérieur gauche correspond aux vrais positifs (TP), c'est-à-dire aux personnes qui ont donné du sang et qui ont été prédites comme telles par le classificateur ;

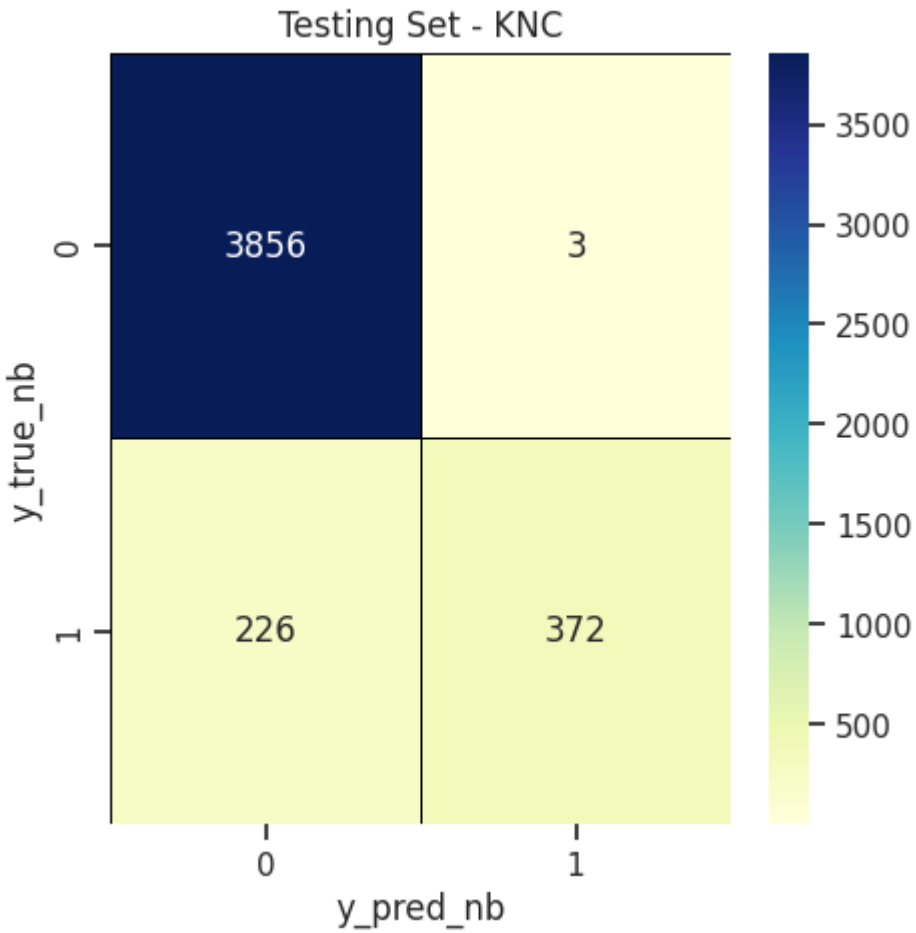
le coin inférieur droit est constitué de vrais négatifs (TN) et correspond aux personnes qui n'ont pas donné de sang et qui ont été prédites comme telles par le classificateur ;

le coin supérieur droit correspond aux faux négatifs (FN) et correspond aux personnes qui ont donné du sang mais qui ont été prédites comme n'ayant pas donné de sang ;

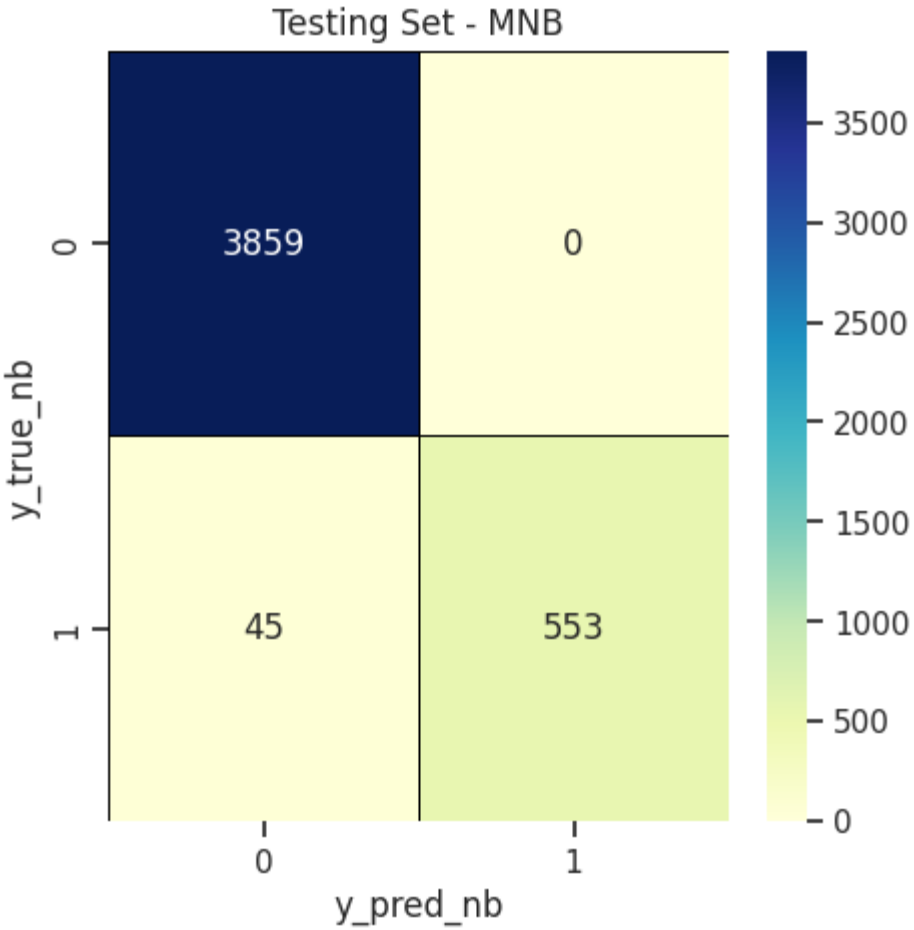
In [ ]:
```
cm = confusion_matrix(y_train,y_train_pred_SVC)
f, ax = plt.subplots(figsize =(5,5))
sns.heatmap(cm,annot = True,
            linewidths=0.5,
            linecolor="black",
            fmt = ".0f",
            cmap="YlGnBu",
            ax=ax)
plt.xlabel("y_pred_nb")
plt.ylabel("y_true_nb")
ax.set_title('Testing Set - SVC')
plt.show()
```



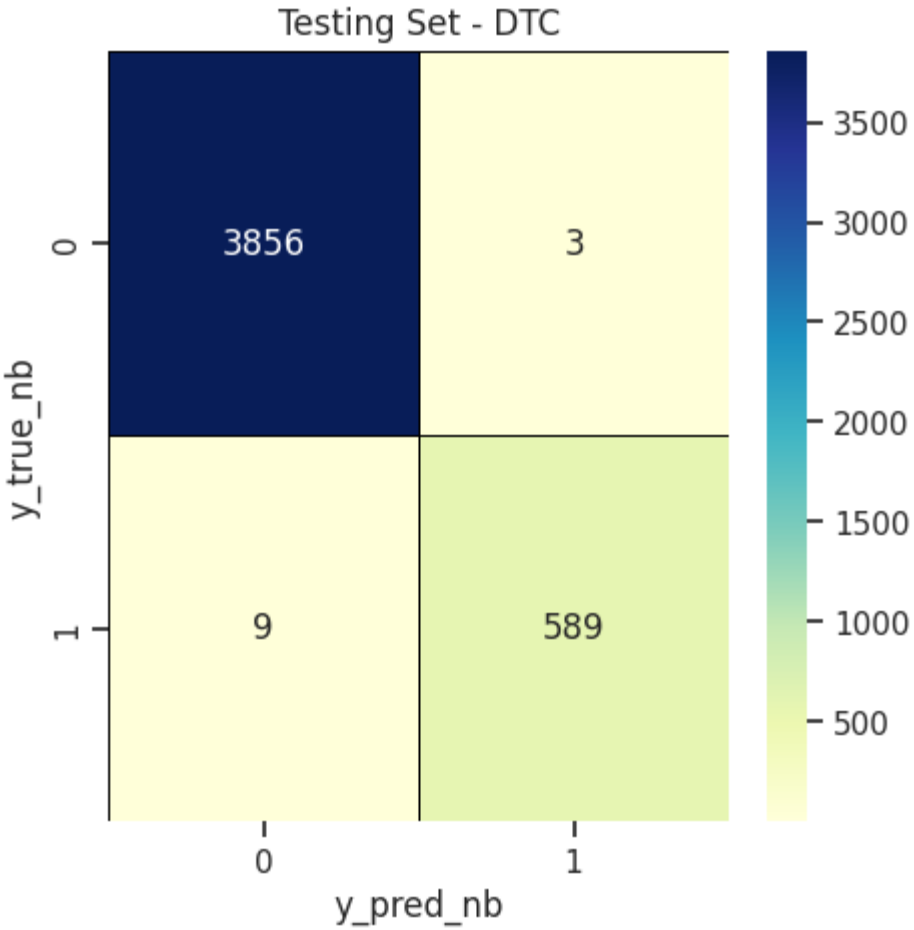
In [ ]:
```
cm = confusion_matrix(y_train,y_train_pred_KNC)
f, ax = plt.subplots(figsize =(5,5))
sns.heatmap(cm,annot = True,
            linewidths=0.5,
            linecolor="black",
            fmt = ".0f",
            cmap="YlGnBu",
            ax=ax)
plt.xlabel("y_pred_nb")
plt.ylabel("y_true_nb")
ax.set_title('Testing Set - KNC')
plt.show()
```



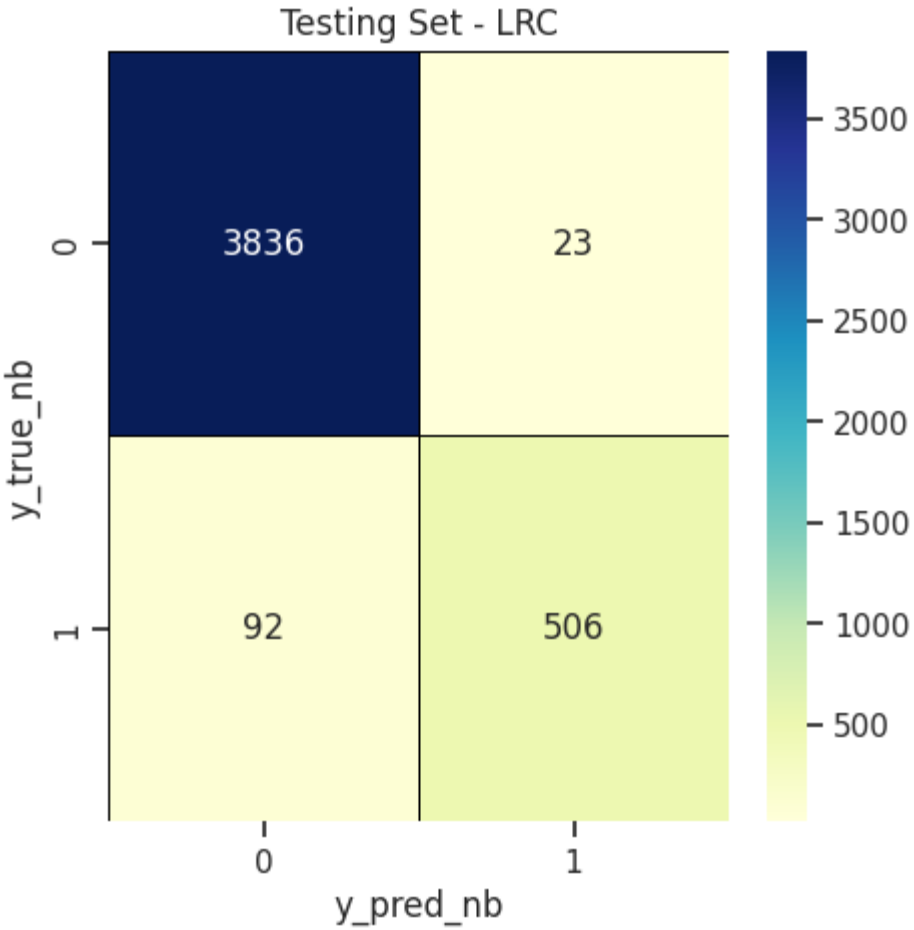
In [ ]:
```
cm = confusion_matrix(y_train,y_train_pred_MNB)
f, ax = plt.subplots(figsize =(5,5))
sns.heatmap(cm,annot = True,
            linewidths=0.5,
            linecolor="black",
            fmt = ".0f",
            cmap="YlGnBu",
            ax=ax)
plt.xlabel("y_pred_nb")
plt.ylabel("y_true_nb")
ax.set_title('Testing Set - MNB')
plt.show()
```



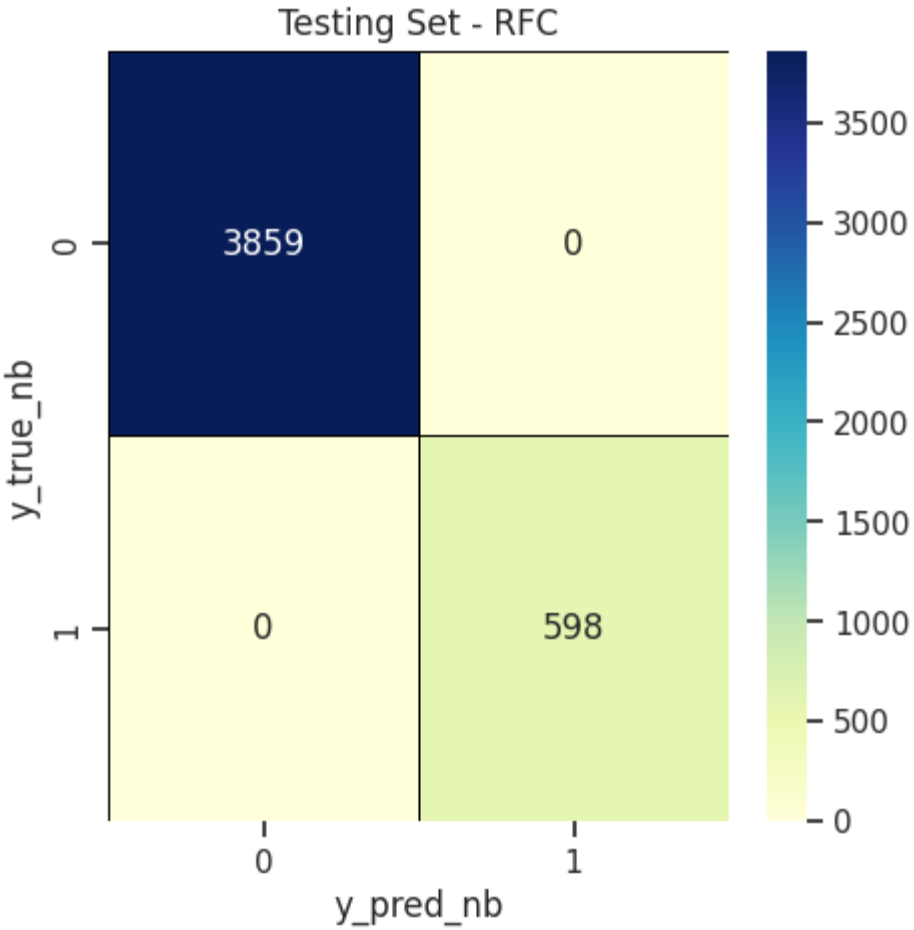
In [ ]:
```
cm = confusion_matrix(y_train,y_train_pred_DTC)
f, ax = plt.subplots(figsize =(5,5))
sns.heatmap(cm,annot = True,
            linewidths=0.5,
            linecolor="black",
            fmt = ".0f",
            cmap="YlGnBu",
            ax=ax)
plt.xlabel("y_pred_nb")
plt.ylabel("y_true_nb")
ax.set_title('Testing Set - DTC')
plt.show()
```



In [ ]:
```
cm = confusion_matrix(y_train,y_train_pred_LRC)
f, ax = plt.subplots(figsize =(5,5))
sns.heatmap(cm,annot = True,
            linewidths=0.5,
            linecolor="black",
            fmt = ".0f",
            cmap="YlGnBu",
            ax=ax)
plt.xlabel("y_pred_nb")
plt.ylabel("y_true_nb")
ax.set_title('Testing Set - LRC')
plt.show()
```



In [ ]:
```
cm = confusion_matrix(y_train,y_train_pred_RFC)
f, ax = plt.subplots(figsize =(5,5))
sns.heatmap(cm,annot = True,
            linewidths=0.5,
            linecolor="black",
            fmt = ".0f",
            cmap="YlGnBu",
            ax=ax)
plt.xlabel("y_pred_nb")
plt.ylabel("y_true_nb")
ax.set_title('Testing Set - RFC')
plt.show()
```



Comparaison des models

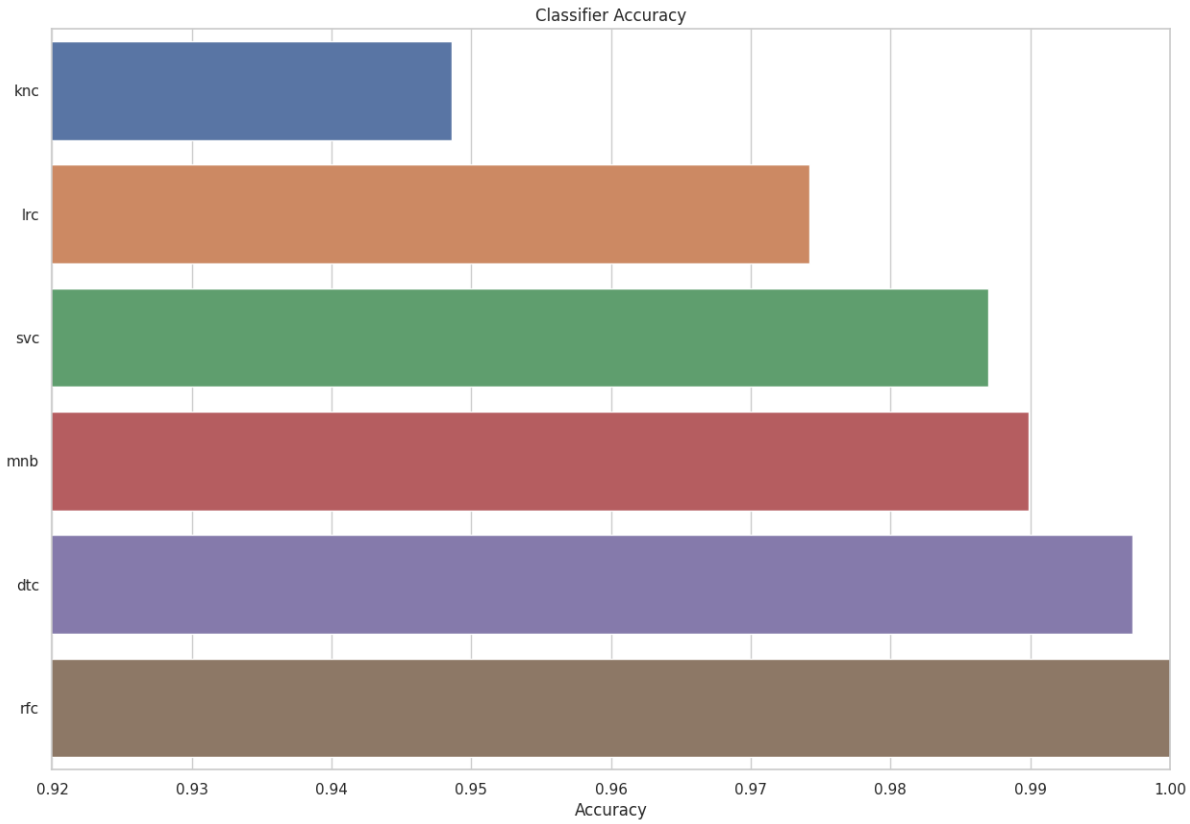
In [ ]:
```
Classifier = [svc, knc,  mnb,dtc, lrc, rfc]
Accuracy=(accuracy_score(y_train,y_train_pred_SVC),
            accuracy_score(y_train,y_train_pred_KNC),
            accuracy_score(y_train,y_train_pred_MNB),
            accuracy_score(y_train,y_train_pred_DTC),
            accuracy_score(y_train,y_train_pred_LRC),
            accuracy_score(y_train,y_train_pred_RFC))
```

In [ ]:
```
df = pd.DataFrame( Accuracy,
              index =  ["svc", "knc",  "mnb","dtc", "lrc", "rfc"])
df = df.reset_index()
df=df.rename(columns={0:"Valeur", "index":"Model"}) # Renommer le nom des variables
df
```

Out[ ]:

	Model	Valeur
0	svc	0.986987
1	knc	0.948620
2	mnb	0.989904
3	dtc	0.997308
4	lrc	0.974198
5	rfc	1.000000

In [ ]:
```
fig = plt.figure(figsize=(15, 10))
orderedA= df.sort_values(by='Valeur')# ordination
sns.set_theme(style="whitegrid")
sns.barplot(x="Valeur",y="Model",data=orderedA)
plt.xlim([0.92, 1])
plt.xlabel('Accuracy')
plt.ylabel('')
plt.title('Classifier Accuracy')
plt.show()
```



Prédictions

test (avec stratificaion) > prédiction avec nouvelles données.

Un modèle de prédiction en Python est un algorithme mathématique utilisé pour faire des prédictions ou des prévisions basées sur des données d'entrée. Il utilise l'apprentissage automatique ou des techniques statistiques pour analyser des données historiques et apprendre des modèles, qui peuvent ensuite être utilisés pour prédire des résultats ou des tendances futurs.

Nous allons ici utiliser le model ayant le meilleur résultat : RFC

In [ ]:
```
#print ('Veuillez entrer votre sms suspect : ')
#new_sms =input()
```

```
Veuillez entrer votre sms suspect :
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In [61], line 2
      1 print ('Veuillez entrer votre sms suspect : ')
----> 2 new_sms =input()

File /shared-libs/python3.9/py-core/lib/python3.9/site-packages/ipykernel/kernelbas
e.py:1177, in Kernel.raw_input(self, prompt)
   1173 if not self._allow_stdin:
   1174     raise StdinNotImplementedError(
   1175         "raw_input was called, but this frontend does not support input req
uests."
   1176     )
-> 1177 return self._input_request(
   1178     str(prompt),
   1179     self._parent_ident["shell"],
   1180     self.get_parent("shell"),
   1181     password=False,
   1182 )

File /shared-libs/python3.9/py-core/lib/python3.9/site-packages/ipykernel/kernelbas
e.py:1219, in Kernel._input_request(self, prompt, ident, parent, password)
   1216             break
   1217 except KeyboardInterrupt:
   1218     # re-raise KeyboardInterrupt, to truncate traceback
-> 1219     raise KeyboardInterrupt("Interrupted by user") from None
   1220 except Exception:
   1221     self.log.warning("Invalid Message:", exc_info=True)

KeyboardInterrupt: Interrupted by user
```

In [ ]:
```
prediction = RFC_vectorizer.predict([new_sms])

if prediction[0] == "spam":
    print("This sms is spam.")
else:
    print("This sms is not spam.")
```

```
This email is spam.
```