EL RAKAAWI HANI

5 janvier 2023

- etape1 :

1. telechargement de la base de donnees :(download dataset)

In [282]:
```
import pandas as pd

df = pd.read_csv('spambase.txt', sep='\t')

df.head(10)
```

Out[282]:

	wf_make	wf_address	wf_all	wf_3d	wf_our	wf_over	wf_remove	wf_internet	wf_order	wf_m
0	0.00	0.52	0.52	0.0	0.52	0.00	0.00	0.00	0.00	0.
1	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.
2	0.00	0.00	0.66	0.0	0.00	0.66	0.00	0.00	0.00	0.
3	0.08	0.00	0.16	0.0	0.00	0.08	0.00	0.08	0.73	0.
4	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.
5	0.00	0.36	0.00	0.0	0.00	0.36	1.47	0.00	0.00	0.
6	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.
7	0.00	0.00	1.85	0.0	0.00	0.00	0.00	0.00	0.00	1.
8	0.00	0.00	0.53	0.0	0.00	0.53	0.00	0.00	0.00	0.
9	0.40	0.40	0.26	0.0	0.13	0.20	0.06	0.33	0.00	1.

10 rows × 57 columns

In [283]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4601 entries, 0 to 4600
Data columns (total 57 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   wf_make                     4601 non-null   float64
 1   wf_address                  4601 non-null   float64
 2   wf_all                      4601 non-null   float64
 3   wf_3d                       4601 non-null   float64
 4   wf_our                      4601 non-null   float64
 5   wf_over                     4601 non-null   float64
 6   wf_remove                   4601 non-null   float64
 7   wf_internet                 4601 non-null   float64
 8   wf_order                    4601 non-null   float64
 9   wf_mail                     4601 non-null   float64
 10  wf_receive                  4601 non-null   float64
 11  wf_will                     4601 non-null   float64
 12  wf_people                   4601 non-null   float64
 13  wf_report                   4601 non-null   float64
 14  wf_addresses                4601 non-null   float64
 15  wf_free                     4601 non-null   float64
 16  wf_business                 4601 non-null   float64
 17  wf_email                    4601 non-null   float64
 18  wf_you                      4601 non-null   float64
 19  wf_credit                   4601 non-null   float64
 20  wf_your                     4601 non-null   float64
 21  wf_font                     4601 non-null   float64
 22  wf_000                      4601 non-null   float64
 23  wf_money                    4601 non-null   float64
 24  wf_hp                       4601 non-null   float64
 25  wf_hpl                      4601 non-null   float64
 26  wf_lab                      4601 non-null   float64
 27  wf_labs                     4601 non-null   float64
 28  wf_telnet                   4601 non-null   float64
 29  wf_857                      4601 non-null   float64
 30  wf_data                     4601 non-null   float64
 31  wf_415                      4601 non-null   float64
 32  wf_85                       4601 non-null   float64
 33  wf_technology               4601 non-null   float64
 34  wf_1999                     4601 non-null   float64
 35  wf_parts                    4601 non-null   float64
 36  wf_pm                       4601 non-null   float64
 37  wf_direct                   4601 non-null   float64
 38  wf_cs                       4601 non-null   float64
 39  wf_meeting                  4601 non-null   float64
 40  wf_original                 4601 non-null   float64
 41  wf_project                  4601 non-null   float64
 42  wf_re                       4601 non-null   float64
 43  wf_edu                      4601 non-null   float64
 44  wf_table                    4601 non-null   float64
 45  wf_conference               4601 non-null   float64
 46  cf_comma                    4601 non-null   float64
 47  cf_bracket                  4601 non-null   float64
 48  cf_sqbracket                4601 non-null   float64
 49  cf_exclam                   4601 non-null   float64
 50  cf_dollar                   4601 non-null   float64
 51  cf_hash                     4601 non-null   float64
```

```
 52   capital_run_length_average  4601 non-null   float64
 53   capital_run_length_longest  4601 non-null   int64
 54   capital_run_length_total    4601 non-null   int64
 55   spam                        4601 non-null   object
 56   status                      4601 non-null   object
dtypes: float64(53), int64(2), object(2)
memory usage: 2.0+ MB
```

In [284]:
```
df_num = df.drop(columns=['spam', 'status'])
df_num
```

Out[284]:

	wf_make	wf_address	wf_all	wf_3d	wf_our	wf_over	wf_remove	wf_internet	wf_order	w
0	0.00	0.52	0.52	0.0	0.52	0.00	0.00	0.00	0.00	
1	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	
2	0.00	0.00	0.66	0.0	0.00	0.66	0.00	0.00	0.00	
3	0.08	0.00	0.16	0.0	0.00	0.08	0.00	0.08	0.73	
4	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	
...	...	...	...	...	...	...	...	...	...	
4596	0.00	0.44	0.44	0.0	0.00	0.00	0.00	0.00	0.00	
4597	0.28	0.14	0.14	0.0	0.00	0.00	0.14	0.00	0.42	
4598	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	
4599	0.32	0.00	1.64	0.0	0.98	0.00	0.32	0.00	0.65	
4600	0.00	0.00	0.00	0.0	0.43	0.43	0.43	0.00	0.00	

4601 rows × 55 columns

2 .repartistion de la base de donnees d'entrainement et de donnees de test :

In [285]:
```
spam_train_all=df[df.status=="train"]
spam_test_all=df[df.status=="test"]

X_train=spam_train_all.drop(columns=['spam', 'status'])
X_test=spam_test_all.drop(columns=['spam', 'status'])
```

3. Réalisez une standardisation des deux sous-ensembles des données

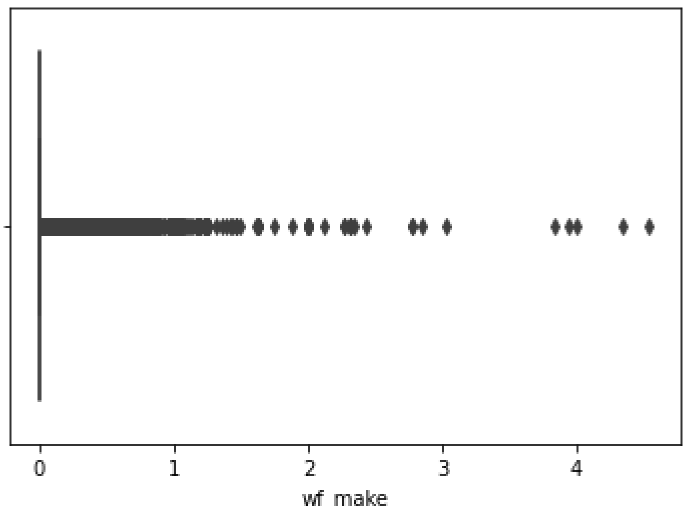
In [286]:
```
#df.isnull().sum()

missing_row = df[df.isnull().any(axis=1)]
missing_row
```

Out[286]:

wf_make	wf_address	wf_all	wf_3d	wf_our	wf_over	wf_remove	wf_internet	wf_order	wf_ma

0 rows × 57 columns

In [287]:
```
import seaborn as sns
import matplotlib.pyplot as plt
#on verifie si y a des outliers


sns.boxplot(df["wf_make"])

# Show the plot
plt.show()
```



In [288]:
```
from sklearn.preprocessing import RobustScaler

scaler = RobustScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

In [289]:
```
print(X_train.shape)
print(X_test.shape)
```

```
(3601, 55)
(1000, 55)
```

4.Déterminez la taille des deux sous-ensembles de données

In [290]:
```
# donnees de test
print(f"le nombre de vars est : {X_test.shape[1]}")
print(f"le nombre de individual est : {X_test.shape[0]}")
```

```
le nombre de vars est : 55
le nombre de individual est : 1000
```

In [291]:
```
# donnees d<entrainements
print(f"le nombre de vars est : {X_train.shape[1]}")
print(f"le nombre de individual est : {X_train.shape[0]}")
```

```
le nombre de vars est : 55
le nombre de individual est : 3601
```

In [ ]:

5. diagramme de dispersion de paires de variables par classes

(spam), représentez la dispersion des 6 premières variables

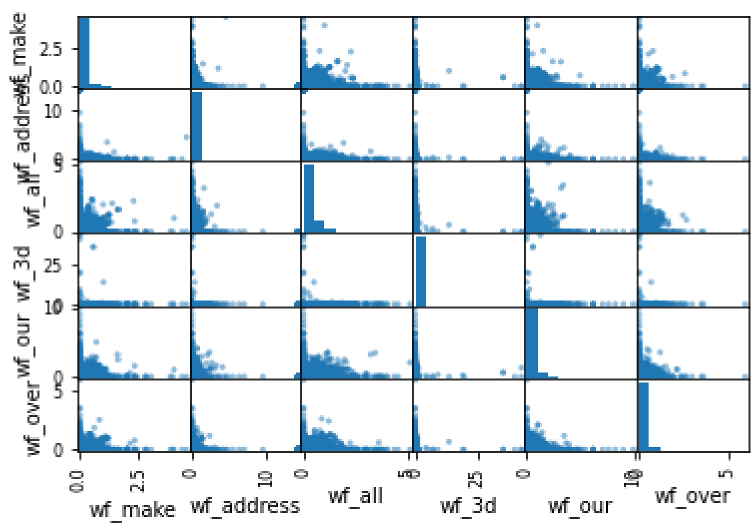
In [292]:
```
from pandas.plotting import scatter_matrix
```

In [293]:
```
plt.figure(figsize=(50,50))
featur = ['wf_make','wf_address','wf_all','wf_3d','wf_our','wf_over']
scatter_matrix(df[featur])
```

Out[293]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fba9d73a160
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9da91cd0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9d8a3430
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9d8edbb0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9d922370
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9e297a30
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fba9e297b20
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fbabb7be340
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9d97d1f0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9d9a6970
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9e41d130
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9e4458b0
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fba9e4700d0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9e7d17f0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9e7fbf70
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9e830730
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9e85ceb0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9e892670
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fba9e8bbdf0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9e9585b0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9e982d30
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9e9b94f0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9e9e2c70
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9ea19430
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fba9ea43bb0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9ea7b370
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9eaa4af0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9eada2b0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9eb06a30
```

```
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9eb3d1f0
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fba9eb65970
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9eb9e130
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9ebc68b0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9ebf00d0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9ec277f0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fba9ec52f70
>]],
      dtype=object)
```

```
<Figure size 3600x3600 with 0 Axes>
```



-6 des informations préliminaires sur l’importance

(pouvoir discriminant) de ces variables

selons les graphes on remarque que y a une forte correlation entre certaines features comme wf_our et wf_all

et d'autres qui ne l'ónt pas' comme wf_adresse et wf_3d

In [ ]:

etape 2 :

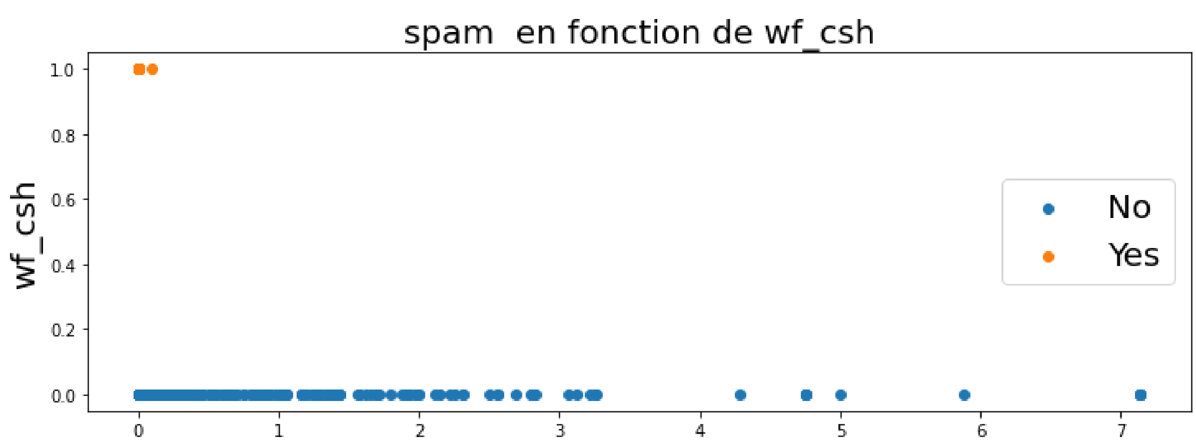
1. Réalisez un encodage de la variable cible en vue d’une régression logistique.m

In [294]:
```
dfetape2 = df.copy()
from sklearn.preprocessing import LabelEncoder
sc = RobustScaler()
X = dfetape2['wf_cs']
X = sc.fit_transform(pd.DataFrame(X))
Y = dfetape2['spam']
le = LabelEncoder()
Y = le.fit_transform(Y)
```

In [295]:
```
y_train =spam_train_all["spam"].replace({'yes':1,'no':0})
y_test =spam_test_all["spam"].replace({'yes':1,'no':0})
```

2.Représentez la dispersion de la variable cible (spam) encodée en fonction de la variable wf_cs.

In [296]:
```
plt.figure(figsize=(12,4))
plt.scatter(X[Y==0,0], Y[Y==0], label="No")
plt.scatter(X[Y==1,0], Y[Y==1], label="Yes", )
plt.xlabel('Y',fontsize=2)
plt.ylabel('wf_csh',fontsize=20)
plt.title('spam  en fonction de wf_csh' ,fontsize=20)
plt.legend(loc="center right", fontsize=20)
plt.show()
```



3.En considérant la variable wf_cs, entrainez un modèle de régressionlogistique sur l'ensemble des données d'entrainement (Standardisées).(random_state=20)

In [297]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split



# Crer le model regression logistique
log_reg = LogisticRegression(max_iter =1000 ,random_state =42)

# entrainement le model sur les donnees d'entrainement
log_reg.fit(X_train, y_train)
```

Out[297]:
```
LogisticRegression(max_iter=1000, random_state=42)
```

4 - Déterminez les paramètres du modèle et écrivez l'équation du modèle

de régression.

In [298]:
```
print(f"le coefficient est de valeur {log_reg.coef_}")
print(f"lintercept est de valeur {log_reg.intercept_}")
```

```
le coefficient est de valeur [[-1.87290172e-01 -1.15067551e-01  1.10009
537e-01  1.07851947e+00
   2.16634748e-01  6.03948059e-01  2.45363136e+00  7.48753840e-01
   4.18500518e-01  2.55282565e-02  2.47255901e-02 -1.03119966e-01
   1.89302183e-02  2.64029162e-01  4.95910157e-01  8.90255735e-02
   1.18504276e+00  1.87208514e-01  2.85531887e-01  7.62840970e-01
   3.05287916e-01  3.21204803e-01  1.91324071e+00  4.49413616e-01
  -1.85399225e+00 -6.60353305e-01 -1.22947363e+00 -1.77671336e-01
  -8.82663186e-01 -3.02224859e-01 -9.55006959e-01 -4.49905671e-01
  -1.41920656e+00  7.22107870e-01 -3.44553385e-01 -9.69964453e-02
  -4.55571443e-01 -7.06480127e-01 -1.42343030e+00 -1.60598278e+00
  -1.02460694e+00 -1.17464882e+00 -9.66961469e-02 -1.41350335e+00
  -9.11195491e-01 -1.65118299e+00 -1.21132868e+00 -1.83450736e-03
  -4.57047751e-01  1.17541532e-01  3.77930232e-01  1.06815448e+00
  -3.30052713e-02  3.51818406e-01  1.75750682e-01]]
lintercept est de valeur [-1.55089411]
```

5-la frontiere de decision

6- Déterminez la probabilité d'appartenance à la classe (spam) de l'observation

dont la valeur de wf_cs = 1.07

ETAPE 3 :

In [299]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(3601, 55)
(1000, 55)
(3601,)
(1000,)
```

In [300]:
```
from sklearn.linear_model import Perceptron
from sklearn.model_selection import train_test_split
```

1. En utilisant la libraire Sklearn, développez un perceptron simple pourprédire la classe (spam) (random_state=100, max_iter = 1500).

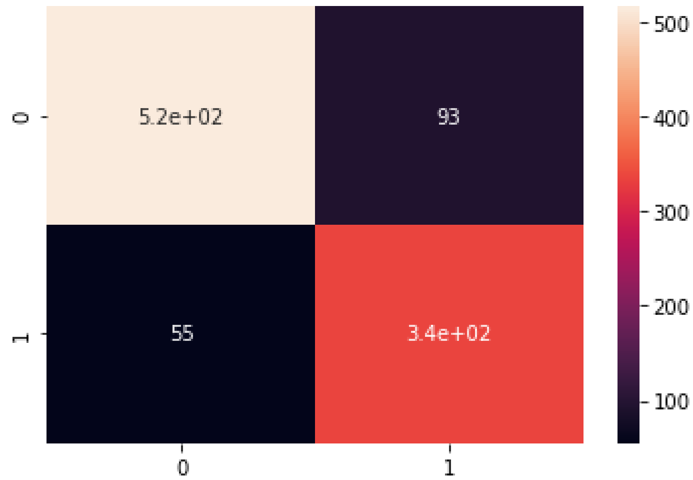
In [301]:
```
# Créez un perceptron avec un état aléatoire de 100 et un nombre maximal
d'itérations de 1500
perceptron = Perceptron(random_state=100, max_iter=1500)

# Entraînez le perceptron sur les données d'entraînement
perceptron.fit(X_train, y_train)
```

Out[301]:
```
Perceptron(max_iter=1500, random_state=100)
```

2. Représentez la matrice de confusion et évaluez les performances. Déterminez la valeur du score F1.

In [302]:
```
import seaborn as sns
sns.heatmap(confusion_matrix ,annot=True)
```

Out[302]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbab8a938b0>



3. Quelle est la valeur du biais du modèle de perceptron obtenu ?

In [303]:
```
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.linear_model import Perceptron

# Utilisez le perceptron entraîné pour prédire
y_pred = perceptron.predict(X_test)

# Représentez la matrice de confusion en utilisant les étiquettes de cla
sse prédites et réelles
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
[[516  93]
 [ 55 336]]
```

In [304]:
```
perceptron.score(X_train,y_train )
```

Out[304]: 0.8797556234379339

In [305]:
```
pred=perceptron.predict(X_test)
print(pred)
```

```
[1 1 0 0 1 0 1 0 0 0 1 0 0 1 1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0
1 0
 1 0 1 1 1 1 0 1 1 1 0 1 0 0 1 1 0 0 1 0 0 1 1 0 1 0 1 0 0 0 1 1 0 1 0
1 0
 0 0 1 0 0 1 1 1 0 1 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 1 1 1 0 0 1 0 1 0
1 1
 1 1 1 0 0 1 1 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 1 0
0 0
 1 1 1 0 0 0 0 0 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 1 0 1 0 1 1 0 0 0 0 1 0
1 1
 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 1 0 1 1 1 1 0 0 0 1 0 1 0 1 1 0 0 1 0 1
0 1
 1 0 1 1 1 1 1 0 1 1 1 0 1 0 0 0 1 1 0 1 1 1 1 0 0 1 1 0 1 0 1 1 0 0 1
1 1
 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0
0 1
 1 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 1 0 1 1 0 0 0 1 0 1 1 0 0 1 0 1 1
1 1
 0 0 1 0 0 0 0 1 1 0 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 0 0 1 0 0 0 1 0 0 0
0 1
 1 0 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0
1 1
 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0 1 1 0 1 1 1 0 0 1 1 1 0 0 0 1 0 1 0 0 1
1 0
 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 1 0 0 0 1 0 0 1 1
1 0
 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 0 1 0 1 1 0 0 1 1 0 1 0 1 0 0 0 1 0 0 1
0 1
 0 1 1 1 0 1 0 0 1 1 1 0 1 0 0 1 1 1 1 1 1 0 0 0 0 1 0 1 1 1 0 1 0 1 1
0 0
 1 0 0 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 0 0 1 0 1 0 1
1 0
 1 0 0 0 1 1 0 0 0 0 1 1 0 0 1 1 0 0 1 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0
0 1
 0 0 0 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 1 0 0 0 0 0 0 1 1 0 1 1 0 1 0 1 1
1 0
 0 0 1 0 0 1 0 1 0 0 1 0 0 0 1 1 0 1 1 1 0 1 1 0 0 0 0 0 0 0 1 0 1 0 0
1 1
 0 0 0 1 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 1 1 1 0 0 0 1 0
1 0
 0 0 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 1 1 1 0 0 0 1 1 0 0 0 1 1
0 1
 0 0 0 1 0 1 1 0 0 1 0 0 0 1 0 1 1 0 0 1 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0
1 0
 0 1 0 1 1 0 0 1 1 0 0 0 1 1 0 0 0 0 1 0 1 0 1 0 0 1 0 1 1 1 0 0 1 0 1
1 0
 1 1 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 1 0 1 1 0 0
1 0
 0 0 0 0 0 0 1 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 1 1 0
0 0
 0 1 0 0 1 0 1 1 1 0 0 1 1 1 0 1 0 1 0 1 0 0 0 1 1 0 0 0 1 1 1 0 1 0 0
0 1
 1 1 0 0 0 1 1 0 1 0 0 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 1 1 0 0 0 1 1
0 1
 1]
```

In [306]:
```
from sklearn.metrics import accuracy_score
import sklearn.metrics as metrics
print("l'erreur esr  : "+str(1-metrics.accuracy_score(y_test,pred)))
```

```
l'erreur esr  : 0.14800000000000002
```

4. Combien de paramètres possède ce modèle. Pourquoi ?

In [307]:
```
parametres = perceptron.coef_.shape[1]
print("le nombre de paramètres possède ce modèle" , {parametres})
```

```
le nombre de paramètres possède ce modèle {55}
```

notre model a 55 parametres , le nombre de paramètres est égal au nombre de coefficients de régression

on a 55 descripteurs

5. En se basant sur les poids synaptique, réalisez un ordonnancement de

l’importance des caractéristiques. Justifiez la faisabilité de l’ordonnancement des caractéristiques à partir des poids synaptiques.

In [308]:
```
car = perceptron.coef_
car
cdf=pd.DataFrame(columns = ['variable','poids'])
j = -1
for i in  perceptron.coef_:
    for a in i:
        j = j+1
        cdf.loc[j]=[df.columns[j],a]
```

In [309]:
```
cdf.sort_values(by = 'poids')
```

Out[309]:

	variable	poids
24	wf_hp	-150.460000
37	wf_direct	-103.060000
39	wf_meeting	-99.570000
43	wf_edu	-84.810000
32	wf_85	-74.280000
46	cf_comma	-64.514000
25	wf_hpl	-60.240000
30	wf_data	-57.450000
28	wf_telnet	-56.230000
45	wf_conference	-50.870000
41	wf_project	-48.870000
38	wf_cs	-47.730000
26	wf_lab	-45.180000
42	wf_re	-40.000000
34	wf_1999	-38.570000
40	wf_original	-34.150000
9	wf_mail	-32.888889
48	cf_sqbracket	-32.647000
54	capital_run_length_total	-29.414634
36	wf_pm	-28.780000
47	cf_bracket	-23.730159
27	wf_labs	-13.430000
44	wf_table	-9.570000
29	wf_857	-8.280000
21	wf_font	-7.010000
31	wf_415	-6.970000
0	wf_make	-6.730000
10	wf_receive	-5.910000
12	wf_people	-2.620000
11	wf_will	0.075949
35	wf_parts	1.800000
20	wf_your	2.676923
1	wf_address	4.050000
51	cf_hash	4.664000

	variable	poids
23	wf_money	5.690000
15	wf_free	7.250000
17	wf_email	8.140000
8	wf_order	11.790000
53	capital_run_length_longest	13.666667
13	wf_report	15.520000
2	wf_all	21.095238
18	wf_you	28.235741
49	cf_exclam	29.231481
5	wf_over	29.570000
50	cf_dollar	33.192982
3	wf_3d	38.180000
14	wf_addresses	38.850000
4	wf_our	40.641026
33	wf_technology	47.170000
19	wf_credit	52.870000
16	wf_business	83.510000
7	wf_internet	89.310000
22	wf_000	90.920000
6	wf_remove	141.210000
52	capital_run_length_average	423.124186

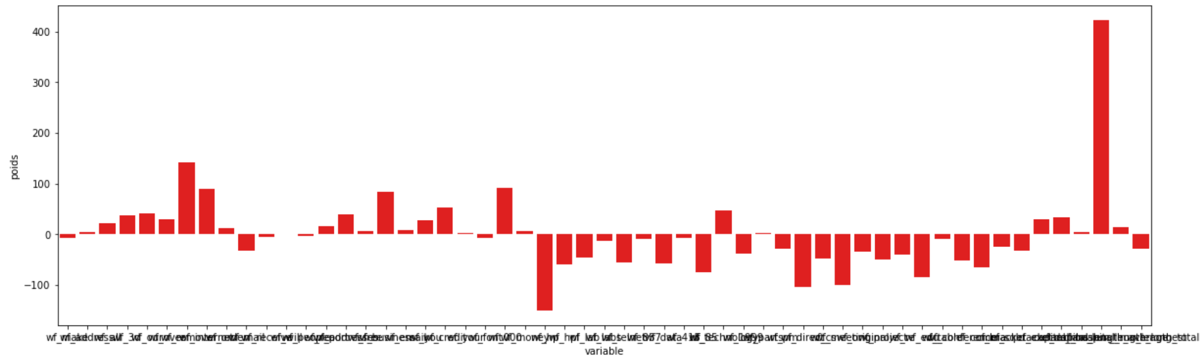
In [310]:
```
perceptron.intercept_
```

Out[310]:
```
array([-45.])
```

6. Représentez graphiquement l'importance des caractéristiques. Interprétez

les résultats. Quelles sont les caractéristiques les plus importantes (discriminantes). 2

In [311]:
```
fi = plt.subplots(figsize=(20,6))
sns.barplot(x="variable",y="poids",data=cdf ,color='red')
plt.show()
```



les caractreistique les plus importantes sont : wf_technology 47.170000 wf_credit wf_business wf_internet wf_000 wf_remove capital_run_length_average

7 . En utilisant la libraire Sklearn, développez un perceptron multicouche

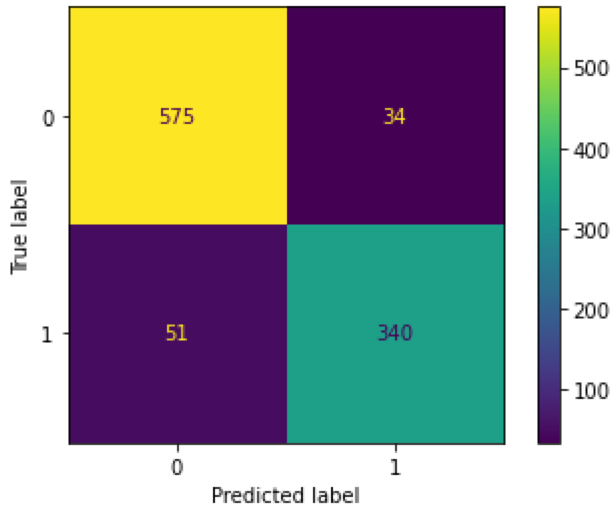
(hidden_layer_sizes=(2),activation='logistic',random_state=100 ,max_iter=1500).

In [312]:
```
import numpy as np
from sklearn.neural_network import MLPClassifier


# Creation  du modèle de perceptron multicouche
mlp = MLPClassifier(hidden_layer_sizes=(2), activation='logistic',  rand
om_state=100, max_iter=1500)

# on entraine le modèle sur les données d'entraînement
mlp.fit(X_train, y_train)

# on Évalue le modèle sur les données de test
accuracy = mlp.score(X_test, y_test)
print("Accuracy: ", accuracy)
```

```
Accuracy:  0.915
```

8. Représentez la matrice de confusion et évaluez les performances.

In [313]:
```
mat_confu =metrics.plot_confusion_matrix(mlp,X_test,y_test)
```



9 .comparez les performances du perceptron simple avec le perceptron multicouche :

In [314]:
```
print(f"le score du perceptron multicouche est   : {mlp.score(X_test,y_test)}")
print(f"le nscore du perceptron simple  est : {perceptron.score(X_test,y_test)}")
```

```
le score du perceptron multicouche est   : 0.915
le nscore du perceptron simple  est : 0.852
```

les performances :

In [315]:
```
from sklearn.metrics import classification_report
#performances perceptron simple
per= perceptron.predict(X_test)

print(classification_report(y_test,per))
```

```
              precision    recall  f1-score   support

           0       0.90      0.85      0.87       609
           1       0.78      0.86      0.82       391

    accuracy                           0.85      1000
   macro avg       0.84      0.85      0.85      1000
weighted avg       0.86      0.85      0.85      1000
```

In [316]:
```
#performances multicouches
ppper= mlp.predict(X_test)
print(classification_report(y_test,ppper))
```

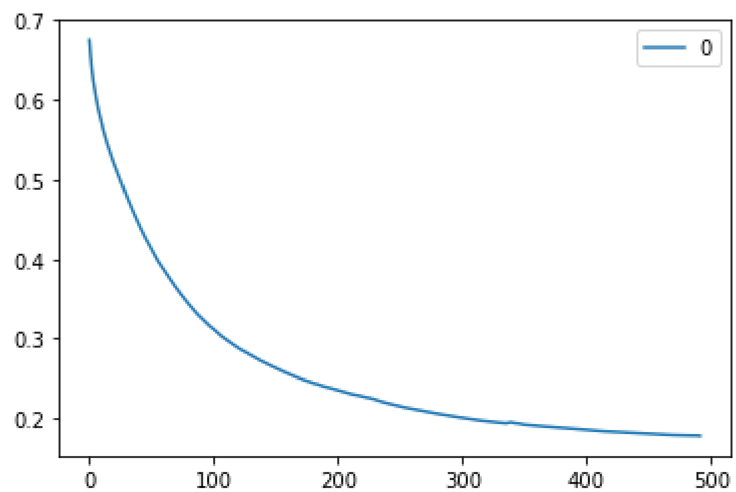
```
              precision    recall  f1-score   support

           0       0.92      0.94      0.93       609
           1       0.91      0.87      0.89       391

    accuracy                           0.92      1000
   macro avg       0.91      0.91      0.91      1000
weighted avg       0.91      0.92      0.91      1000
```

on remarque que les performance dans le perceptron multicouche est bcp plus elevee que le perceptron simple

10 .la variation de la fonction perte du perceptron multicouches en fonction du nombre díteration :

In [317]:
```
pd.DataFrame(mlp.loss_curve_).plot()
```

Out[317]: <matplotlib.axes._subplots.AxesSubplot at 0x7fba9d8175e0>



In [ ]:

In [ ]: