Étude Pratique (5) – Master Informatique – IAA – 2021 Université d'Aix-Marseille

Nous nous sommes aidés pour le code avec Anaïs Artaud, c'est pour cela qu'il est similaire. Mais nous avons bien pris le soin de faire et de comprendre chaque partie chacune.

1 - Classification à partir d'un jeu de données déséquilibré

1.1 Jeu de données artificiellement généré

Le code renvoie le rapport de classification de plusieurs classifier sur un jeu de données déséquilibré.

Le rapport renvoi comme information :

- Precision : le rapport entre les vrais positifs et la somme des vrais et faux positifs. C'est l'exactitude du classifier pour une classe.
  - *Accuracy of positive prédictions.*
  - *TP/(TP + FP)*
- Recall : mesure de la complétude du classifier, c'est-à-dire la capacité d'un classifier à trouver correctement toutes les instances positives.
  - *Fraction of positives that were correctly identified.*
  - *TP/(TP+FN)*
- f1-score : moyenne harmonique pondérée de la précision et du rappel
  - *« En règle générale, la moyenne pondérée de F1 devrait être utilisée pour comparer les modèles de classificateurs, et non la précision globale. »*
  - *2x(Recall x Precision) / (Recall + Precision)*
- Support : le nombre d'occurrences réelles de la classe dans l'ensemble de données spécifié

- macro avg : moyenne non pondérée
- weighted avg : moyenne pondérée par le support (le nombre d'instances vraies pour chaque étiquette). Cela modifie " macro " pour prendre en compte le déséquilibre des étiquettes ; cela peut donner un score F qui ne se situe pas entre la précision et le rappel.

https://www.scikit-yb.org/en/latest/api/classifier/classification_report.html

https://muthu.co/understanding-the-classification-report-in-sklearn/

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html - sklearn.metrics.precision_recall_fscore_support

Variation du paramètre Weight :

*** Rapport avec déséquilibre / weights = [0.1, 0.9]***

Majort:	precision	recall	f1-score	support
M	0.97	0.86	0.91	43
m	0.14	0.50	0.22	2
accuracy			0.84	45
macro avg	0.56	0.68	0.57	45
weighted avg	0.94	0.84	0.88	45

NBort:	precision	recall	f1-score	support
M	1.00	0.98	0.99	43
m	0.67	1.00	0.80	2
accuracy			0.98	45
macro avg	0.83	0.99	0.89	45
weighted avg	0.99	0.98	0.98	45

DTort:	precision	recall	f1-score	support
M	0.95	0.95	0.95	43
m	0.00	0.00	0.00	2
accuracy			0.91	45
macro avg	0.48	0.48	0.48	45
weighted avg	0.91	0.91	0.91	45

KPort:	precision	recall	f1-score	support
M	1.00	1.00	1.00	43
m	1.00	1.00	1.00	2
accuracy			1.00	45
macro avg	1.00	1.00	1.00	45
weighted avg	1.00	1.00	1.00	45

*** Rapport avec déséquilibre / weights = [0.2, 0.8]***

Majort:	precision	recall	f1-score	support
M	0.83	0.89	0.86	38
m	0.00	0.00	0.00	7
accuracy			0.76	45
macro avg	0.41	0.45	0.43	45
weighted avg	0.70	0.76	0.73	45

NBort:	precision	recall	f1-score	support
M	0.97	0.97	0.97	38
m	0.86	0.86	0.86	7
accuracy			0.96	45
macro avg	0.92	0.92	0.92	45
weighted avg	0.96	0.96	0.96	45

DTort:	precision	recall	f1-score	support
M	0.95	0.92	0.93	38
m	0.62	0.71	0.67	7
accuracy			0.89	45
macro avg	0.79	0.82	0.80	45
weighted avg	0.90	0.89	0.89	45

KPort:	precision	recall	f1-score	support
M	0.88	1.00	0.94	38
m	1.00	0.29	0.44	7
accuracy			0.89	45
macro avg	0.94	0.64	0.69	45
weighted avg	0.90	0.89	0.86	45

*** Rapport avec déséquilibre / weights = [0.3, 0.7]***

Majort:	precision	recall	f1-score	support
M	0.95	0.88	0.91	42
m	0.17	0.33	0.22	3
accuracy			0.84	45
macro avg	0.56	0.61	0.57	45
weighted avg	0.90	0.84	0.87	45

NBort:	precision	recall	f1-score	support
M	0.98	1.00	0.99	42
m	1.00	0.67	0.80	3
accuracy			0.98	45
macro avg	0.99	0.83	0.89	45
weighted avg	0.98	0.98	0.98	45

DTort:	precision	recall	f1-score	support
M	0.95	1.00	0.98	42
m	1.00	0.33	0.50	3
accuracy			0.96	45
macro avg	0.98	0.67	0.74	45
weighted avg	0.96	0.96	0.94	45

KPort:	precision	recall	f1-score	support
M	0.95	1.00	0.98	42
m	1.00	0.33	0.50	3
accuracy			0.96	45
macro avg	0.98	0.67	0.74	45
weighted avg	0.96	0.96	0.94	45

*** Rapport avec déséquilibre / weights = [0.5, 0.5]***

Majort:	precision	recall	f1-score	support
M	0.88	0.95	0.91	39
m	0.33	0.17	0.22	6
accuracy			0.84	45
macro avg	0.61	0.56	0.57	45
weighted avg	0.81	0.84	0.82	45

NBort:	precision	recall	f1-score	support
M	0.90	0.92	0.91	39
m	0.40	0.33	0.36	6
accuracy			0.84	45
macro avg	0.65	0.63	0.64	45
weighted avg	0.83	0.84	0.84	45

DTort:	precision	recall	f1-score	support
M	0.92	0.85	0.88	39
m	0.33	0.50	0.40	6
accuracy			0.80	45
macro avg	0.62	0.67	0.64	45
weighted avg	0.84	0.80	0.82	45

KPort:	precision	recall	f1-score	support
M	0.89	1.00	0.94	39
m	1.00	0.17	0.29	6
accuracy			0.89	45
macro avg	0.94	0.58	0.61	45
weighted avg	0.90	0.89	0.85	45

La diminution du déséquilibre permet d'observer une amélioration des résultats pour certains modèles, tandis que d'autres paraissent meilleurs lorsque le weight est déséquilibré.

1.2 Sur de vraies données

```
********** Rapport avec déséquilibre ************
Maj:               precision    recall  f1-score   support

   malignant       0.38      0.36      0.37        64
      benign       0.63      0.64      0.64       107

    accuracy                           0.54       171
   macro avg       0.50      0.50      0.50       171
weighted avg       0.53      0.54      0.54       171

NB:                precision    recall  f1-score   support

   malignant       0.95      0.84      0.89        64
      benign       0.91      0.97      0.94       107

    accuracy                           0.92       171
   macro avg       0.93      0.91      0.92       171
weighted avg       0.93      0.92      0.92       171

DT:                precision    recall  f1-score   support

   malignant       0.87      0.86      0.87        64
      benign       0.92      0.93      0.92       107

    accuracy                           0.90       171
   macro avg       0.89      0.89      0.89       171
weighted avg       0.90      0.90      0.90       171

KP:                precision    recall  f1-score   support

   malignant       0.91      0.78      0.84        64
      benign       0.88      0.95      0.91       107

    accuracy                           0.89       171
   macro avg       0.89      0.87      0.88       171
weighted avg       0.89      0.89      0.89       171
```

Méthode de Sub-samplig

Code :

```
while max(Counter(y).values()) - min(Counter(y).values()) != 0:
    i = randint(0, len(y))
    if y[i] == 1 :
        X = np.delete(X,(i),axis=0)
        y = np.delete(y,i)
print(Counter(y).values())
```

```
dict_values([212, 212])
```

Résultats :

```
********** Rapport avec déséquilibre ***********
Maj:               precision    recall  f1-score   support

   malignant       0.96      0.89      0.93        57
      benign       0.92      0.97      0.95        71

    accuracy                           0.94       128
   macro avg       0.94      0.93      0.94       128
weighted avg       0.94      0.94      0.94       128

NB:                precision    recall  f1-score   support

   malignant       0.96      0.88      0.92        57
      benign       0.91      0.97      0.94        71

    accuracy                           0.93       128
   macro avg       0.93      0.92      0.93       128
weighted avg       0.93      0.93      0.93       128

DT:                precision    recall  f1-score   support

   malignant       0.91      0.93      0.92        57
      benign       0.94      0.93      0.94        71

    accuracy                           0.93       128
   macro avg       0.93      0.93      0.93       128
weighted avg       0.93      0.93      0.93       128

KP:                precision    recall  f1-score   support

   malignant       0.96      0.88      0.92        57
      benign       0.91      0.97      0.94        71

    accuracy                           0.93       128
   macro avg       0.93      0.92      0.93       128
weighted avg       0.93      0.93      0.93       128
```

Les résultats de certains classifier paraissent meilleurs mais d'autres sont détérioré, on peut donc faire un choix dans les classifier lorsque les données sont déséquilibrées. Ainsi, ici les classifier Dummy et KNeighbors sont moins performant lors de la classification de données équilibrées pour certains entrainement (pas dans le cas précédent … chaque entrainement a des résultats différents). Nous ne pouvons pas dire si ces différences sont significatives.

Over-Sampling

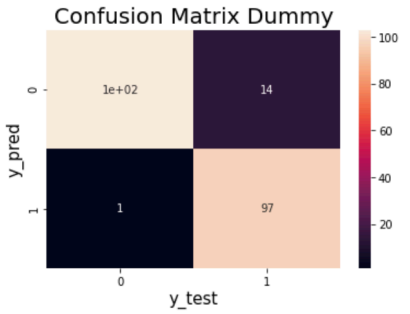
Code :

```
: while max(Counter(y).values()) != min(Counter(y).values()):
      i = randint(0, len(y)-1)
      if y[i] == 0 :
          X = np.append(X,[X[i]], axis = 0)
          y = np.append(y,[y[i]], axis = 0)

  print(Counter(y).values())
```

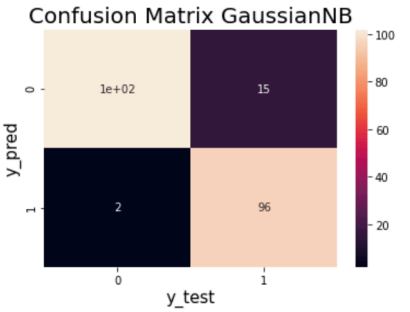
```
dict_values([357, 357])
```

Résultats :

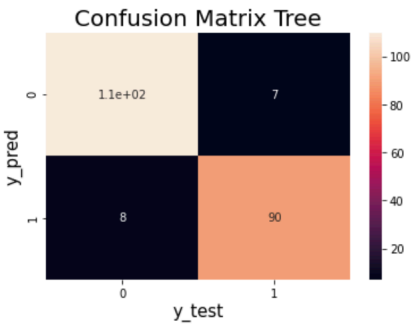
Maj: ort	precision	recall	f1-score	supp
malignant	0.99	0.88	0.93	117
benign	0.87	0.99	0.93	98
accuracy			0.93	215
macro avg	0.93	0.94	0.93	215
weighted avg	0.94	0.93	0.93	215



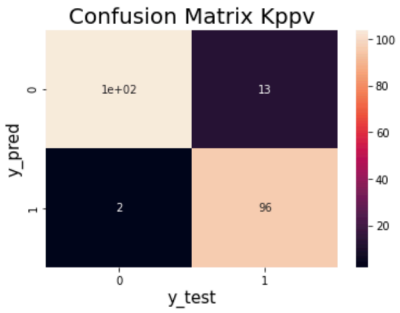
NB: ort	precision	recall	f1-score	supp
malignant	0.98	0.87	0.92	117
benign	0.86	0.98	0.92	98
accuracy			0.92	215
macro avg	0.92	0.93	0.92	215
weighted avg	0.93	0.92	0.92	215



DT: ort	precision	recall	f1-score	supp
malignant	0.93	0.94	0.94	117
benign	0.93	0.92	0.92	98
accuracy			0.93	215
macro avg	0.93	0.93	0.93	215
weighted avg	0.93	0.93	0.93	215



KP: ort	precision	recall	f1-score	supp
malignant	0.98	0.89	0.93	117
benign	0.88	0.98	0.93	98
accuracy			0.93	215
macro avg	0.93	0.93	0.93	215
weighted avg	0.94	0.93	0.93	215



2 - Challenge sur données réelles (Kaggle), pour pratiquer

Visualisation des données en partie traité :

	Survived	Pclass	Sex	Age	SibSp	Parch	Embarked
0	1	1	1	28.0	0	0	2
1	0	3	1	36.0	0	0	2
2	1	2	0	3.0	1	2	0
3	1	2	0	40.0	0	0	2
4	0	3	1	32.0	0	0	1
...	...	...	...	...	...	...	...
886	1	1	0	21.0	2	2	0
887	1	1	0	51.0	1	0	2
888	0	3	1	28.0	0	0	2
889	1	3	0	5.0	2	1	0
890	0	1	1	64.0	0	0	2

891 rows × 7 columns

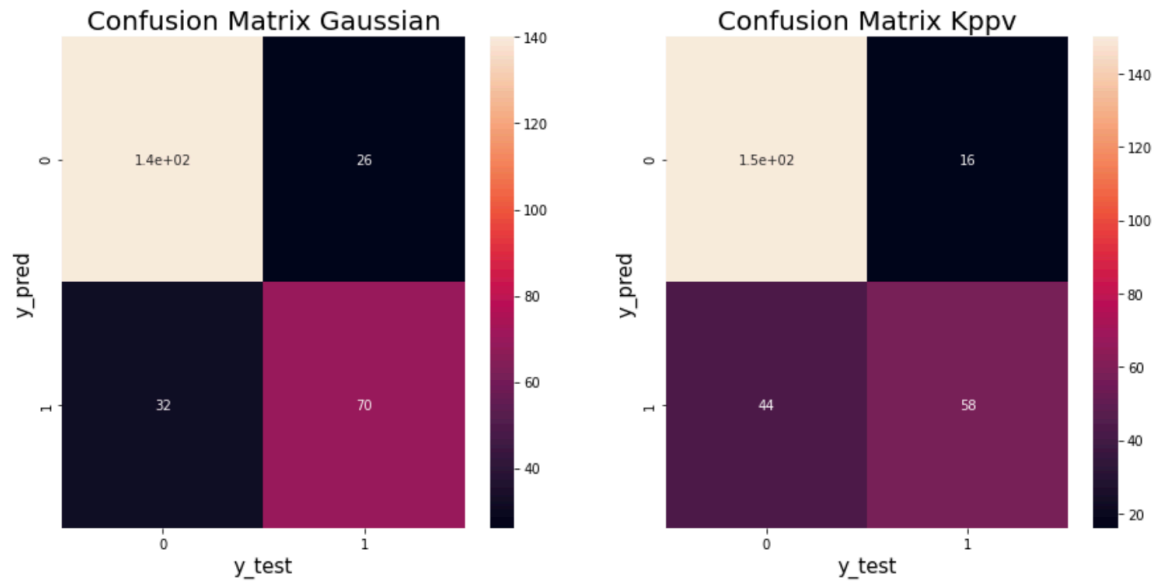
A ce stade il ne reste plus qu'à séparer les données X et y, y étant si la personne survie ou pas et X le reste des données. De plus, il manque quelques âges pour cela nous utilisons le code ci-dessous pour remplacer les valeurs manquantes par la moyenne des âges des passagers.

```
from sklearn.impute import SimpleImputer
remplir = SimpleImputer(missing_values = np.nan)
df = remplir.fit_transform(df)
```

Par default SimpleImputer a la strategy « mean ».

J'ai choisi d'utiliser comme classifier GaussianNB et KNeighbors.

```
********** Rapport avec déséquilibre ***********
NB:                precision    recall  f1-score   support

         Yes       0.81      0.84      0.83       166
          No       0.73      0.69      0.71       102

    accuracy                           0.78       268
   macro avg       0.77      0.76      0.77       268
weighted avg       0.78      0.78      0.78       268

KP:                precision    recall  f1-score   support

         Yes       0.77      0.90      0.83       166
          No       0.78      0.57      0.66       102

    accuracy                           0.78       268
   macro avg       0.78      0.74      0.75       268
weighted avg       0.78      0.78      0.77       268
```



Le classifieur GaussienNB peut paraitre meilleur que le classifieur KNeighbors, en effet la le recall est meilleur en moyenne, mais pas forcément la precision. De plus, le nombre de faux positif est moins important mais c'est KNeighbors qui le remporte que les faux négatifs à un taux plus bas.

Le test de McNemar :

```
table = [[conf_mat_nb[0][1]+conf_mat_nb[1][0] + conf_mat_kppv[0][1]+conf_mat_kppv[1][0],
         conf_mat_nb[0][0]+conf_mat_nb[1][1]+conf_mat_kppv[0][1]+conf_mat_kppv[1][0]],
         [conf_mat_nb[0][1]+conf_mat_nb[1][0]+ conf_mat_kppv[0][0]+conf_mat_kppv[1][1],
          conf_mat_nb[0][0]+conf_mat_nb[1][1] + conf_mat_kppv[0][0]+conf_mat_kppv[1][1]]]
```

```
# calculate mcnemar test
result = mcnemar(table, exact=True)
# summarize the finding
print('statistic=%.3f, p-value=%.3f' % (result.statistic, result.pvalue))
# interpret the p-value
alpha = 0.05
if result.pvalue > alpha:
    print('Non rejet de H0')
else:
    print('Rejet de H0')
```

```
statistic=266.000, p-value=0.897
Non rejet de H0
```

Les résultats ne sont pas statistiquement différents, on ne peut donc pas dire si un classifier est meilleur qu'un autre.

https://machinelearningmastery.com/mcnemars-test-for-machine-learning/

En moyennant sur 10 train_test_split j'ai obtenu :

```
statistic=11.800, p-value=1.000
Non rejet de H0
```

Nous pouvons donc confirmer que de ces deux modèles de classification aucun n'est meilleur que l'autre.