

Classification avec le jeu de données TITANIC

L'objectif est de poursuivre le TP précédente où de nombreux prétraitements ont été effectués en ingénierie des données pour pouvoir faire un modèle de prédiction des survivants ou non.

Executer les importations suivantes :

```
In [1]: 1 import pandas as pd
2 import numpy as np
3 import sklearn
4 from sklearn.naive_bayes import GaussianNB
5 from sklearn.metrics import accuracy_score
6 from sklearn.model_selection import train_test_split
7 from sklearn.model_selection import KFold
8 from sklearn.model_selection import cross_val_score
9 from sklearn.metrics import confusion_matrix
10 from sklearn.metrics import classification_report
11 import seaborn as sns
12 import matplotlib.pyplot as plt
13 from sklearn.metrics import precision_recall_fscore_support as s
14 from sklearn.linear_model import LogisticRegression
15 from sklearn.tree import DecisionTreeClassifier
16 from sklearn.neighbors import KNeighborsClassifier
17 from sklearn.naive_bayes import GaussianNB
18 from sklearn.svm import SVC
19 from sklearn.model_selection import GridSearchCV
20 from sklearn.ensemble import RandomForestClassifier
21
22 #Sickit learn met régulièrement à jour des versions et indique d
23 #ces deux lignes permettent de ne pas les afficher.
24 import warnings
25 warnings.filterwarnings("ignore", category=FutureWarning)
```

Si vous n'avez pas terminé le TP précédent vous pouvez récupérer le fichier titanic2.csv qui contient les différentes transformations. Le mettre dans le répertoire Dataset.

```
In [2]: 1 df=pd.read_csv('Dataset/titanic2.csv', sep=';')
        2 df.head()
```

Out[2]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	4	1	0	0	2
1	1	1	0	0	1	0	3	0
2	1	3	0	6	0	0	0	2
3	1	1	0	6	1	0	3	2
4	0	3	1	6	0	0	1	2

Il faut maintenant définir les variables d'apprentissage et la variable à prédire. Ici Survived est la variable à prédire. Attention elle est positionné à la première colonne.

```
In [3]: 1 array = df.values
        2 X = array[:,1:8]
        3 y = array[:,0]
        4
```

Découper le jeu de données en jeu de test et jeu d'apprentissage. Prenez 30% du jeu de données pour le test.

In [4]:

```
1
2
3
4
5
6
7
8
```

Essai d'un classifieur

Avec le classifieur GaussianNB effectuer une première prédiction en donnant la valeur de l'accuracy.

In [5]:

```
1
2
3
4
5
6
7
```

```
accuracy : 0.7155963302752294
```

Afficher la matrice de confusion et le classification report.

In [6]:

```
1
2
3
4
5
```

```
matrice de confusion
[[57 17]
 [14 21]]
```

	precision	recall	f1-score	support
0	0.80	0.77	0.79	74
1	0.55	0.60	0.58	35
micro avg	0.72	0.72	0.72	109
macro avg	0.68	0.69	0.68	109
weighted avg	0.72	0.72	0.72	109

Effectuer une cross validation avec 10 splits (Kfold)

In [7]:

```
1 seed=7
2 k_fold = KFold(n_splits=10, shuffle=True, random_state=seed)
```

Appliquer le classifieur GaussianNB et donner les différentes accuracy pour les 10 évaluations.

In [8]:

```
1
2
3
4
5
6
7
8
```

Les différentes accuracy pour les 10 évaluations sont :

```
[0.8125      0.8125      0.75      0.75      0.875      0.86666667
 0.73333333 0.8        0.73333333 0.73333333]
```

Accuracy moyenne : 0.7866666666666667 standard deviation 0.051901348730066764

Essai de plusieurs classifieurs

Utiliser à présent différents classifieurs : KNeighborsClassifier, DecisionTreeClassifier, GaussianNB, SVC et RandomForestClassifier.

In [9]:

```
1
2
3
4
5
6
7
8
9
10
11
12
13
```

Quels sont les résultats des classifications ? Quel est le classifieur le plus efficace ?

In [10]:

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12
```

```
KNN: 0.647083 (0.109117)  
CART: 0.735417 (0.044576)  
NB: 0.768750 (0.072821)  
SVM: 0.730000 (0.071317)  
RFO: 0.767917 (0.079649)
```

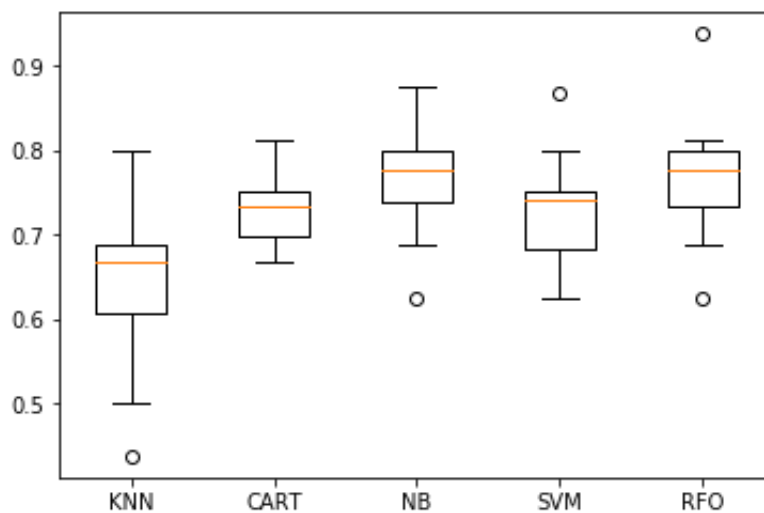
A l'aide de boxplot afficher les résultats des différents classifieurs.

In [11]:

```
1
2
3
4
5
6
7
8
9
10
11
```

```
Out[11]: [Text(0, 0, 'KNN'),
Text(0, 0, 'CART'),
Text(0, 0, 'NB'),
Text(0, 0, 'SVM'),
Text(0, 0, 'RFO')]
```

Comparaison des algorithmes



Le classifieur RandomForestClassifier donne de meilleurs résultats. A l'aide de GridSearch, évaluer les différents paramètres :

```
grid_param = {'n_estimators': [4, 6, 9],
'max_features': ['log2', 'sqrt', 'auto'],
'criterion': ['entropy', 'gini'],
'max_depth': [2, 3, 5, 10],
'min_samples_split': [2, 3, 5],
'min_samples_leaf': [1, 5, 8]
}
```

```
In [12]: 1  grid_param = {'n_estimators': [4, 6, 9],
2              'max_features': ['log2', 'sqrt', 'auto'],
3              'criterion': ['entropy', 'gini'],
4              'max_depth': [2, 3, 5, 10],
5              'min_samples_split': [2, 3, 5],
6              'min_samples_leaf': [1, 5, 8]
7              }
8
9
10
11
12
13
14
```

Quelle est la meilleure accuracy et pour quels paramètres ?

```
In [13]:
```

```
1
2
3
4
5
6
7
8
9
```

```
meilleur score 0.7608695652173914
```

```
meilleurs paramètres {'criterion': 'entropy', 'max_depth': 3, 'max_f
eatures': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 2, 'n_
estimators': 9}
```

```
meilleur estimateur RandomForestClassifier(bootstrap=True, class_wi
ght=None, criterion='entropy',
      max_depth=3, max_features='sqrt', max_leaf_nodes=None,
      min_impurity_decrease=0.0, min_impurity_split=None,
      min_samples_leaf=5, min_samples_split=2,
      min_weight_fraction_leaf=0.0, n_estimators=9, n_jobs=Non
e,
      oob_score=False, random_state=None, verbose=0,
      warm_start=False)
```

GaussianNB obtient un assez bon score mais il n'a pas d'hyperparamètres à rechercher. Il est suivi par DecisionTreeClassifier et SVC.

Appliquer GridSearchCV pour ces deux classifieurs.

In [14]:

```
1
2
3   grid_param = {
4       'max_depth': [1,2,3,4,5,6,7,8,9,10],
5       'criterion': ['gini', 'entropy'],
6       'min_samples_leaf': [1,2,3,4,5,6,7,8,9,10]
7   }
8
9
10
11
12
13
14
15
```

meilleur score 0.782608695652174

meilleurs paramètres {'criterion': 'gini', 'max_depth': 1, 'min_samples_leaf': 1}

meilleur estimateur DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=1, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort=False, random_state=None, splitter='best')

In [15]:

```

1  ▼ grid_param = {
2      'C': [0.001, 0.01, 0.1, 1, 10],
3      'gamma': [0.001, 0.01, 0.1, 1],
4      'kernel': ['linear', 'rbf']}
5
6
7
8
9
10
11
12
13
14
15

```

meilleur score 0.7391304347826086

meilleurs paramètres {'C': 1, 'gamma': 0.001, 'kernel': 'linear'}

meilleur estimateur SVC(C=1, cache_size=200, class_weight=None, coef_0=0.0,
 decision_function_shape='ovr', degree=3, gamma=0.001, kernel='linear',
 max_iter=-1, probability=False, random_state=None, shrinking=True,
 tol=0.001, verbose=False)

Vous constatez que RandomForestClassifier et DecisionTreeClassifier peuvent avoir la même accuracy.

Faire un gridsearch avec les deux classifieurs en prenant les paramètres précédents.

In [16]:

```

1  ▼ classifiers = {
2      'RandomForestClassifier': RandomForestClassifier(),
3      'DecisionTreeClassifier': DecisionTreeClassifier()
4  }
5
6  ▼ params = {'RandomForestClassifier': [{'n_estimators': [4, 6, 9]
7      'max_features': ['log2', 'sqrt', 'auto']},
8      'criterion': ['entropy', 'gini']},
9      'max_depth': [2, 3, 5, 10]},
10     'min_samples_split': [2, 3, 5]},
11     'min_samples_leaf': [1, 5, 8]}],
12     'DecisionTreeClassifier': [{'max_depth': [1, 2, 3, 4, 5, 6
13     'criterion': ['gini', 'entropy']},
14     'min_samples_leaf': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]}]}
15
16

```

In [17]:

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14

```

Le meilleur resultat :

```

Classifier : DecisionTreeClassifier score 0.7826 avec DecisionTr
eeClassifier(class_weight=None, criterion='gini', max_depth=1,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, presort=False, random_stat
e=None,
              splitter='best')

```

Tous les résultats :

```

Classifier : DecisionTreeClassifier score 0.7826 avec DecisionTr
eeClassifier(class_weight=None, criterion='gini', max_depth=1,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, presort=False, random_stat
e=None,
              splitter='best')

```

```

Classifier : RandomForestClassifier score 0.6957 avec RandomFore
stClassifier(bootstrap=True, class_weight=None, criterion='gini',
              max_depth=None, max_features='auto', max_leaf_nodes=None
,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=No
ne,
              oob_score=False, random_state=None, verbose=0,
              warm_start=False)

```

Conclusion : il faut toujours faire attention aux paramètres passés dans un classifieur. Ils peuvent tout changer !

Sauvegarde du modèle

Sauvegarder le meilleur modèle appris et recharger le pour le tester avec `y_test`.

```
In [18]: 1 clf=gd_sr.best_estimator_
2 import pickle
3 filename = 'pkl_Titanicbestmodel.sav'
4 pickle.dump(clf, open(filename, 'wb'))
```

```
In [19]: 1
2
3
4
5
6
7
8
9
10
```

```
Modèle chargé DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=1,
max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, presort=False, random_state=None,
splitter='best')
```

accuracy:

0.8165137614678899

matrice de confusion

```
[[62 12]
 [ 8 27]]
```

	precision	recall	f1-score	support
0	0.89	0.84	0.86	74
1	0.69	0.77	0.73	35
micro avg	0.82	0.82	0.82	109
macro avg	0.79	0.80	0.80	109
weighted avg	0.82	0.82	0.82	109

