SORBONNE UNIVERSITÉ

PROGRES - MINI-PROJET 2

Classification of connected objects

Directed by:
Nassim Bouchama
Yacine Fodil

Projectgoal:

The goal of Mini-Project 2 is to identify connected objects based on supervised classification. We will use attributes extracted from the traffic of objects. First, we will focus on a first set of attributes characterizing network flows, then we will deal with a second set of textual attributes represented by a binary matrix.

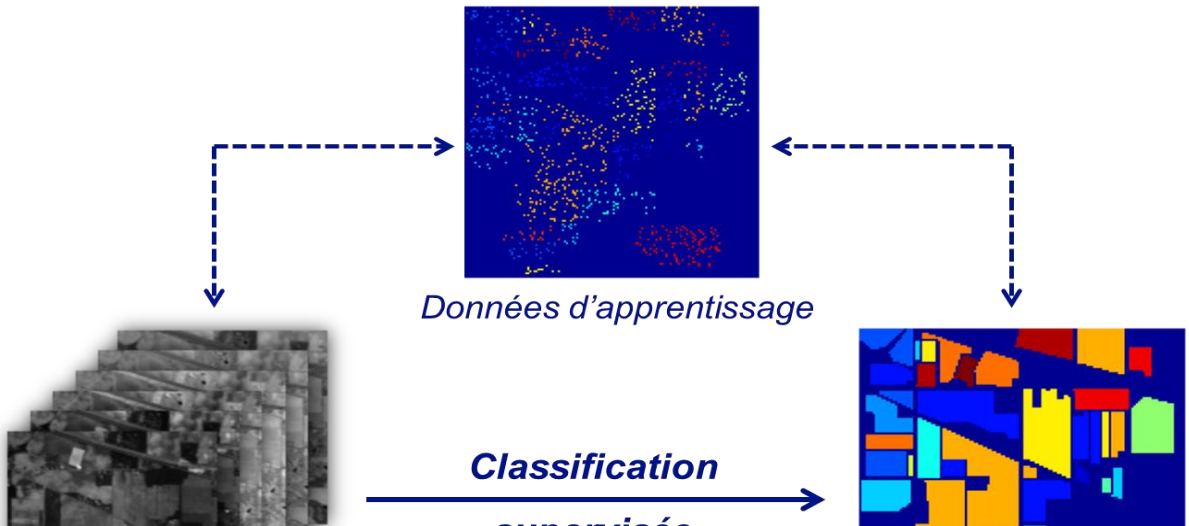
Definitions:

Supervised classification:

The main objective of supervised classification is to define rules for classifying objects in classes based on qualitative or quantitative variables characterizing these objects. Methods often extend to quantitative Y variables (regression).

At the outset, a so-called learning sample is available, the classification of which is known. This sample is used to learn the grading rules.

It is necessary to study the reliability of these rules to compare and apply them, to assess cases of under-learning or over-learning (complexity of the model). A second independent sample, known as validation or testing, is often used.



The naïve Bayesian classification:

Itis a simple probabilistic Bayesian classification based on the Bayes theorem with a strong (so-called naïve) independence of assumptions. It implements a naïve Bayesian classifier, or Bayes naïve classifier, belonging to the linear classifier family. Chaque classified characteristic is independent of the value of any other feature.

K-NN (K-nearestneighbors) :

Est a supervised learning method. It can be used for both regression and classification. Its functioning can be likened to the following analogy "tell me who your neighbors are, I'll tell you who you are...".

To make a prediction, the K-NN algorithm will not calculate a predictive model from a TrainingSet, it does not need to build a predictive model. Thus, for K-NN there is no learning phase per se. That's why it is sometimes categorized in Lazy Learning. To make a prediction, K-NN relies on the dataset to produce a result.

Learning by decision tree:

Denotes a method based on the use of a decision tree as a predictive model. It is used in data searches and machine learning.

In these tree structures, the leaves represent the values of the target variable and the branch lines correspond to combinations of input variables that lead to these values. In decision analysis, a decision tree can be used to explicitly represent the decisions made and the processes that lead them. In learning and data search, a decision tree describes the data but not the decisions themselves, the tree would be used as a starting point for the decision-making process.

Part1:

First of all, we import the modules and bookstores necessary for the smooth running of our project:

```
import pandas as pd
import os
import matplotlib.pyplotas  plt
fromsklearn.model_selectionimport train_test_split
fromsklearn.treeimport DecisionTreeClassifier
fromsklearn.metricsimport accuracy_score
fromsklearn.metricsimport confusion_matrix
fromsklearn.metricsimport precision_score,recall_score
fromsklearn.model_selectionimport cross_val_score
fromsklearn.naive_bayesimport GaussianNB
fromsklearn.neighborsimport KNeighborsClassifier
```

Then we go to the reading of the data via Pandas

```
P = "/Users/GL72/Desktop/progresv2"
defdatasetFlows(path=P):csv_path  = os.path.join(P,"datasetFlows.csv")
return pd.read_csv(csv_path)data=datasetFlows()  #Lecture des données
```

In addition, comes the preparation of data and the organization of attributes

The label table

```
Y = data["type"]
 0            Aria
 1            Aria
 2            Aria
 3            Aria
 4            Aria
            ...
```

All attributes without labels

```
X = data.drop(labels="label", axis=1).drop(labels="type", axis=1)
```

	size	average	min	max	iat	UDP	...	TCP	DNS	DHCP	SSDP	ICMP	TFTP
0	17	141	54	674	0.029050	0	...	0	0	0	0	0	0
1	2	113	74	153	0.000040	0	...	0	1	0	0	0	0
2	4	342	342	342	0.502240	0	...	0	0	1	0	0	0
3	4	342	342	342	0.522690	0	...	0	0	1	0	0	0
4	13	126	54	779	0.023449	0	...	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
18035	8	125	54	431	0.017070	0	...	0	0	0	0	0	0
18036	3	342	342	342	0.972008	0	...	0	0	1	0	0	0
18037	2	88	80	96	0.000044	0	...	0	1	0	0	0	0
18038	3	342	342	342	0.038818	0	...	0	0	1	0	0	0
18039	42	242	54	662	0.063041	0	...	0	0	0	0	0	0

The data is therefore separated between a training set, on which one learns the model, and a test game, on which it is evaluated (using70% of the data forlearning)

```
X_train,X_test,Y_train, Y_test- train_test_split(X,Y, test_size-0.3,random_state-10)
```

Implementation of the Naive-Bayes algorithm:

The Naive-Bayes algorithm is used

```
GaussNb = GaussianNB()
```

We train him on the learning set

```
GaussNb.fit(X_train, Y_train)
```

Predictions are made about test and learning sets

```
pred1= GaussNb.predict(X_test) pred12=GaussNb.predict(X_train)
```

The forecast error is estimated via accuracy_score

```
test_score00=accuracy_score(Y_test, pred1, normalize= True)train_score01=accuracy_score(Y_train
,pred12)
```

Then we calculate theaccuracy and recall scores

```
pre1=precision_score(Y_test,pred1,average=None).mean() rec1=recall_score(Y_test
,pred1,average=None).mean() pre12=precision_score(Y_train
,pred12,average=None).mean()rec12=recall_score(Y_train
,pred12,average=None).mean()
```

The results are:

```
**********Naive_Bayes algorithme**********

 taux de reconnaissance pour la base de test : 0.4569475240206948

**************************
 taux de reconnaissance pour la base d'apprentissage : 0.456921127652835

**************************
score de précision pour la base de test : 0.1926988643783844

score de précision pour la base d'appentissage : 0.20352903932658567

**************************
score de rappel pour la base de test:  0.2586916878681637

score de rappel pour la base d'apprentissage:  0.2748684324896952
```

We recall the accuracy,which represents the proportion of positive predictions among the values being predicted positive,and the recall, which represents the positive predictions among all the positive values in the test set.

We notice that the recognition rate on the basis of learning is equal to that of the test base, which shows thatwe have experienced learning.

Themodel of NaiveBayes remains underperforming with respect to the recognition rate,which isquite low (up0.45).

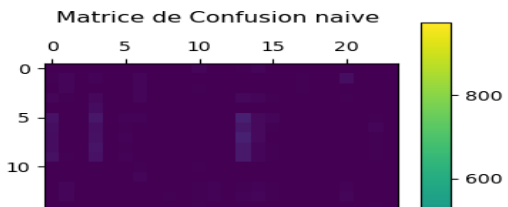
The same goes for accuracy and recall score, which do not increase after implementation of the algorithme,indicating that we have real positives.

Construction of the confusion matrix:

The confusion matrix is a matrix that measures the quality of a classification system. Thediagonal represents the number of points for which predictions turn out tobe goodand equalto thetruex labels, while non-diagonal elementsrepresentthe items for which the classifier was mistaken. One of the interests of the confusion matrix is that it quickly shows whether a classification system manages to classify correctly.

```
table(Y_test,
pred1)plt.matshow (table)plt.title("Matrix  of NaiveConfusion")plt.colorbar()plt.show()

matrice_test=confusion_matrix(Y_test,pred1) matrice_train=confusion_matrix(Y_train
,pred12)
```



```
matrice de confusion (test):
 [[ 0  0  0 ...  0  1  0]
 [ 5  0 16 ...  0  0  0]
 [ 0  0 15 ...  0  0  0]
 ...
 [ 1  0  0 ...  7 13  0]
 [ 0  0  0 ...  0 11  0]
 [ 0  0  2 ...  0  0  0]]
```

```
matrice de confusion (appentissage):
 [[ 2  0  0 ...  0  1  0]
 [ 7  0 36 ...  0  0  0]
 [ 0  0 40 ...  0  0  0]
 ...
 [ 2  0  0 ... 13 19  0]
 [ 0  0  0 ...  0 29  0]
 [ 0  0  2 ...  0  0  0]]
```

On sees that the diagonal of the matrix is not very conclusive and that there are many points hors diagonal, indicating that the classifier miscon predicted some data.

Implementation of the Kneighborsalgorithm:

The Kneighbors algorithm is used, with No. 5

```
Kneigh = KNeighborsClassifier(n_neighbors=5)
```

The same steps are repeated as in the previous case, namely training, predictions, estimating the forecast error and calculating accuracy and recall scores:

```
Kneigh.fit(X_train, Y_train)
pred21 = Kneigh.predict(X_test) test_score03 =accuracy_score(Y_test
,pred21)
pre21=precision_score(Y_test,pred21,average=None).mean()rec21=recall_score(Y_test
,pred21,average=None).mean()pred22 = Kneigh.predict(X_train)train_score03 =accuracy_score(Y_train
,pred22) pre22=precision_score(Y_train
,pred22,average=None).mean()rec22=recall_score(Y_train
,pred22,average=None).mean()
```

The results are:

```
taux de reconnaissance pour le set de test:
 0.7797487065779749

taux de reconnaissance pour le set d'appentrissage:  0.8457396262274312

**************************

score de précision pour le set de test
: 0.6246418928942477

score de précision pour le set d'apprentissage : 0.7589707400696197

**************************

score de rappel pour le set de test: 0.5985064613794903

score de rappel pour le set d'apprentissage:  0.7178265611849549
```

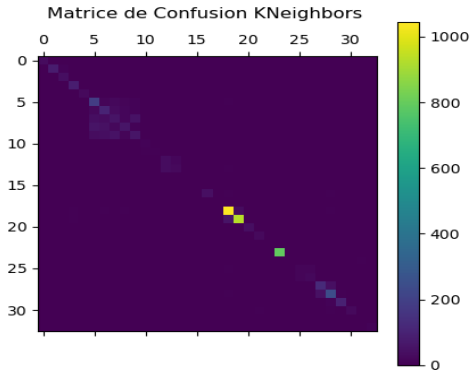
We see that the recognition rate on the basis of learning (0.84) is greater than the rate obtained on the basis of test (0.77), the difference between them is not very large, indicating that the mode did not over-learn.

One also notices that the accuracy and recall score on the learning basis is larger than it on the basis of test.

Thus the accuracy value is almostequal to the recallvalue, which indicates that there is also real positis, while having some false predictions negligible.

Construction of the confusion matrix:

```
table(Y_test),pred21)plt.matshow (table)plt.title
("Confusion Matrix KNeighbors"n")plt.colorbar()plt.show()matrice_test-confusion_matrix(Y_test,
atrix (Y_train,
```




```
matrice de confusion (test):
 [[30  0  0 ...  1  0  0]
 [ 0 79 10 ...  0  0  0]
 [ 0  0 38 ...  0  0  0]
 ...
 [ 0  0  0 ... 25  0  0]
 [ 0  0  0 ...  0  3  0]
 [ 0  0  0 ...  1  0 10]]
```

```
matrice de confusion (appentissage):
 [[ 66   0   0 ...   0   0   0]
 [  0 211   9 ...   0   0   0]
 [  0   3  92 ...   0   0   0]
 ...
 [  0   0   0 ...  45   0   0]
 [  0   0   0 ...   0  19   0]
 [  0   0   0 ...   0   0  29]]
```

Thediagonal matrix is veryclear, even if there are some elements outside, as said earlier, the difference in the recognition rate between the learning set and the test set is not very large, we can see it well by the off-diagonal elements

Implementation of the DecisionTree algorithm:

We follow the same pattern:

```
dtree = DecisionTreeClassifier() dtree.fit(X_train
, Y_train) X_test_pred=dtree.predict(X_test)
X_train_pred=dtree.predict(X_train)test_score=accuracy_score(Y_test
,X_test_pred) train_score=accuracy_score(Y_train
,X_train_pred) pre3=precision_score(Y_test
,X_test_pred,average=None).mean()rec3=recall_score(Y_test
,X_test_pred,average=None).mean()pre4=precision_score(Y_train
```

```
,X_train_pred,average=None).mean()rec4=recall_score(Y_train
,X_train_pred,average=None).mean()
```

The results are:

```
accuracy_score of the test set: 0.7812269031781227

**************************

accuracy_score of the train set : 0.9762432689261957

**************************

precison score of the test set : 0.6286316901917346

precison score of the train set : 0.9720144287807753

**************************

recall score of the test set: 0.6161652764364989

recall score of the train set : 0.9659619791053712
```

It is han
tha ng a
slig rn
and which are sensitive to the slightest variation, such a model will perform then perform on the training game but will be less good on new data.

Theaccuracy and recall score increases every two years after implementation of the DecisionTree, for a score reaching almost 100%, indicating that the predictions are right, or even a little too much. Indeed, this model does not generalize as well on new data.

To use all of our data totrain, to test and to avoid a potential bias related to making a single evaluation, we willgothrough a cross-validation:

```
Scores - cross_val_score (dtree, X_train, Y_train, scoring-None, CV-4)
print('cross val scores are:',Scores)print
("average scores obtained by Cross_vall is",Scores.mean())
```

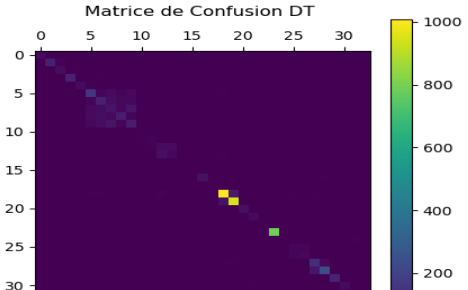
```
cross val scores are : [0.78333861 0.7804878  0.76750079 0.78808996]
**************************

la moyennes des scores obtenus par Cross_vall est  0.779854292049414
**************************
```

Construction of the confusion matrix:

```
table(Y_test)X_test_pred)plt.matshow (table)plt.title
("Confusion Matrix DT")plt.colorbar()plt.show()matrice_test-confusion_matrix (Y_test

X_test_pred)matrice_train-confusion_matrix (Y_train
,X_train_pred)
```




```
matrice de confusion (test):
 [[29  0  0 ...  0  0  0]
 [ 0 83  8 ...  0  0  0]
 [ 0  0 35 ...  0  0  0]
 ...
 [ 0  0  0 ... 30  1  0]
 [ 0  0  0 ...  0  4  0]
```

```
matrice de confusion (appentissage):
 [[ 69   0   0 ...   0   0   0]
 [  0 221   5 ...   0   0   0]
 [  0   0  96 ...   0   0   0]
 ...
 [  0   0   0 ...  58   0   0]
 [  0   0   0 ...   0  28   0]
```

Thediagonal a of the confusion matrix is clearyou,there are not many elementsoutside, which indicates that the classifier has well knew the predictions.

The modelis reconstred by varying the hyperparameters of the DT model:

This is done in varying the maximum depthof the tree

```
i=0
for  in    range(4):i=i+1
dtree = DecisionTreeClassifier(max_depth=i) dtree.fit(X_train
,  Y_train) X_test_pred=dtree.predict(dtree.predict(dtree.)
X_test)X_train_pred=dtree.predict(X_train)test_score=accuracy_score(Y_test,X_test_pred)train_score=accurac
y_score(Y_train,X_train_pred)
```

So we get:

```
taux de reconnaissance du set de test pour une pronfondeur max de ( 1 ) est : 0.348669623059867

taux de reconnaissance du set d'apprentissage pour une profondeur max de ( 1 ) est : 0.3469274627811213

*************************

taux de reconnaissance du set de test pour une pronfondeur max de ( 2 ) est : 0.42886178861788615

taux de reconnaissance du set d'apprentissage pour une profondeur max de ( 2 ) est : 0.4292841305036427

*************************

taux de reconnaissance du set de test pour une pronfondeur max de ( 3 ) est : 0.5269770879526977

taux de reconnaissance du set d'apprentissage pour une profondeur max de ( 3 ) est : 0.5367437440608173

*************************

taux de reconnaissance du set de test pour une pronfondeur max de ( 4 ) est : 0.540650406504065

taux de reconnaissance du set d'apprentissage pour une profondeur max de ( 4 ) est : 0.5493348115299335
```

There isa linear relationship between the two parameters, the greater the depth of the maximum tree, the higher the recognition rate. This shows that the model learns, but when the depth reaches too much value, the model will begin to over-learn.

Creating a model with new subsets

The model is trained only on subsets, which are taken directly according to the csv file

```
model_bis=data[list(data.columns[:ss_ensemble])]
```

Separation of data between a test set and a learning set

```
X2_train,X2_test,Y2_train, Y2_test- train_test_split(model_bis,Y,  test_size-0.3)
```

We use the DecisionTree model and adapt it for the learning set

```
dtree = DecisionTreeClassifier() dtree.fit(X2_train
, Y2_train)
```

Then we make the prediction and calculation of the accuracy_score

```
predict_test_set=dtree.predict(X2_test)
predict_train_set=dtree.predict(X2_train)test_score2=accuracy_score(Y2_test,predict_test_set)
train_score2=accuracy_score(Y2_train,predict_train_set)
```

Weget, for 4 subsets for example: Size, Average, Min and Max

```
le taux de reconnaissance sur l'ensemble de test du nouveau modele
: 0.7594235033259423

le taux de reconnaissance sur l'ensemble d'apprentissage du nouveau modele
: 0.8002850807728856
```

En comparing the recognition rates between the test game and the learning game, we notice that the model is successful in learning, and thus making good predictions.

Training only on DHCP streams of learning data:

A model trained therefore only on DHCP:

```
Re=data[data['DHCP']==1]
```

Added labels:

```
Y_dhcp = Re["type"] X_dhcp = Re[
list(data.columns[:16])]
```

Separation of data between a test set and a learning set

```
X_dhcp_train,X_dhcp_test,Y_dhcp_train, Y_dhcp_test = train_test_split(X_dhcp,Y_dhcp, test_size=0.3)
```

The DecisionTree model is used and adapted for the learning set

```
dtree = DecisionTreeClassifier()dtree.fit(X_dhcp_train, Y_dhcp_train)
```

Make predictions about test data and calculating the mode recognition rate to the DHCP test data set

```
pred_dhcp=dtree.predict(X_dhcp_test) test_score3=accuracy_score(Y_dhcp_test
,pred_dhcp)
```

Make predictions about learning data and calculating the rate of recognition of the model to the DHCP train data set

```
pred_dhcp2=dtree.predict(X_dhcp_train) train_score3=accuracy_score(Y_dhcp_train
,pred_dhcp2
```

We finally get:

```
taux de reconnaissance sur base de test pour l'attribut DHCP uniquement  :  0.5405405405405406
taux de reconnaissance sur base d'appentissage pour l'attribut DHCP uniquement  :  0.9817578772802653
```

Takingonly the DHCP attribute as a study model and calculating the recognition rate on this newmodel,we notice that the rate on the learning basis is much higher than that on the test basis, indicating that thesister-sister-learnsmodele because the data in thismodel are simple.

Part 2:

We will now deal with a second set of textual attributes represented by a binary matrix,to be done wewillc omine the textual attributes to digital attributes and redo the study:

Reading digital and textual attributes file data via Pandas

```
defdataset(path=P):csv_path = os.path.join(P,"dataset.csv")
return pd.read_csv(csv_path)data_merge=dataset()
```

Added labels and separating data in 2 sets

```
Y = data_merge["type"] X = data_merge.drop(
labels="type", axis=1) X_train
,X_test,Y_train, Y_test = train_test_split(X,Y, test_size=0.3)
```

The DecisionTree model is used and adapted for the learning set

```
dtree = DecisionTreeClassifier() dtree.fit(X_train
,Y_train
```

Predictions, estimating forecast error, and calculating accuracy and recall scores:

```
X_test_pred=dtree.predict(X_test) X_train_pred=dtree.predict(X_train) test_score=accuracy_score(Y_test


,X_test_pred) train_score=accuracy_score(Y_train
,X_train_pred)pre5=precision_score(Y_test

,X_test_pred,average=None).mean()pre6=precision_score(Y_train,X_train_pred,average=None).mean()rec5=re
call_score(Y_test
,X_test_pred,average=None).mean()rec6=recall_score(Y_train,X_train_pred,average=None).mean()
```

We get:

```
accuracy_score du test set: 0.9848484848484849

accuracy_score du train set : 0.9996832435856826

precison score du test set : 0.9528871005607223

precison score du train set : 0.9993339993339994

recall score du test set: 0.9486062261836419

recall score du train set  : 0.9992993519005079
```

We have good predictions for test data,where we don't have under-learning.

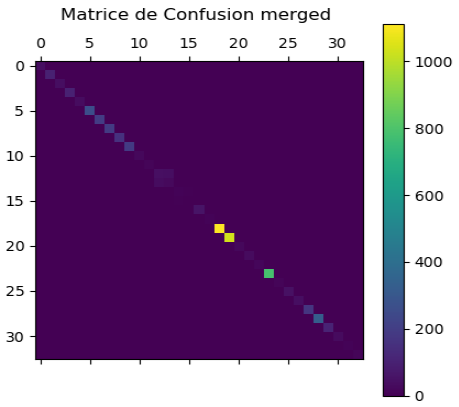
We have very good performance on the training game and for the test data because the recognition rate of the learning base "train score - 0.99" and that of the test base "test score 0.98", so we have no over-learning.

The accuracy and recallscore are almost identical, either on the learning basis or on the test basis.

In additiontothe accuracy and recall values of the models obtained on the numerical attributes of the flows and on the combined attributes, we notice that there is an improvement in the accuracy and recall values of the results obtained from the prediction on the test data, so we can say that the performance of the model obtained improves by combining the two bases.

Construction of the confusion matrix:

```
table(Y_test),
X_test_pred)plt.matshow (table)plt.title("Confusion Matrix merged")plt.colorbar()plt.show()matrice_test-
confusion_matrix(Y_test,X_test_pred)matrice_train-confusion_matrix (Y_train
,X_train_pred)
```




```
matrice de confusion (test):
 [[ 28   0   0 ...   0   0   0]
 [  0 116   0 ...   0   0   0]
 [  0   0  36 ...   0   0   0]
 ...
 [  0   0   0 ...  30   0   0]
 [  0   0   0 ...   0  11   0]
 [  0   0   0 ...   0   0  23]]
```

```
matrice de confusion (appentissage):
 [[ 72   0   0 ...   0   0   0]
 [  0 203   0 ...   0   0   0]
 [  0   0  99 ...   0   0   0]
 ...
 [  0   0   0 ...  82   0   0]
 [  0   0   0 ...   0  29   0]
 [  0   0   0 ...   0   0  40]]
```

Unsurprisingly, the matrix of confusion is clear and clear, with no elements outside, good predictions in short.

Conclusion:

During this project, connected objects were identified based on supervised classification, using two sets of differentattributes.

Through differentclassification methods and by comparing them, the DecisionTree model was ultimately selected as the best performing model, based on its evaluation via performance metrics: accuracy, recall and the matrix of confusion on learning and test data.