Collège de Bois de Boulogne

420-BD8-BB
Fouille de données

TP Proposition I – Été 2018
Apprendisage Supervise

Alejandra Jimenez Nieto (1695997)

Travail présenté à: Hafed Betenfita
Nesrine Zemirli

29 août 2018

1. Identifier les étudiants qui ont besoin d'un plan d'aide à la réussite. Est-ce que l'on est dans un cas de classification ? Expliquer.
   R/
   Oui, il s'agit d'un cas de classification: nous connaissons la valeur d'une variable ou d'un descripteur X (par exemple, "Studytime", " Paid" ou " Higher ") et on considère la sortie ou prédiction Y (étudiant a réussi o pas ses études). Un point sera classé en fonction de la classification des autres points connus.

2. Exploration des données :
   a. déterminer les statistiques suivantes : nombre d'étudiants, nombre d'étudiants qui ont réussi, nombre d'étudiants en échec, taux de graduation, autres statistiques que vous pensez être intéressantes pour ce projet
      R/
      Nombre d'étudiants : 397
      Nombre d'étudiants qui ont réussi : 265
      Nombre d'étudiants en échec : 130
      Taux de graduation : 265/397=74.3%

3. Préparation des données :
   a. Identifier les features/prédicteurs et la colonne cible
      R/
      On va choisir les prédicteurs suivants:
      - Higher : Veut suivre des études supérieures (binaire: yes or no)
      - Reason : Raison de choisir cette école (nominal: close to "home", school "reputation", "course" preference or "other")
      - Guardian: Tuteur de l'étudiant (nominal: "mother", "father" or "other")

- Studytime: Temps d'étude hebdomadaire (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
- Paid: Cours supplémentaires payés dans le cours de la matière (Math or Portuguese) (binary: yes or no)
- Failures: Nombre d'échecs de cours passés (numeric: n if 1<=n<3, else 4)

b. Effectuer un prétraitement sur les colonnes featured
R/
On va utiliser le prétraitement d'encodage catégoriel pour les colonnes "Reason","Higher","Paid","Guardian" et "Passed"

```
from sklearn import preprocessing
lb_make = preprocessing.LabelEncoder()
data["reason_code"] = lb_make.fit_transform(data["reason"])
data["guardian_code"] = lb_make.fit_transform(data["guardian"])
data["higher_code"] = lb_make.fit_transform(data["higher"])
data["paid_code"] = lb_make.fit_transform(data["paid"])
```

```
data[["reason", "reason_code"]].head(6)
```

	reason	reason_code
0	course	0
1	course	0
2	other	2
3	home	1
4	home	1
5	reputation	3

```
data[["higher", "higher_code"]].head
```

```
29      yes            1
..      ...          ...
365     yes            1
366     yes            1
367     yes            1
368     yes            1
369     yes            1
370     yes            1
371      no            0
372     yes            1
373     yes            1
374     yes            1
375     yes            1
376     yes            1
```

```
data[["guardian",  "guardian_code"]].head(15)
```

	guardian	guardian_code
0	mother	1
1	father	0
2	mother	1
3	mother	1
4	father	0
5	mother	1
6	mother	1
7	mother	1
8	mother	1
9	mother	1
10	mother	1
11	father	0
12	father	0
13	mother	1
14	other	2

```
In [121]: data[["higher", "higher_code"]].head
```

```
370     yes            1
371      no            0
372     yes            1
373     yes            1
```

```
data[["paid", "paid_code"]].head(3)
```

	paid	paid_code
0	no	0
1	no	0
2	yes	1

c. Partager les données en training et test.

```
targets = lb_make.fit_transform(data_clean["passed"])
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, targets, test_size=.5)
```

4. Modèle d'apprentissage et test évaluation :

   a. Choisir les modèles d'apprentissage supervisé qui sont appropriés pour ce problème. On pourra utiliser scikit-learn pour l'implémentation.

      R/
      Les modèles d'apprentissage de classification doivent être utilisés car la source de données contient des variables (prédicteurs) qui sont des catégories, telles que "yes" ou "no".

      On peut utiliser des méthodes comme les Arbres de Décision, qui sont capable de gérer des données numériques et catégoriques.
      Une autre méthode qu'on peut utiliser est la Régression Logistique qui mesure la relation entre la variable dépendante catégorielle et une ou plusieurs variables indépendantes en estimant les probabilités.

      J'ai choisi le modelé de l'arbre de décision.

   b. Indiquer la nature du modèle proposé en termes des avantages et des inconvénients.
      R/

      Arbre de Décision

      Avantages

- Ils sont flexibles qu'ils peuvent être surdimensionnés.
- Capable de gérer les données numériques et catégorielles
- Simple à comprendre et à interpréter. Les arbres peuvent être visualisés.
- Ils peuvent automatiquement prendre en compte les interactions entre les variables, par exemple si vous avez deux entités indépendantes x et y.

Inconvénients

- Si l'arbre de décision est constitué d'un grand nombre de nœuds, il peut être nécessaire d'effectuer un effort considérable pour comprendre tous les fractionnements qui mènent à une prédiction particulière.
- Les arbres de décision peuvent être instables car de petites variations dans les données peuvent générer un arbre complètement différent.

c. Quelles sont les critères qui ont fait que vous avez choisi cette approche? Est-ce que c'est basé sur la nature des données que vous avez reçu?
R/

Les données contiennent des variables qui sont principalement des catégories. On a aussi la variable cible "passed" avec les valeurs possibles "yes" ou "no". De même, prends plusieurs prédicteurs e choisi les meilleur et voir le résultat graphiquement sont des avantages qu'on peut prendre de cette modelé.

d. Pour chaque modèle, donner une table qui montre le temps nécessaire pour l'apprentissage, temps pour faire la prédiction, le score F1 sur le set training, le score F1 sur le test set.

    i. On répète la même approche pour trois tailles différentes du training set (100, 200, 300). Pour chaque approche, on détermine
        1. Faire un fit du modèle d'apprentissage
        2. Prédire les labels pour apprentissage et test
        3. Mesurer le score F1.
        4. Note : à chaque fois, il faut garder la taille de l'ensemble de test constant. 5. Choix du meilleur modèle : indiquer lequel serait le meilleur modèle pour les données que vous avez reçu. Indiquer vos critères tel que : données disponibles, ressources de calcul, cout, performance.

Le meilleur modèle est avec un training set de 100. La ‘Accuracy’ et la Précision est plus grande. Même le F1 Score :

Training set 100 :

```
targets = lb_make.fit_transform(data_clean["passed"])
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, targets, test_size=100)
```

```
pred_train.shape
pred_test.shape
tar_train.shape
tar_test.shape
```

```
(100,)
```

```
classifier=DecisionTreeClassifier()
classifier=classifier.fit(pred_train,tar_train)
```

```
print("Matrice de confusion:")
predictions=classifier.predict(pred_test)
sklearn.metrics.confusion_matrix(tar_test,predictions)
```

```
Matrice de confusion:

array([[12, 15],
       [14, 59]], dtype=int64)
```

```
print("Accuracy: ")
sklearn.metrics.accuracy_score(tar_test, predictions)
```

```
Accuracy:

0.71
```

```
print("Precision: ")
sklearn.metrics.precision_score(tar_test, predictions)
```

```
Precision:

0.7972972972972973
```

```
print("Recall: ")
sklearn.metrics.recall_score(tar_test, predictions)
```

```
Recall:

0.8082191780821918
```

```
print("F1 Score: ")
sklearn.metrics.f1_score(tar_test, predictions)
```

```
F1 Score:

0.802721088435374
```

Training set 200 :

```
targets = lb_make.fit_transform(data_clean["passed"])
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, targets, test_size=200)
```

```
pred_train.shape
pred_test.shape
tar_train.shape
tar_test.shape
```

```
(200,)
```

```
classifier=DecisionTreeClassifier()
classifier=classifier.fit(pred_train,tar_train)
```

```
print("Matrice de confusion:")
predictions=classifier.predict(pred_test)
sklearn.metrics.confusion_matrix(tar_test,predictions)
```

```
Matrice de confusion:

array([[ 28,  42],
       [ 22, 108]], dtype=int64)
```

```
print("Accuracy: ")
sklearn.metrics.accuracy_score(tar_test, predictions)
```

```
Accuracy:

0.68
```

```
print("Precision: ")
sklearn.metrics.precision_score(tar_test, predictions)
```

```
Precision:

0.72
```

```
print("Recall: ")
sklearn.metrics.recall_score(tar_test, predictions)
```

```
Recall:

0.8307692307692308
```

```
print("F1 Score: ")
sklearn.metrics.f1_score(tar_test, predictions)
```

```
F1 Score:

0.7714285714285715
```

Training set 300 :

```
targets = lb_make.fit_transform(data_clean["passed"])
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, targets, test_size=300)
```

```
pred_train.shape
pred_test.shape
tar_train.shape
tar_test.shape
```

```
(300,)
```

```
classifier=DecisionTreeClassifier()
classifier=classifier.fit(pred_train,tar_train)
```

```
print("Matrice de confusion:")
predictions=classifier.predict(pred_test)
sklearn.metrics.confusion_matrix(tar_test,predictions)
```

```
Matrice de confusion:

array([[ 46,  49],
       [ 79, 126]], dtype=int64)
```

```
print("Accuracy: ")
sklearn.metrics.accuracy_score(tar_test, predictions)
```

```
Accuracy:

0.5733333333333334
```

```
print("Precision: ")
sklearn.metrics.precision_score(tar_test, predictions)
```

```
Precision:

0.72
```

```
print("Recall: ")
sklearn.metrics.recall_score(tar_test, predictions)
```

```
Recall:

0.6146341463414634
```

```
print("F1 Score: ")
sklearn.metrics.f1_score(tar_test, predictions)
```

```
F1 Score:

0.6631578947368422
```

5. Choix du meilleur modèle : indiquer lequel serait le meilleur modèle pour les données que vous avez reçu. Indiquer vos critères tel que : données disponibles, ressources de calcul, cout, performance.

R/

Régression Logistique

Autre option est utiliser la Régression Logistique.

Avantages

- La régression logistique a tendance à être moins sensible au sur-ajustement.
- Les modèles de régression logistique sont décrits par leurs coefficients. Très intéressant pour les utilisateurs qui connaissent un peu leurs données et souhaitent connaître l'influence de champs de saisie sur l'objectif.

Inconvénients

- Avec la régression logistique, vous devrez ajouter manuellement les interactions entre les variables.
- Les modèles de régression logistique étant entièrement décrits par leurs coefficients, ils attirent les utilisateurs qui connaissent un peu leurs données et souhaitent connaître l'influence de champs de saisie particuliers sur l'objectif.

Avec la application du modèle de Regression Logistic, les valeurs de «Accuracy », « Sensitivity » et « Precision » sont adequats, mais le valeur de False Positive Rate est très élevé. Il montre que le modelé obtenu est incorrect.

```
print("Classification Accuracy: ")
print("Question: Globalement, quel est le pourcentage ou le modele est correct: ")
print((TP+TN)/float(TP+TN+FP+FN))
print(metrics.accuracy_score(tar_test,tar_pred_class))
```

```
Classification Accuracy:
Question: Globalement, quel est le pourcentage ou le modele est correct:
0.7171717171717171
0.7171717171717171
```

```
print("Classification Error: ")
print("Question: Globalement, quel est le pourcentage ou le modele est incorrect: ")
print((FP+FN)/float(TP+TN+FP+FN))
print(1-metrics.accuracy_score(tar_test,tar_pred_class))
```

```
Classification Error:
Question: Globalement, quel est le pourcentage ou le modele est incorrect:
0.2828282828282828
0.2828282828282829
```

```
print("Sensitivity: True positive Rate ou Recall")
print("Question: Quand la valeur vraie est positive, combien de fois la prediction est correcte: ")
print((TP)/float(TP+FN))
```

```
Sensitivity: True positive Rate ou Recall
Question: Quand la valeur vraie est positive, combien de fois la prediction est correcte:
0.9682539682539683
```

```
print("Specificity: ")
print("Question: Quand la valeur vraie est negative, combien de fois la prediction est correcte: ")
print((TN)/float(TN+FP))
```

```
Specificity:
Question: Quand la valeur vraie est negative, combien de fois la prediction est correcte:
0.2777777777777778
```

```
print("False positive rate: ")
print("Question: Quand la valeur vraie est negative, combien de fois la prediction est incorrecte: ")
print((FP)/float(TN+FP))
```

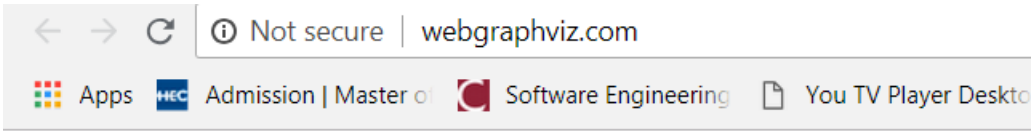
```
False positive rate:
Question: Quand la valeur vraie est negative, combien de fois la prediction est incorrecte:
0.7222222222222222
```

```
print("Precision: ")
print("Question: Quand une valeur positive est predite, combien de fois la prediction est correcte: ")
print((TP)/float(TP+FP))
print(metrics.precision_score(tar_test,tar_pred_class))
```

```
Precision:
Question: Quand une valeur positive est predite, combien de fois la prediction est correcte:
0.7011494252873564
0.7011494252873564
```

Arbre de Décision

Après l'utilisation de l'application web webgraphviz.com on peut voir l'arbre de décision complète :



WebGraphviz is Graphviz in the Browser

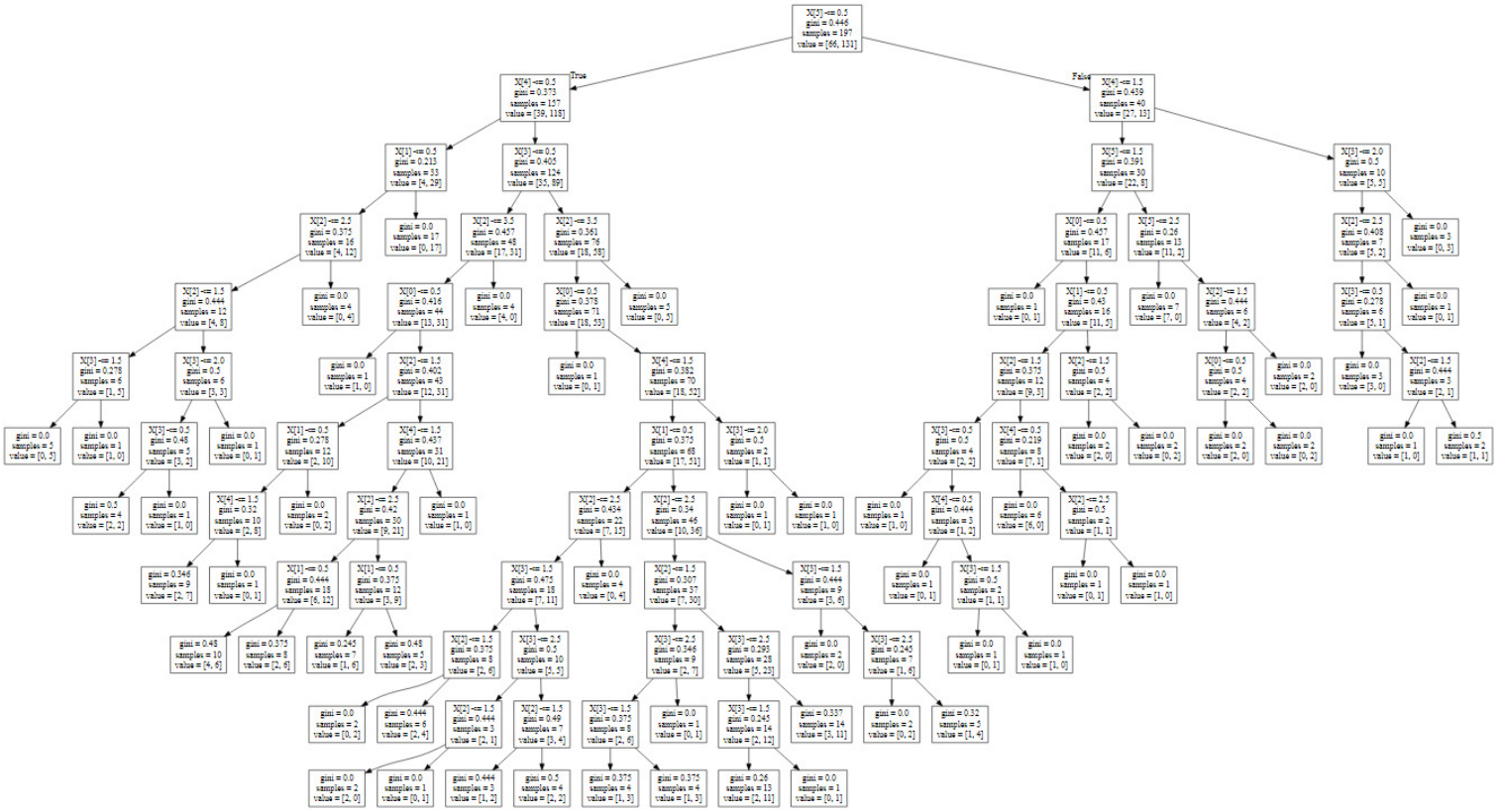
Enter your graphviz data into the Text Area:

(Your Graphviz data is private and never harvested)

Sample 1 Sample 2 Sample 3 Sample 4 Sample 5

```
100 [label="gini = 0.0\nsamples = 6\nvalue = [6, 0]"] ;
96 -> 100 ;
101 [label="gini = 0.0\nsamples = 1\nvalue = [0, 1]"] ;
77 -> 101 ;
102 [label="X[2] <= 2.5\ngini = 0.48\nsamples = 10\nvalue = [4, 6]"] ;
76 -> 102 ;
103 [label="X[0] <= 0.5\ngini = 0.5\nsamples = 8\nvalue = [4, 4]"] ;
102 -> 103 ;
104 [label="gini = 0.0\nsamples = 1\nvalue = [1, 0]"] ;
103 -> 104 ;
105 [label="X[2] <= 1.5\ngini = 0.49\nsamples = 7\nvalue = [3, 4]"] ;
103 -> 105 ;
106 [label="gini = 0.0\nsamples = 1\nvalue = [0, 1]"] ;
105 -> 106 ;
107 [label="X[5] <= 2.5\ngini = 0.5\nsamples = 6\nvalue = [3, 3]"] ;
105 -> 107 ;
108 [label="X[5] <= 1.5\ngini = 0.48\nsamples = 5\nvalue = [3, 2]"] ;
107 -> 108 ;
109 [label="X[1] <= 0.5\ngini = 0.5\nsamples = 4\nvalue = [2, 2]"] ;
108 -> 109 ;
110 [label="X[4] <= 0.5\ngini = 0.444\nsamples = 3\nvalue = [2, 1]"] ;
```

Generate Graph!



- X(0) : Higher : Veut suivre des études supérieures (yes=1 or no=0)
- X(1) : Paid: Cours supplémentaires payés dans le cours de la matière (Math or Portuguese) (yes=1 or no=0)
- X(2): Studytime: Temps d'étude hebdomadaire (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
- X(3): Reason : Raison de choisir cette école (nominal: close to "home"=1, school "reputation"=3, "course"=0 preference or "other"=2)
- X(4): Guardian: Tuteur de l'étudiant ("mother"=1, "father"=0 or "other"=2)
- X(5) : Failures: Nombre d'échecs de cours passés (numeric: n if 1<=n<3, else 0)

Interprétation de l’arbre :

1. Le premier nœud nous indique qu’il y a 197 individus dont 131 ont 1 ou plusieurs nombre d'échecs de cours passés.
2. Le nœud (2em niveau) à gauche nous indique que de 157 individus, 118 a comme tuteur sa mère ou autre.
3. Le nœud (3em niveau) à gauche nous indique que de 124 individus, 89 ont un raison de choisir cette école diffèrent à suivre un cours spécial.
4. Le nœud (4em niveau) à droit nous indique que de 76 individus, 58 étudient plus de 5 heures par semaine.
5. Le nœud (5em niveau) à gauche nous indique que de 71 individus, 53 veulent suivre des études supérieures.
6. Le nœud (6em niveau) à droit nous indique que de 70 individus, 52 a comme tuteur sa mère ou autre.

   On peut noter la relation entre les étudiants qui ont réussi, qui ont comme tuteur sa mère ou autre (diffèrent à son père), et le vouloir de suivre des études supérieures. De même, ils ont aussi une relation avec les étudiantes qui étudie plus de 5 heures par semaine.
   66.49% des étudiants qui ont 1 ou plusieurs nombre d'échecs de cours passés ont réussi.

References

1. https://blog.bigml.com/2016/09/28/logistic-regression-versus-decision-trees/
2. http://scikit-learn.org/stable/modules/tree.html
3. https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/