Implémentation des SVM et performances : Cas de la detection de fraude sur les cartes de crédits

Mohamed HAIDARA, Cheick-Oumar KABA, Ange Michel KOFFI

18 novembre, 2019

Contexte de l'étude

Cette étude s'inscrit dans le cadre de l'implémentation des performances des modèles SVM (Machines à vecteurs de support) dans le cadre de la détection de fraude sur les cartes de crédit.
Pour ce faire, nous disposons d'une base de données disponibe sur le site Kaggle.
Cette base contient 284 807 transactions.

I) Visualisation et transformation des données

1) Visualisation de la base de données

Voici un extrait de la base :

```
        Time          V1          V2          V3          V4         V5
135912  81458  1.21390652  0.04824167 -0.25217247  0.04178873 -0.1061481
71828   54459 -0.28750073  0.55982449  1.61330599 -0.69461537 -0.1749067
201876 134071  0.14398381  1.02061522 -0.41354902 -0.54432921  0.9234124
260420 159550  2.29965758 -1.28036224 -1.42704009 -1.75809404 -0.8773814
234569 148026 -0.06639259  0.25910527  0.04024118 -1.75227339 -0.1048339
94083   64721 -0.21060291  0.92661080  1.03916088  0.56443994  1.1913895
                V6         V7          V8          V9         V10
135912 -1.140235591  0.5046239 -0.36846703 -0.16993469 -0.16113134
71828  -0.416867024  0.4142733  0.03276763 -0.21603550 -0.70790239
201876 -0.838525569  0.9515489 -0.09976900 -0.02212812 -0.82728403
260420 -0.684676800 -0.9205465 -0.17196469 -1.75792042  1.87005171
234569 -0.172437845 -0.2462922  0.05465245 -0.66386764 -0.08788168
94083  -0.007156605  1.2662158 -0.44587661 -0.76339144  0.49072266
             V11        V12         V13         V14        V15
135912 -0.4169656  0.3391568  0.46732332  0.38545851  0.7769177
71828  -0.1070519  0.1622994  0.34781662 -0.01399711  1.2600460
201876 -0.7378178 -0.2031656 -0.13686331 -0.98903096 -0.2210380
260420  0.7446925 -0.6148579 -0.24819891  0.19379425 -0.4119043
234569  1.2511136 -0.3943026  0.04029676 -2.16710104 -0.7019877
94083   1.2872889  0.2877754 -0.26757225  0.31250327  0.4414902
              V16         V17          V18         V19          V20
135912  0.12277858 -0.32047751 -0.916146408  0.39048484  0.087261222
71828   0.01266128 -0.08975091 -0.733538682 -0.19644040  0.020113298
201876  0.34503290  0.36606342 -0.190305050 -0.21803662  0.008815657
260420 -0.37907690  0.18112393  0.642132745 -0.04556992 -0.441188515
234569  1.82878541  1.01949516  0.075034071  0.20640681  0.012037459
94083  -0.76344913 -0.36749546  0.003676771  0.61989169  0.077788135
              V21         V22         V23        V24         V25
135912 -0.36360940 -1.20855935  0.04570258  0.0734923  0.24654814
```

```
71828  -0.01853168 -0.07132669  0.01605853  0.1359985 -0.37633792
201876 -0.34524047 -0.85186521  0.09183975  0.5551621 -0.42301219
260420  0.11169573  0.70688008  0.10393788  0.7627088  0.05473332
234569  0.39882843  0.99357430 -0.09169154  0.5975256 -0.42822710
94083   0.10040145  0.64731721 -0.33313857 -0.2849795 -0.22851387
              V26         V27          V28 Amount Class
135912  0.71824636 -0.11737807  0.007945493  70.00     0
71828   0.89214114 -0.03328290  0.014156724  20.61     0
201876  0.11426441  0.22163958  0.083876266   1.29     0
260420  0.01657899 -0.02628694 -0.064889712  15.00     0
234569 -0.24872642 -0.34109433  0.047856153  19.95     0
94083  -0.33950318 -0.29367580 -0.308416024   1.00     0
```

Proportion de de la variable d'interet "Class" dans la base de données

0 : cas de non fraude, 1 : Cas de fraude

```
          0           1
0.998272514 0.001727486
```

On remarque un déséquilibre dans les modalités de la variable d'intéret. La classification non équilibrée pose des problèmes à de nombreux algorithmes d'apprentissage.Dans un soucis d'éfficacité dans l'estimation de notre modèle, il convient tout d'abord de procéder à un reéquilibrage des donnnées.

2) Fonction SMOTE de R et réequilibrage des données

La fonction SMOTE du package "DMwR" de R permet de gérer les problèmes de classification non équilibrée. L'idée générale de cette méthode est de générer artificiellement de nouveaux exemples de la classe minoritaire en utilisant les voisins les plus proches et à sous-échantillonner la classe majoritaire pour aboutir à un jeu de données plus équilibré.

Les arguments de la fonction SMOTE

Perc.over : controle la quantité du sur-échantillonnage de la classe minoritaire.
Perc.under : controle la quantité du sous-échantillonnage de la classe majoritaire.
k : Nombre indiquant le nombre de voisins les plus proches utilisés pour générer les nouveaux exemples de la classe minoritaire.

Pour plus de détails sur la fonction SMOTE de R, cliquez ici

Nous créons ainsi une nouvelle base de données contenant **4428** lignes qui nous servira pour la modélisation. Cette nouvelle base admet la répartition suivante pour la variable d'intérêt "Class" :

```
        0         1
0.6666667 0.3333333
```

II) Implémentation des SVM sur la base de données

Les SVM (Machines à Vecteur de Support, ou encore séparateurs à vaste marge) sont des techniques d'apprentissage supervisé qui s'inscrivent dans le cadre de la résolution de problèmes d'apprentissage. Ces

méthodes reposent sur deux notions clés : les notions de marge maximale et de fonction de noyau (kernel). Selon que l'on soit en présence d'un échantillon linéairement séparable (moins probable, mais utile pour expliquer de façon simple le fonctionnement des SVM) soit non linéairement séparable, le principe fondamentale des SVM consiste à partir d'un jeu d'entré X, à trouver une frontière séparatrice sur l'échantillon d'apprentissage qui discrmine entre les classes de la variable d'intéret Y.

1) Principe du SVM

Cas d'un échantillon linéairement séparable

Dans le cas des échantillons linéairement séparables,l'algorithme des SVM consiste à déterminer sur l'échantillon d'apprentissage la frontière de séparation optimale. Celle-ci est obtenue par un programme de maximisation la marge. La marge désigne la distance entre la frontière (hyperplan) de séparation et les observations les plus proches. Ces derniers sont appelés vecteurs supports.

Cas d'un échantillon non linéairement séparable

Dans le cas où les données ne sont pas linéairement séparables, la deuxième idée clé des SVM est de transformer l'espace de représentation des données d'entrée en un espace de plus grande dimension en utilisant une fonction noyau (Kernel) qui ne nécessite pas la connaissance explicite de la transformation à appliquer pour le changement d'espace. La séparation parfaite dans ce cas de figure paraît comme une vue de l'esprit. En pratique, il arrive que des individus soient du mauvais côté de la frontière. On introduit ainsi des variables de relaxation des contraintes de classification appelées variables "ressort". On admet donc qu'il peut y avoir des erreurs dans notre modèle, qu'il convient donc de contrôler. On introduit alors un paramètre de pénalisation C des erreurs qui contrôle par ailleurs les risques de sur-apprentissage ou de sous-appprentissage.

1) Algorithme d'implémentation des SVM

1ère étape: Sélection de la fonction Kernel sur l'échantillon d'apprentissage. Quelques kernels usuels :
- **Linéaire** : $u'v$
- **Polynomial** : $(\gamma u'v + coef0)^{degree}$
- **Radial** : $exp(-\gamma|u-v|^2)$
- **Sigmoïde** : $tanh(\gamma u'v + coef0)$

$\gamma, degree, coef0$ désignant les hyperparamètres selon la fonction noyau choisit.

2ème étape : Sélection du paramètre de pénalisation C sur l'échantillon d'apptentissage

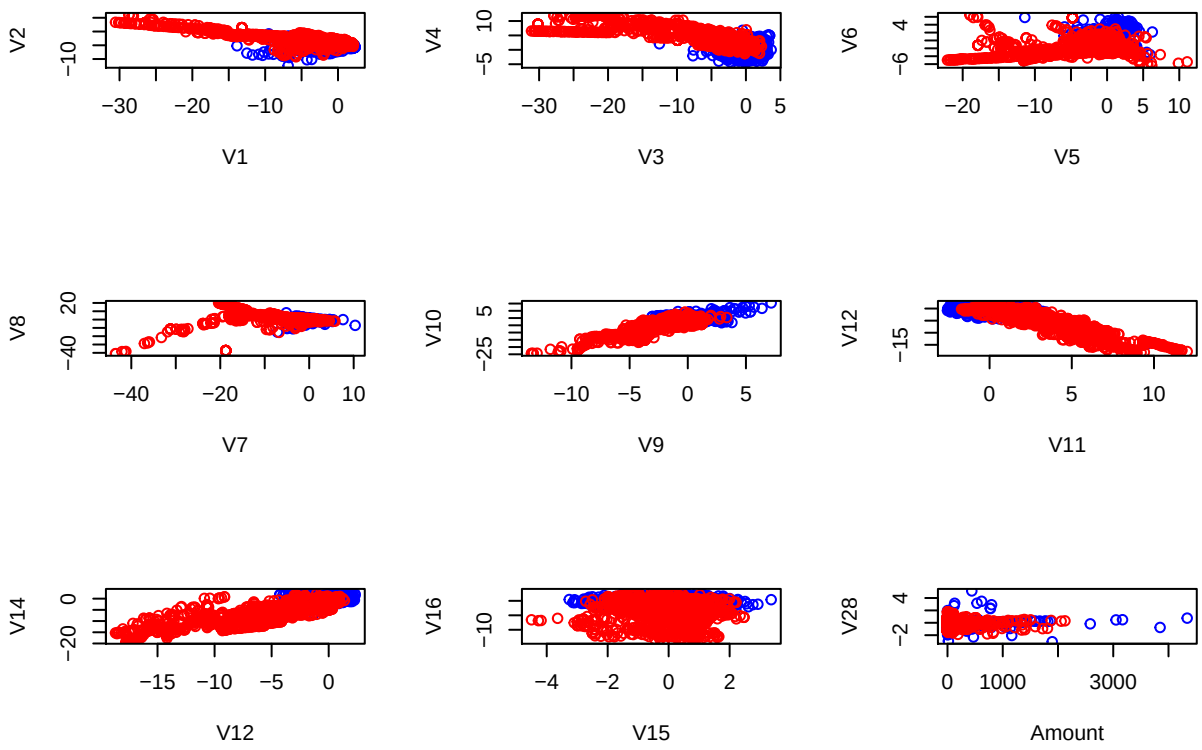
3ème étape : Mesure de la performance prédictive sur l'échantillon test

Remarque : Dans la pratique, les performances du SVM sont très sensibles au choix des hyper-paramètres des fonctions noyaux et du paramètre de pénalisation. Ainsi, une étape intermédiaire consiste à trouver les valeurs optimales de ces paramètres avant d'étudier la performance du modèle. Une solution consiste à déterminer ces hyper-paramètres par validation croisée sur l'échantillon d'apprentissage.

2) Cas pratique : application des SVM au cas de la détection de fraude sur les cartes de crédit

Cas échantillon linéairement/non linéairement séparable

Nous réalisons une représentation de quelques variables explicatives en fonction des modalités de la variable d'intérêt "Class" de la base pour avoir une idée du cas d'échantillon auquel nous sommes confronté.



Nous avons donc affaire à un échantillon non linéairement séparable.

Echantillon d'apprentissage / échantillon test

Nous choisissons d'utiliser 70% des observations de l'écantillon initial comme échantillon d'apprentissage (newdata.train) et les 30% restants comme notre échantillon test (newdata.test).

Sélection des hyper-paramètres optimaux

La fonction tune(svm...) du package "e1071" de R nous donne la possiblité de déterminer automatiquement par validation croisée la fonction kernel et les valeurs des hyper-paramètres optimaux associés de façon automatique.
Ici nous faisons une validation croisée 10-Fold avec différentes valeurs des hyper-paramètres sur l'échantillon d'apprentissage. La syntaxe est la suivante :

```
obj <- tune(svm, Class ~ ., data = newdata.train, ranges =
       list(kernel=c('linear','polynomial','radial', 'sigmoid'),
       cost =c(0.1,0.5,1.0,2.0,10),gamma=c(0.01,0.1,0.5,1,2)),
       tunecontrol = tune.control(sampling="cross"))
```

Nous obtenons le résultat suivant :

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 kernel cost gamma
 radial   10   0.1

- best performance: 0.02
```

Le taux d'erreur minimal sur l'échantillon d'apprentissage est obtenu en utilisant un kernel radial, pour une valeur optimale de l'hyper-paramètre γ égale à 0.1 et à 10 pour le paramètre de pénalisation C.

Evaluation du meilleur modèle sur l'échantillon d'apprentissage

Nous réalisons un modèle SVM sur l'échantillon d'apprentissage

```
svm.rad.best <- svm(Class ~.,data=newdata,subset=train,kernel="radial",probability=T,gamma=0.1,cost=10)
summary(svm.rad.best)
```

```
Call:
svm(formula = Class ~ ., data = newdata, kernel = "radial", probability = T,
    gamma = 0.1, cost = 10, subset = train)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  10

Number of Support Vectors:  658

 ( 361 297 )


Number of Classes:  2

Levels:
 0 1
```

Performance prédictive du meilleur modèle sur l'échantillon test

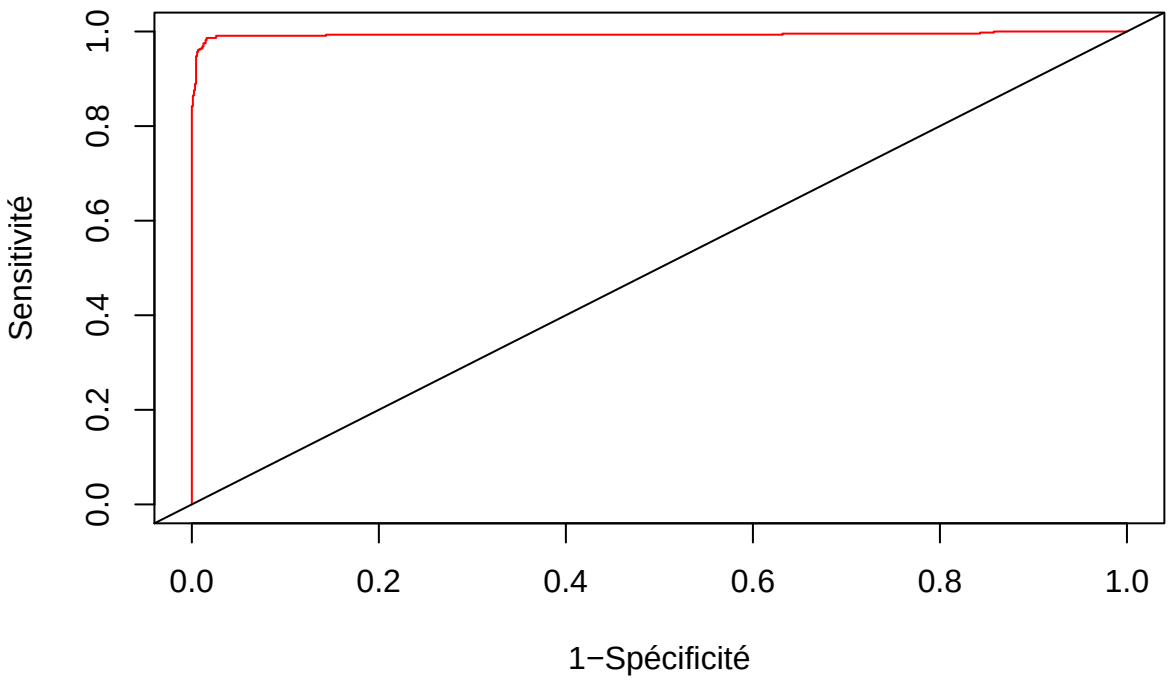
matrice de confusion

```
             newdata.class
pred.rad.best   0   1
            0 876  16
            1   9 427
```

taux d'erreur

```
[1] 0.0188253
```

Courbe ROC



AUC

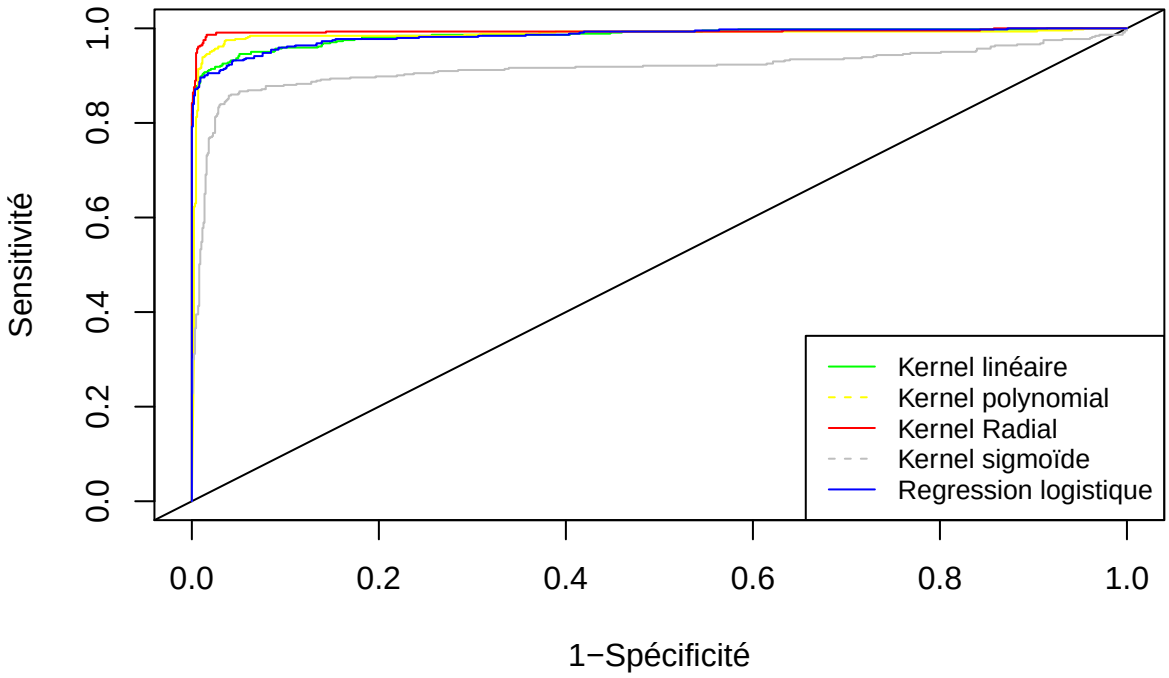
```
[[1]]
[1] 0.9935239
```

Indice de Gini

```
[1] 0.9870477
```

3) Comparaison des performances du meilleur modèle à d'autres kernels et benchmarks

Courbe ROC



Conclusion

Les SVM, méthodes d'apprentissage supervisé peuvent se montrer très utiles pour plusieurs raisons, à savoir leur :

- Capacité à traiter de grandes dimensionnalités
- Traitement des problèmes non linéaires avec le choix des noyaux
- Robustesse par rapport aux points aberrants (contrôlée avec le paramètre C)
- Robustesse par rapport aux problèmes de multicolinéarité Cependant, cette technique presénte également quelques inconvénients, qui sont :
- La difficulté à identifier les bonnes valeurs des paramètres (et sensibilité aux paramètres)
- Le problème lié au cas d'échantillon déséquilibré
- La difficulté d'interprétations (exemple : pertinence des variables)

Nous avons construit un démonstrateur en ligne avec R shiny qui vous permettra de vous rendre compte par vous même de la facilité d'emploi des SVM, de visualiser ses performances en jouant sur le choix des kernels et des hyper-paramètres dans le cadre de la détection de fraude sur les cartes de crédit. Ce démonstrateur vous permettra par aillleurs de Comparez les performances des SVM à celle de la régression logistique.