Partie3-Prédiction

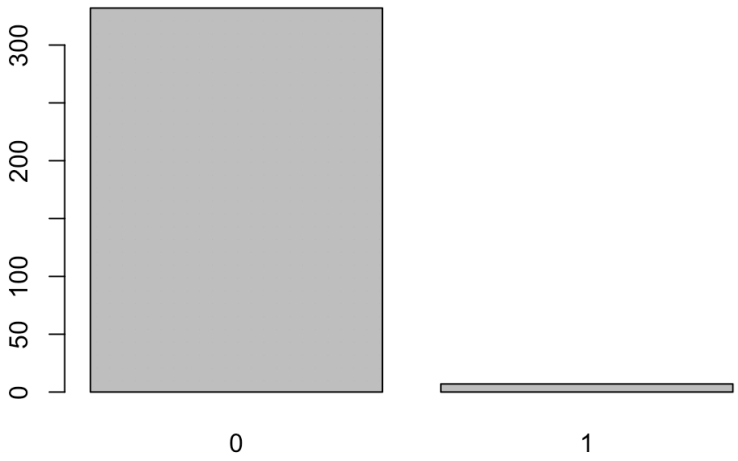
(1) Prédiction d'un Pokémon légendaire

1.Pour cette partie, j'ai utilisé trois algorithmes. Decision Tree， logistic et svm.

Decision Tree

```
setwd("/Users/shaqianqian/Desktop")

library("readr",
lib.loc="/Library/Frameworks/R.framework/Versions/3.4/Resources/library
")


data <- read_csv("pokemon.csv")


data <- data.frame(data)


data$japanese_name <- NULL


data$name <- NULL


data$abilities <- NULL


library("rpart",
lib.loc="/Library/Frameworks/R.framework/Versions/3.4/Resources/library
")


data2 <- na.omit(data)


m1 <- rpart(factor(is_legendary) ~ . , data = data2)


pred <- predict(m1, type = "class")

library("caret",
lib.loc="/Library/Frameworks/R.framework/Versions/3.4/Resources/library
")

tb <- table(pred, data2$is_legendary)

tb
```

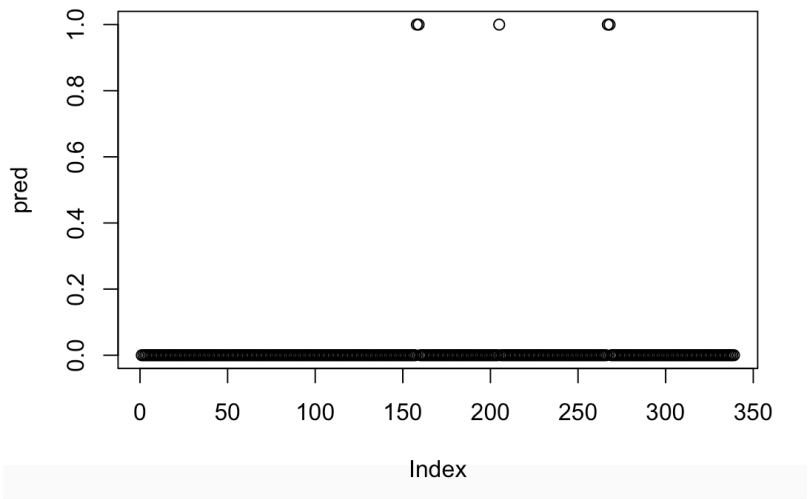
```
## pred   0   1
##    0 332   0
##    1   2   5

is-legendary :

precision <-  5/(5 + 2)
recall <-  5/(5 + 0)
accuracy<-(332+5)/(332+5+2+0)=99.4%
Fmeasure <- 2 * precision * recall / (precision + recall)
Fmeasure<-0.8333333

non is-legendary :

precision <-  332/(332 + 0)
recall <-  332/(332 + 2)=99.4%
accuracy<-(332+5)/(332+5+2+0)=99.4%
Fmeasure <- 2 * precision * recall / (precision + recall)
Fmeasure<-0.996
```



logistic

```
m2 <- glm(factor(is_legendary) ~ . , data = data2, family =
binomial(link = "logit"))

pred <- round(predict(m2, type = "response"))

tb <- table(pred, data2$is_legendary)

tb
```

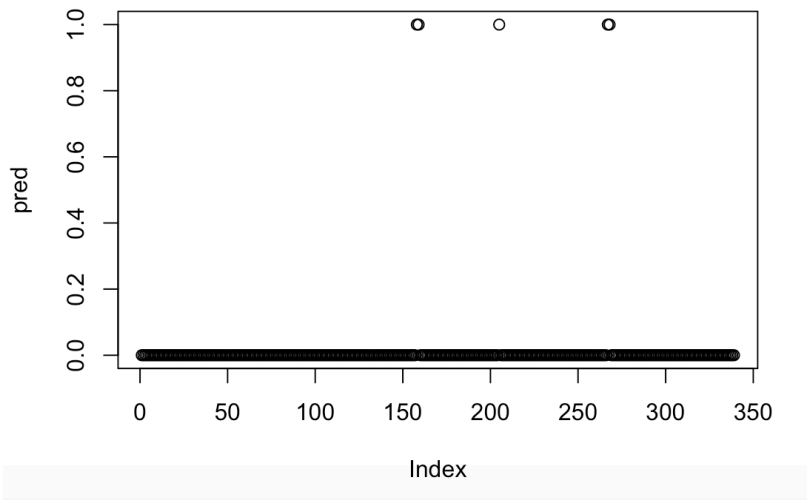
```
##
## pred   0   1
##    0 334   0
##    1   0   5

is-legendary :

precision <-  5/(5 + 0)
recall <-  5/(5 + 0)

Fmeasure <- 2 * precision * recall / (precision + recall)
Fmeasure=1



non is-legendary :

precision <-  334/(334 + 0)=1
recall <-  334/(334 + 0)=1

Fmeasure <- 2 * precision * recall / (precision + recall)
Fmeasure=1
```



svm

```
library("e1071",
lib.loc="/Library/Frameworks/R.framework/Versions/3.4/Resources/library
")

m3 <- svm(factor(is_legendary) ~ . , data = data2)

pred <- predict(m3, type = "class")
```

```
tb <- table(pred, data2$is_legendary)

tb

##
## pred   0   1
##    0 334   1
##    1   0   4

is-legendary :

precision <-  4/(4 + 0)
recall <-  4/(4 + 1)

Fmeasure <- 2 * precision * recall / (precision + recall)
Fmeasure= 0.8888889

non is-legendary :

precision <-  334/(334 + 1)=99.7%
recall <-  334/(334 + 0)=1
```



2.La modèle donnée plus performante est le logistique. Dans cette méthode, nos précision ,recall et F-mersure sont tous 1.

3. La régression logistique ou modèle logistique est un modèle de régression binomiale. Comme pour tous les modèles de régression binomiale, il s'agit de modéliser au mieux un modèle mathématique simple à des observations réelles

nombreuses. En d'autres termes d'associer à un vecteur de variables aléatoires {x1….xk} à une variable aléatoire binomiale génériquement notée y.

La régression logistique est une régression non linéaire probabiliste Pour la régression logistique binaire, la variable dépendante y n'a que deux valeurs de "oui et non", et les deux valeurs sont "oui et non", qui sont enregistrées comme 0 et 1. Supposons que sous l'influence de la variable indépendante x1, x2 ...., xp, la probabilité que y prenne "oui" soit p, alors la probabilité de prendre "non" est 1-p, et l'étude est celle où y prend " Est la probabilité d'occurrence p et l'argument x1, x2, x3 ..., xp

Étapes de modélisation du modèle de régression logistique :

1) Définir les variables d'index (variables dépendantes et variables indépendantes) en fonction du but de l'analyse, puis collectez les données.

2) La probabilité que Y vale 1 est p = P (y = 1 | X), et 0 est 1-p. Les équations de régression linéaire sont listées en utilisant Ln (p / 1-p) et des variables indépendantes pour estimer les coefficients de régression dans le modèle.

3) le test du modèle: La valeur F et la valeur p de la table ANOVA(Analysis of variance) délivrée permettent de vérifier l'équation de régression significative.Si la p-valeur est inférieure au seuil de signification du modèle obtenu par des essais, pouvant être testé suivant le coefficient de régression, alors l’equation est verifie sinon il faut resélectionner les variables indicatrices et rétablir l'équation de régression.

4) test de signification des coefficients de régression: Avec la methode de regression lineaire :l'équation de régression ne signifie pas que tout effet significatif sur la variable indépendante y sont importants, l'équation de régression permet d’éliminer les variables secondaires pour rétablir une équation de régression plus simple et plus efficace, pour cela il faut tester la signification de chaque variable indépendante. En utilisant la méthode de régression par étapes : Il faut enlevez d'abord la variable dépendante la moins significative, reconstruire l'équation de régression et passez le test au modèle et aux coefficients de régression participants.

5) Application du modèle: Il faut entrer la valeur de la variable indépendante pour obtenir la valeur du prédicteur, ou pour contrôler la valeur de la variable indépendante en fonction de la valeur du prédicteur.

(2) Prédiction d'un Pokémon generation

Decision Tree

```
setwd("/Users/shaqianqian/Desktop")

library("readr",
lib.loc="/Library/Frameworks/R.framework/Versions/3.4/Resources/library")



data <- read_csv("pokemon.csv")



data <- data.frame(data)

data2 <- na.omit(data)

newdata <- data2[,c(20,26,29,34,35,36,40)]

library("rpart",
lib.loc="/Library/Frameworks/R.framework/Versions/3.4/Resources/library")




dtree<-rpart(generation~.,data=newdata,method="class")

pred <- predict(dtree, type = "class")

library("caret",
lib.loc="/Library/Frameworks/R.framework/Versions/3.4/Resources/library")

tb <- table(pred, newdata$generation)

tb

pred  1  2  3  4  5  6  7

  1 18  6  9  3  2  4  6

  2  3 10  4  2  0  1  5

  3  3  5 10  2  0  1  0

  4 10  5 16 22 11  5  8

  5 21 19 16 15 47 18 16

  6  2  1  1  2  3  6  1
```

7 0 0 0 0 0 0 0

generation=1 :

precision <- 18/(18+6+9+3+2+4+6)=18/48=37.5%
recall <- 18/(18+3+3+10+21+2)=18/57=31.6%

Fmeasure <- 2 * precision * recall / (precision + recall)
Fmeasure= 34.3%

generation=2:

precision <- 10/(3+10+4+2+6)=40%
recall <- 10/(6+10+10+20)=21.7%

Fmeasure= 28.14%

generation=3:

precision <- 10/21=47.61%
recall <- 10/56=17.86%

Fmeasure= 25.98%

generation=4:

precision <- 22/77=28.57%
recall <- 11/60=18.33%

Fmeasure= 22.33%

generation=5:

precision <- 47/208=22.60%
recall <- 11/60=74.60%

Fmeasure= 34.69%

generation=6:

precision <- 6/16=37.5%
recall <- 11/60=17.14%

Fmeasure= 23.53%

Svm

```
library("e1071",
lib.loc="/Library/Frameworks/R.framework/Versions/3.4/Resources/library
")

m3 <- svm(factor(generation) ~ . , data = newdata)

pred <- predict(m3, type = "class")

tb <- table(pred, data2$generation)

tb

pred  1  2  3  4  5  6  7

   1 21  4  5  5  3  5  3

   2  1 16  2  2  0  0  0

   3 16  9 27 13  9  6  9

   4  0  0  0  5  0  0  1

   5 19 16 22 21 51 23 17

   6  0  0  0  0  0  0  0

   7  0  1  0  0  0  1  6
```

generation=1 :

precision <- 21/(21+4+10+11)=45.65%
recall <- 21/57=36.84%

Fmeasure <- 2 * precision * recall / (precision + recall)
Fmeasure= 40.77%

generation=2:

precision <- 16/21=76.19%
recall <- 16/46=34.78%

Fmeasure= 47.76%

generation=3:

precision <- 27/89=30.34%
recall <- 27/56=48.21%

Fmeasure= 37.24%

generation=4:

precision <- 5/6=83.33%
recall <- 5/41=12.19%

Fmeasure= 21.27%

generation=5:

precision <- 51/169=30.18%
recall <- 51/63=80.95%

Fmeasure= 43.98%

generation=7:

precision <- 6/8=75%
recall <- 6/36=16.67%

Fmeasure= 27.28%

2.La modèle donnée plus performante est svm.

3.svm : Les machines à vecteurs de support ou séparateurs à vaste marge (en anglais support vector machine, SVM) sont un ensemble de techniques d'apprentissage supervisé destinées à résoudre des problèmes de discriminationnote et de régression. Les SVM sont une généralisation des classifieurs linéaires.

Les SVM peuvent être utilisés pour résoudre des problèmes de discrimination, c'est-à-dire décider à quelle classe appartient un échantillon, ou de régression, c'est-à-dire prédire la valeur numérique d'une variable. La résolution de ces deux problèmes passe par la construction d'une fonction h qui à un vecteur d'entrée x fait correspondre une sortie y :

y=h(x)

On se limite pour l'instant à un problème de discrimination à deux classes (discrimination binaire), c'est-à-dire y in {-1,1}, le vecteur d'entrée x étant dans un espace X muni d'un produit scalaire. On peut prendre par exemple X= R ^N

Prédiction d’un Pokémon type1

Decision Tree

```
setwd("/Users/shaqianqian/Desktop")

library("readr",
lib.loc="/Library/Frameworks/R.framework/Versions/3.4/Resources/library")



data <- read_csv("pokemon.csv")
```

```
data <- data.frame(data)

newdata <- data[,c(20,26,29,34,35,36,37)]

dtree<-rpart(type1~.,data=newdata,method="class")


pred <- predict(dtree, type = "class")

library("caret",
lib.loc="/Library/Frameworks/R.framework/Versions/3.4/Resources/library")

tb <- table(pred, newdata$type1)

tb
```

```
pred      bug dark dragon electric fairy fighting fire flying ghost grass ground ice
 bug     29   1    0     1    1     1    0    0    1    6    3    0
 dark     0   0    0     0    0     0    0    0    0    0    0    0
 dragon   0   0    8     0    0     0    0    0    0    0    1    0
 electric 1   2    0     8    0     0    0    0    3    2    0    0
 fairy    0   0    0     0    0     0    0    0    0    0    0    0
 fighting 0   0    0     0    0     8    0    0    0    0    1    0
 fire     5   6    0     9    0     1   18    2    0    6    0    2
 flying   0   0    0     0    0     0    0    0    0    0    0    0
 ghost    0   1    0     0    0     0    0    0    1    0    0    0
 grass    0   0    1     5    4     0    4    0    4   18    0    3
 ground   1   0    0     0    0     0    0    0    0    0    8    0
 ice      0   0    0     0    0     0    0    0    0    0    0    0
 normal   8   7    7     6    2    15    6    0    3    5   12    4
 poison   0   0    0     0    0     0    0    0    0    0    0    0
 psychic  2   1    1     0    3     0    2    0    4    1    0    3
 rock     4   1    0     0    0     0    0    0    3    2    0    2
 steel    0   1    0     0    0     0    0    0    0    0    0    0
 water   22   9   10    10    8     3   22    1    8   38    7    9


pred      normal poison psychic rock steel water
```

```
bug        3     0     4    0    1    2
dark       0     0     0    0    0    0
dragon     0     0     0    0    2    1
electric   2     1     1    0    0    1
fairy      0     0     0    0    0    0
fighting   1     0     0    0    0    0
fire       4     1     2    2    0    5
flying     0     0     0    0    0    0
ghost      0     0     0    0    0    0
grass      1     2     6    0    1    7
ground     0     0     0    0    0    1
ice        0     0     0    0    0    0
normal     73    14    2    9    0    14
poison     0     0     0    0    0    0
psychic    2     0     24   1    1    2
rock       0     1     1    17   6    5
steel      0     0     0    1    3    0
water      19    13    13   15   10   76
```

svm

```
library("e1071",
lib.loc="/Library/Frameworks/R.framework/Versions/3.4/Resources/library")

m3 <- svm(factor(type1) ~ . , data = newdata)

pred <- predict(m3, type = "class")

tb <- table(pred, newdata$type1)

tb
```

```
pred       bug dark dragon electric fairy fighting fire flying ghost grass ground ice
bug        29  1    0      1        1     1        1    0      0     1     6      3   0
dark       0   0    0      0        0     0        0    0      0     0     0      0   0
dragon     0   0    8      0        0     0        0    0      0     0     1      0
electric   1   2    0      8        0     0        0    0      3     2     0      0
fairy      0   0    0      0        0     0        0    0      0     0     0      0   0
fighting   0   0    0      0        0     8        0    0      0     0     1      0
```

fire	5	6	0	9	0	1	18	2	0	6	0	2	
flying	0	0	0	0	0	0	0	0	0	0	0	0	
ghost	0	1	0	0	0	0	0	0	1	0	0	0	
grass	0	0	1	5	4	0	4	0	4	18	0	3	
ground	1	0	0	0	0	0	0	0	0	0	8	0	
ice	0	0	0	0	0	0	0	0	0	0	0	0	
normal	8	7	7	6	2	15	6	0	3	5	12	4	
poison	0	0	0	0	0	0	0	0	0	0	0	0	
psychic	2	1	1	0	3	0	2	0	4	1	0	3	
rock	4	1	0	0	0	0	0	0	3	2	0	2	
steel	0	1	0	0	0	0	0	0	0	0	0	0	
water	22	9	10	10	8	3	22	1	8	38	7	9	

pred	normal	poison	psychic	rock	steel	water
bug	3	0	4	0	1	2
dark	0	0	0	0	0	0
dragon	0	0	0	0	2	1
electric	2	1	1	0	0	1
fairy	0	0	0	0	0	0
fighting	1	0	0	0	0	0
fire	4	1	2	2	0	5
flying	0	0	0	0	0	0
ghost	0	0	0	0	0	0
grass	1	2	6	0	1	7
ground	0	0	0	0	0	1
ice	0	0	0	0	0	0
normal	73	14	2	9	0	14
poison	0	0	0	0	0	0
psychic	2	0	24	1	1	2
rock	0	1	1	17	6	5
steel	0	0	0	1	3	0
water	19	13	13	15	10	76