Regression linéaire et régression logistique

Dans ce TP on considère la tâche de classification binaire suivante : prédire la survie des passagers du Titanic à partir d'informations sur ces passagers (sexe, âge, classe, nombre de membre de la famille à bord, etc.). Les données sont contenues dans le fichier `titanic.csv`.

1 Préparation des données

Charger les données et les stocker dans un dataframe `titanic`. Quelle est la variable à prédire ? En inspectant les données, vous réaliserez qu'elle contiennent des *valeurs manquantes* (notées "NA" pour "Not Available"). On peut visualiser le nombre de valeurs manquantes pour chaque prédicteur à l'aide de la commande

```
sapply(titanic,function(x) sum(is.na(x)))
```

Retirer de la base de données les variables `PassengerId, Name, Ticket, Cabin`, peu intéressantes pour la prédiction. Parmi les variables restantes, l'âge (variable quantitative) et le lieu d'embarquement (variable qualitative) contiennent toutes deux des valeurs manquantes. Appliquer les traitements suivants aux données : que font-il ?

```
titanic$Age[is.na(titanic$Age)] <- mean(titanic$Age,na.rm=T)
titanic <- titanic[!is.na(titanic$Embarked),]
```

2 Regression linéaire

On commencera comme d'habitude par séparer les données en un ensemble d'apprentissage `train` et un ensemble de test `test`. On peut faire une régression linéaire (sur l'ensemble d'apprentissage) avec

```
linmodel <- lm(Survived ~ .,data=train)
```

ou

```
linmodel <- lm(Survived ~ Age + Sex,data=train)
```

Dans chaque cas, on indique qu'on veut prédire la survie (`Survived`) en fonction de toutes les variables (`.`) ou de certaines d'entre elles uniquement (`Age + Sex`). `linmodel` est un *objet* de type modèle, qui a plusieurs attributs, comme `linmodel$coefficients`, l'estimateur des moindres carrés $\hat{\beta}^{\text{LS}}$, et différentes méthodes, comme `summary(linmodel)` qui affiche des informations complémentaires sur $\hat{\beta}^{\text{LS}}$.

La régression linéaire nécessite usuellement que toutes les variables explicatives soit quantitatives : comment la fonction `lm` a-t-elle implicitement transformé les variables de type facteur ?

2.1 Calcul de l'erreur de test

La fonction `predict` permet de calculer les valeurs (réelles) prédites à l'aide de l'estimateur des moindres carrés sur une base de données : pour une base de données X, elle retourne le vecteur des valeurs prédites $X\hat{\beta}^{\text{LS}}$. Le classifieur des moindres carrés est lui donné par $f^{\text{LS}}(x) = \mathbb{1}_{(x^T\hat{\beta}^{LS}>0.5)}$. Les commandes suivantes permettent de retourner les étiquettes prédites sur la base de données.

```
fitted.results <- predict(linmodel,newdata=test);
prediction <- ifelse(fitted.results > 0.5,1,0);
```

Calculer l'erreur de test du classifieur des moindres carrés. On pourra également vérifier que `fitted.results` est bien obtenu en calculant $X\hat{\beta}^{LS}$ où X est la matrice des prédicteurs utilisés dans la base de test.

2.2 Sélection de variables

La fonction `summary` affiche les composantes de $\hat{\beta}^{\text{LS}}$ mais également les valeurs de $t_k = \hat{\beta}_k^{\text{LS}}/\sqrt{v_k\hat{\sigma}^2}$ qui permettent de tester la significativité de chaque variable (voir cours).

Quelle est la variable la moins significative ? Refaire une régression linéaire sans cette variable. Par exemple, s'il s'agit de `Embarked` :

```
linmodel <- lm(Survived ~. - Embarked,data=train)
summary(linmodel)
```

Répéter ce processus jusqu'à ce que toutes les variables restantes soient significatives. Quelles sont ces variables ? Calculer l'erreur de test du classifieur des moindres carrés basé sur ces variables uniquement.

D'autres méthodes existent pour la sélection de variables, on pourra regarder notamment l'effet de la fonction `step()` sur le modèle.

Pour les variables sélectionnées, on pourra afficher avec la fonction *pairs* (cf. TP1) les nuages de points de chaque paire de variable, avec une coloration dépendent de la variable à prédire.

2.3 Regression régularisée

La régression régularisée (ou *ridge*) est implémenté avec la fonction `lm.ridge` du package `MASS`. Elle ne fonctionne toutefois pas exactement comme `lm` et on ne peut pas utiliser la fonction `summary`. Exécuter les commandes suivantes.

```
require(MASS)
Lambdas <- c(seq(0.001,0.01,1),seq(1,2000,1))
model_lin <- lm.ridge(Survived ~.,data=train,lambda=Lambdas)
betasRidge <- coef(model_lin)
```

Le vecteur `Lambdas` contient différentes valeurs du paramètre de régularisation. Que contient la matrice `betasRidge` ? On pourra essayer avec des vecteurs `Lambdas` plus petits (voire réduit à un élément) pour comprendre.

Pour calculer les valeurs prédites sur l'ensemble de test, il faut alors former la matrice de $X \in \mathcal{M}_{nTest\times p}(\mathbb{R})$ des p variables utilisées pour la prédiction pour les $nTest$ observations. Après avoir définit les nouveaux attributs `Sexmale`, `EmbarkedQ` et `EmbarkedS`, on pourra utiliser la commande suivante.

```
X=as.matrix(cbind(array(1,dim(test)[1]),test$Pclass,test$
    Sexmale,test$Age,test$SibSp,test$Parch,test$Fare,test$
    EmbarkedQ,test$EmbarkedS))
fitted.results <- X %*% t(betasRidge)
```

`fitted.results` est une matrice `nTest` $\times$ `length(Lambdas)` dont chaque colonne est le vecteur (de taille $nTest$) des prédictions (réelles) sur l'ensemble de test. Calculer l'erreur de test pour chaque valeur de λ dans `Lambdas` et afficher celle-ci en fonction de λ.

```
PredictionError <- array(0,length(Lambdas))

for (k in 1:length(Lambdas)){
  PredictionError[k] <- mean(ifelse(fitted.results[,k
    ]>0.5,1,0)!=test$Survived)}
```

3 Regression logistique

La régression linéaire est une méthode initialement prévue pour la prédiction d'une valeur réelle, mais peut s'appliquer à la classification en utilisant un seuillage. Nous allons maintenant implémenter la régression logistique, dans le cadre de laquelle les sorties doivent être binaires.

```
model <- glm(Survived ~.,family=binomial,data=train)
summary(model)
fitted.results <- predict(model,newdata=test,type='response')
```

Le vecteur `fitted.results` contient les valeurs de

$$P(Y = 1|X = x) = \sigma(x^T\hat{\beta}^{\log})$$

pour x dans la base de données. Sans l'option 'response' la valeur des $x^T\hat{\beta}^{\log}$ est retournée. Calculer le vecteur `predict` des predictions binaires obtenues sur l'ensemble de test, puis l'erreur de test.

La fonction `summary` affiche également des informations complémentaires sur l'estimateur calculé par régression logistique. Effectuer comme pour la régression linéaire une sélection itérative des variables importantes, et calculer l'erreur de test associée au meilleur modèle.