08 Classification

Maribel Diaz | PRO1036

Prédire des données catégorielles

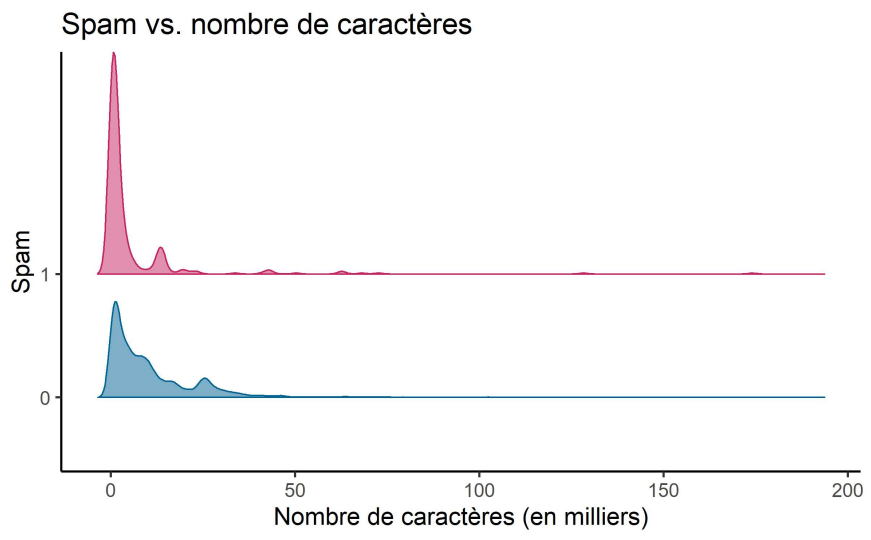
Filtres anti-spam

- Données provenant de 3921 courriels et de 21 variables les concernant
- Résultat : l'e-mail est-il un spam ou non ?
- Prédicteurs : nombre de caractères, présence ou non de " Re : " dans l'objet de l'e-mail, heure d'envoi de l'e-mail, nombre de fois où le mot " inherit " apparaît dans l'e-mail, etc.

```
library(openintro)
glimpse(email)
```

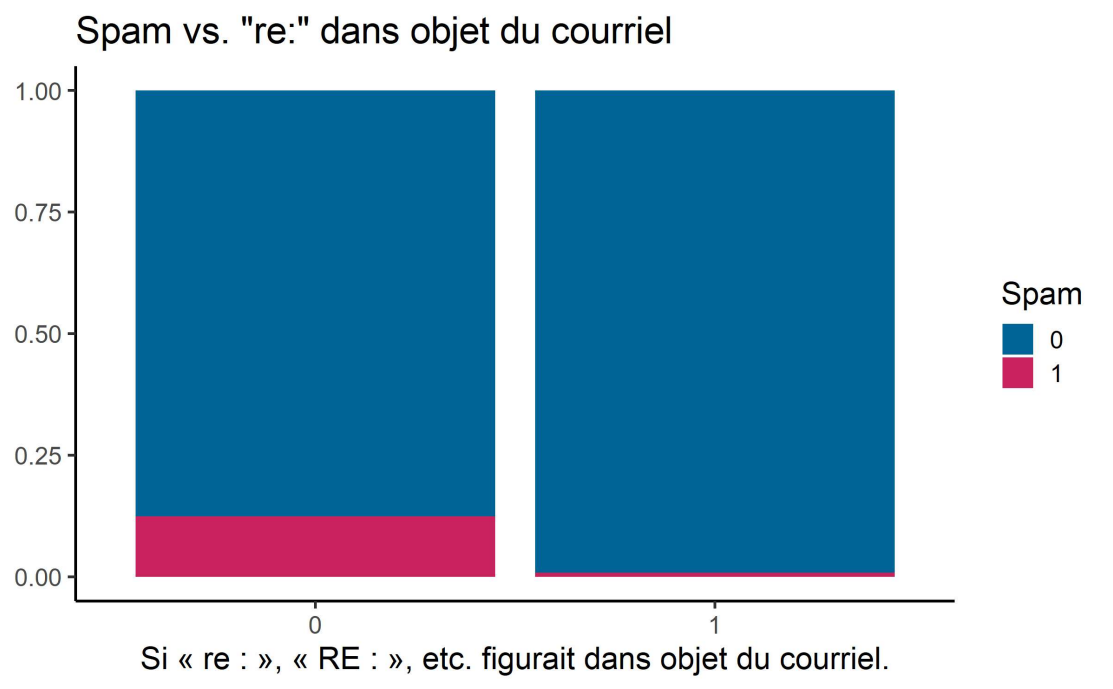
```
## Rows: 3,921
## Columns: 21
## $ spam         <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ to_multiple  <fct> 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, …
## $ from         <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
## $ cc           <int> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, …
## $ sent_email   <fct> 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, …
## $ time         <dttm> 2012-01-01 01:16:41, 2012-01-01 02:03:59,…
## $ image        <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, …
## $ attach       <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, …
## $ dollar       <dbl> 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ winner       <fct> no, no, no, no, no, no, no, no, no, no, no…
## $ inherit      <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ viagra       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ password     <dbl> 0, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ num_char     <dbl> 11.370, 10.504, 7.773, 13.256, 1.231, 1.09…
## $ line_breaks  <int> 202, 202, 192, 255, 29, 25, 193, 237, 69, …
## $ format       <fct> 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, …
## $ re_subj      <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, …
## $ exclaim_subj <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ urgent_subj  <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ exclaim_mess <dbl> 0, 1, 6, 48, 1, 1, 1, 18, 1, 0, 2, 1, 0, 1…
## $ number       <fct> big, small, small, small, none, none, big,…
```

Vous attendriez-vous à ce que des courriels plus longs ou plus courts soient considérés comme du spam ?




```
## # A tibble: 2 × 2
##   spam  mean_num_char
##   <fct>         <dbl>
## 1 0              11.3
## 2 1               5.44
```

Les courriels dont l'objet commence par « Re : », « RE : », « re : » ou « rE : » sont-ils considérés comme du spam ou non ?

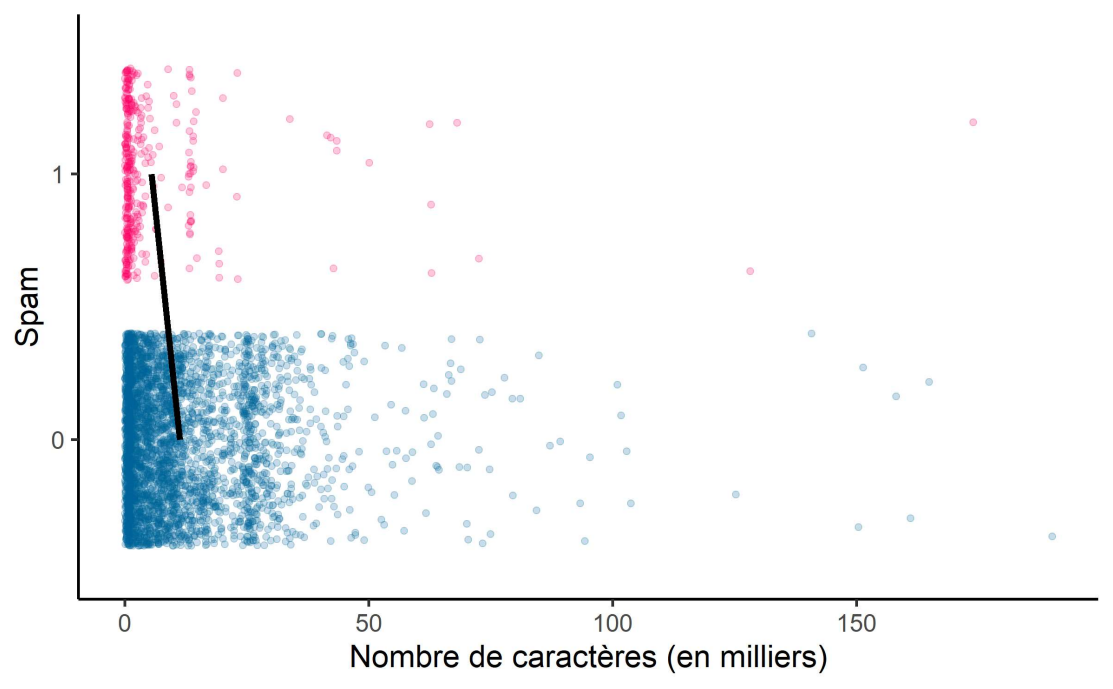


Modélisation du spam

- Le nombre de caractères et la présence ou non de « re : » dans l'objet du message peuvent être liés à la question de savoir si l'e-mail est un spam.
- Pour des raisons de simplicité, nous nous concentrerons sur le nombre de caractères (`num_char`) comme prédicteur, mais le modèle que nous décrivons peut également être étendu pour prendre en compte plusieurs prédicteurs.

Modélisation du spam

Les modèles linéaires ne peuvent pas décrire ce type de relation, donc nous devons donc utiliser un autre type de modèle.



Formuler le problème

- Nous pouvons considérer chaque résultat (spam ou non) comme un succès ou un échec résultant d'essais de Bernoulli distincts.
- Essai de Bernoulli : expérience aléatoire avec exactement deux résultats possibles, " success " et " failure ", dans laquelle la probabilité de succès est la même à chaque fois que l'expérience est menée.
- Chaque essai de Bernoulli peut avoir une probabilité de succès distincte.

$$y_i \sim Bern(p)$$

- Nous pouvons ensuite utiliser les variables prédictives pour modéliser cette probabilité de réussite, p_i
- Nous ne pouvons pas utiliser un modèle linéaire pour p_i (puisque p_i doit être compris entre 0 et 1), mais nous pouvons transformer le modèle linéaire pour obtenir la plage appropriée.

Modèles linéaires généralisés

- Il s'agit d'une façon très générale d'aborder de nombreux problèmes de régression et les modèles qui en résultent sont appelés **modèles linéaires généralisés (GLM)**
- La régression logistique c'est un exemple

Trois caractéristiques des GLM

Tous les GLM présentent les trois caractéristiques suivantes :

1. Une distribution de probabilité décrivant un modèle génératif pour la variable de résultat.
2. Un modèle linéaire :

$$\eta = \beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k$$

3. Une fonction de liaison qui relie le modèle linéaire au paramètre de la distribution des résultats.

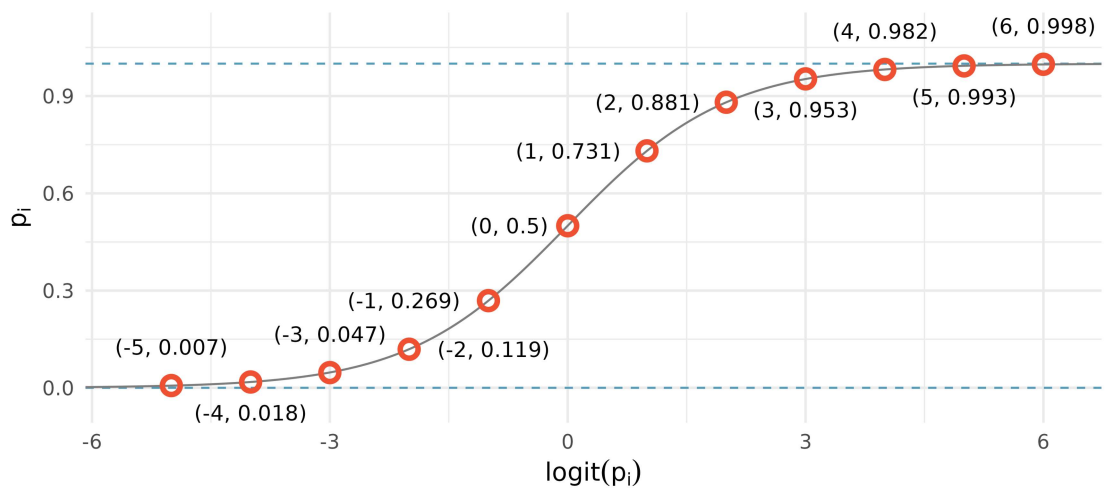
La régression logistique

La régression logistique

- La régression logistique est un GLM utilisé pour modéliser un résultat catégorique binaire à l'aide de prédicteurs numériques et catégoriels.
- Pour finir de spécifier le modèle logistique, il suffit de définir une fonction de lien raisonnable qui relie η_i à p_i : la fonction logit
- **la fonction logit :** Pour $0 \leq p \leq 1$

$$logit(p) = \log\left(\frac{p}{1-p}\right)$$

Visualisée la fonction logit



Les propriétés du logit

- La fonction logit prend une valeur comprise entre 0 et 1 et la fait correspondre à une valeur comprise entre $-\infty$ et ∞.
- La fonction logit (logistique) inverse :

$$g^{-1}(x) = \frac{\exp(x)}{1 + \exp(x)} = \frac{1}{1 + \exp(-x)}$$

- La fonction logit inverse prend une valeur comprise entre $-\infty$ et ∞ et la fait correspondre à une valeur comprise entre 0 et 1.
- Cette formulation est également utile pour l'interprétation du modèle, puisque le logit peut être interprété comme le logarithme des chances de succès.

Le modèle de régression logistique

- Sur la base des trois critères du GLM, nous avons :
  - $y_i \sim \text{Bern}(p_i)$
  - $\eta_i = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_n x_{n,i}$
  - $\text{logit}(p_i) = \eta_i$
- D'où l'on obtient :

$$p_i = \frac{\exp(\beta_0 + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i})}{1 + \exp(\beta_0 + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i})}$$

Modeling spam

Dans R, nous ajustons un GLM de la même manière qu'un modèle linéaire, sauf que nous :

- spécifions le modèle avec `logistic_reg()`
- utilisons « `glm` » au lieu de « `lm` » comme moteur
- définissons `family = "binomial"` pour la fonction de lien à utiliser dans le modèle.

```
spam_fit <- logistic_reg() %>%
  set_engine("glm") %>%
  fit(spam ~ num_char, data = email, family = "binomial")

tidy(spam_fit)
```

```
## # A tibble: 2 × 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)  -1.80     0.0716     -25.1  2.04e-139
## 2 num_char     -0.0621   0.00801     -7.75 9.50e- 15
```

Modèle de spam

```
tidy(spam_fit)
```

```
## # A tibble: 2 × 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)  -1.80     0.0716     -25.1  2.04e-139
## 2 num_char     -0.0621   0.00801     -7.75 9.50e- 15
```

Modèle :

$$\log\left(\frac{p}{1-p}\right) = -1.80 - 0.0621 \times \text{num\_char}$$

P(spam) pour un courriel de 2000 caractères

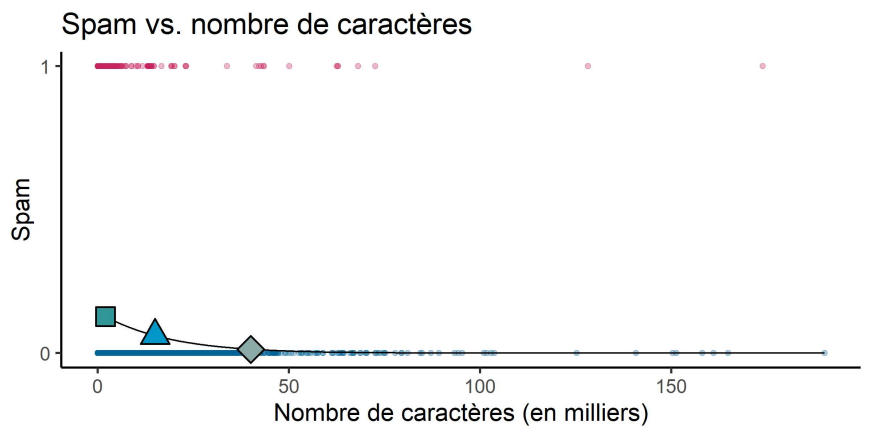
$$\log\left(\frac{p}{1-p}\right) = -1.80 - 0.0621 \times 2$$

$$\frac{p}{1-p} = \exp(-1.9242) = 0.15 \rightarrow p = 0.15 \times (1-p)$$

$$p = 0.15 - 0.15p \rightarrow 1.15p = 0.15$$

$$p = 0.15/1.15 = 0.13$$

Quelle est la probabilité qu'un courriel de 15 000 caractères soit du spam ? Qu'en est-il d'un courriel de 40000 caractères ?



- 2K chars: P(spam) = 0.13
- 15K chars, P(spam) = 0.06
- 40K chars, P(spam) = 0.01

Faire les excercise du 1-5 du Demo

Sensibilité et spécificité

Faux positifs et négatifs

	L'e-mail est un spam	l'e-mail n'est pas un spam
Email étiqueté spam	Vrai positif	faux positif (erreur de type 1)
Email étiqueté non spam	Faux négatif (erreur de type 2)	Vrai négatif

- Taux de faux négatifs = P(étiqueté non spam | Email spam) = FN / (VP + FN)
- Taux de faux positifs = P(spam étiqueté | email non spam) = FP / (FP + VN)

Sensibilité et spécificité

	L'e-mail est un spam	l'e-mail n'est pas un spam
Email étiqueté spam	Vrai positif	faux positif (erreur de type 1)
Email étiqueté non spam	Faux négatif (erreur de type 2)	Vrai négatif

- Sensibilité = P(spam étiqueté | spam par courriel) = VP / VP + FN)
- Sensibilité = 1 - taux de faux négatifs
- Spécificité = P(étiqueté non spam | courriel non spam) = VN / (FP + VN)
- Spécificité = 1 - Taux de faux positifs

> Si vous deviez concevoir un filtre anti-spam, souhaiteriez-vous que la sensibilité soit élevée ou faible ? Et la spécificité ? Quels sont les compromis associés à chaque décision ?

Prédiction

Objectif : Construire un filtre anti-spam

- Données : Ensemble d'e-mails et nous savons si chaque e-mail est un spam ou non et d'autres caractéristiques.
- Utiliser la régression logistique pour prédire la probabilité qu'un email entrant soit un spam
- Utiliser la sélection de modèle pour choisir le modèle ayant la meilleure performance prédictive
- La construction d'un modèle permettant de prédire la probabilité qu'un courriel soit du spam ne représente que la moitié de la bataille ! Nous avons également besoin d'une règle de décision pour déterminer quels courriels sont marqués comme spam (par exemple, quelle probabilité devrions-nous utiliser comme critère d'exclusion ?)
- Une approche simple : choisir un seuil de probabilité unique et tout courriel qui dépasse cette probabilité est considéré comme du spam.

Une approche de régression multiple

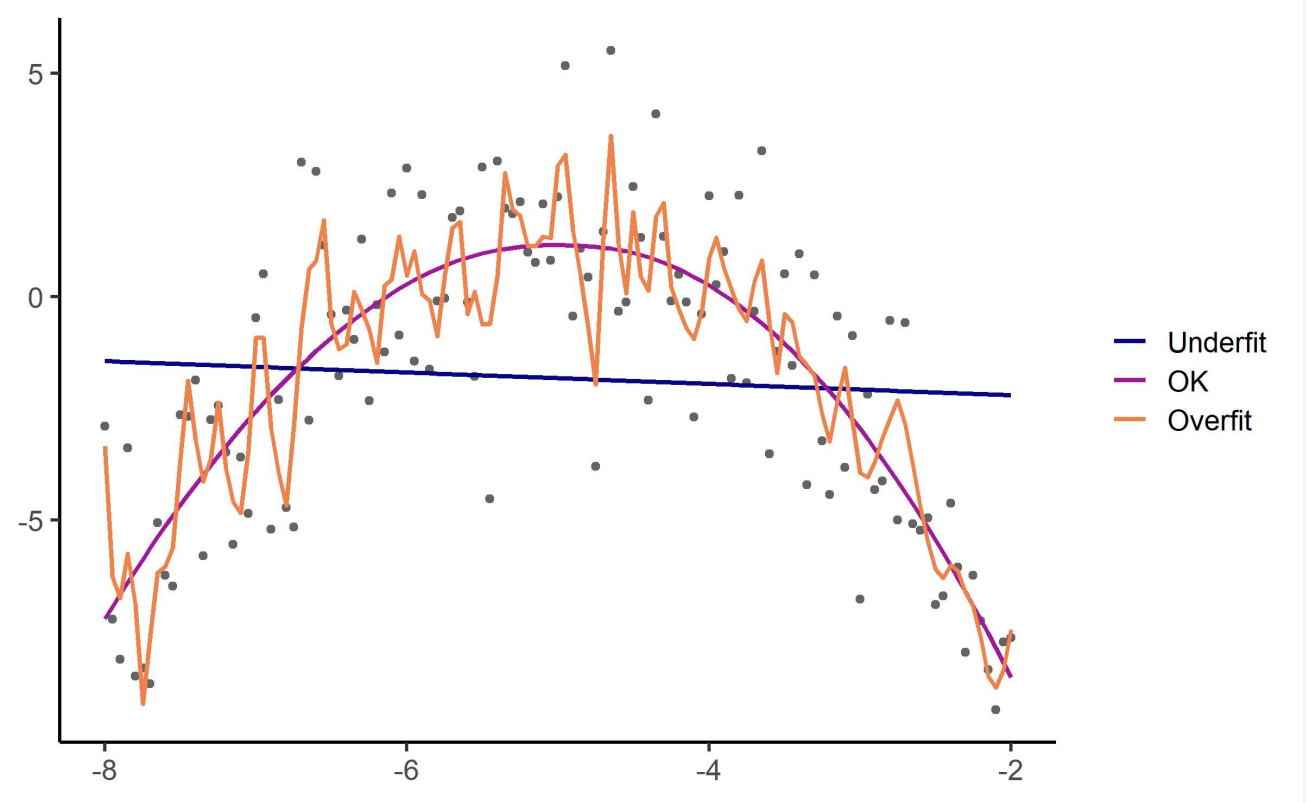
Output | Code

```
## # A tibble: 22 × 5
##    term         estimate std.error statistic  p.value
##    <chr>           <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)  -9.09e+1   9.80e+3  -0.00928 9.93e- 1
##  2 to_multiple1 -2.68e+0   3.27e-1  -8.21    2.25e-16
##  3 from1        -2.19e+1   9.80e+3  -0.00224 9.98e- 1
##  4 cc            1.88e-2   2.20e-2   0.855   3.93e- 1
##  5 sent_email1  -2.07e+1   3.87e+2  -0.0536  9.57e- 1
##  6 time          8.48e-8   2.85e-8   2.98    2.92e- 3
##  7 image        -1.78e+0   5.95e-1  -3.00    2.73e- 3
##  8 attach        7.35e-1   1.44e-1   5.09    3.61e- 7
##  9 dollar       -6.85e-2   2.64e-2  -2.59    9.64e- 3
## 10 winneryes     2.07e+0   3.65e-1   5.67    1.41e- 8
## 11 inherit       3.15e-1   1.56e-1   2.02    4.32e- 2
## 12 viagra        2.84e+0   2.22e+3   0.00128 9.99e- 1
## 13 password     -8.54e-1   2.97e-1  -2.88    4.03e- 3
## 14 num_char      5.06e-2   2.38e-2   2.13    3.35e- 2
## 15 line_breaks  -5.49e-3   1.35e-3  -4.06    4.91e- 5
## 16 format1      -6.14e-1   1.49e-1  -4.14    3.53e- 5
## 17 re_subj1     -1.64e+0   3.86e-1  -4.25    2.16e- 5
## 18 exclaim_subj  1.42e-1   2.43e-1   0.585   5.58e- 1
## 19 urgent_subj1  3.88e+0   1.32e+0   2.95    3.18e- 3
## 20 exclaim_mess  1.08e-2   1.81e-3   5.98    2.23e- 9
## 21 numbersmall  -1.19e+0   1.54e-1  -7.74    9.62e-15
## 22 numberbig    -2.95e-1   2.20e-1  -1.34    1.79e- 1
```

Prédiction

- La mécanique de la prédiction est **facile** :
- Insérer les valeurs des prédicteurs dans l'équation du modèle
- Calculer la valeur prédite de la variable réponse, $\hat{y}$.
- Il est **difficile** d'obtenir de bons résultats !
- Il n'y a aucune garantie que les estimations du modèle que vous avez soient correctes
- ou que votre modèle fonctionnera aussi bien avec les nouvelles données qu'avec les données de votre échantillon.

Underfitting et overfitting



- Plusieurs étapes pour créer un modèle utile : estimation des paramètres, sélection du modèle, évaluation des performances, etc.
- Faire tout cela sur l'ensemble des données dont nous disposons peut conduire à un **overfitting**.
- Affecter des sous-ensembles spécifiques de données à différentes tâches, plutôt que d'affecter la plus grande quantité possible à la seule estimation des paramètres du modèle (ce que nous avons fait jusqu'à présent).

Fractionnement des données

Données de fractionnement

- **Ensemble d'entraînement:**
- Sandbox pour l'élaboration du modèle
- Passez la majeure partie de votre temps à utiliser l'ensemble d'entraînement pour développer le modèle.

- La majorité des données (généralement 80 %)

- **Ensemble de test :**

- Gardé en réserve pour déterminer l'efficacité d'un ou deux modèles choisis
- Il est essentiel de l'examiner une fois, sinon il devient partie intégrante du processus de modélisation.
- Reste des données (généralement 20 %)

Réalisation du fractionnement

```
# Fix random numbers by setting the seed
# Enables analysis to be reproducible when random numbers are used
set.seed(1116)

# Put 80% of the data into the training set
email_split <- initial_split(email, prop = 0.80)

# Create data frames for the two sets:
train_data <- training(email_split)
test_data  <- testing(email_split)
```

Coup d'œil sur la division

```
glimpse(train_data)
```

```
## Rows: 3,136
## Columns: 21
## $ spam         <fct> 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, …
## $ to_multiple  <fct> 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, …
## $ from         <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
## $ cc           <int> 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 35,…
## $ sent_email   <fct> 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ time         <dttm> 2012-01-25 17:46:55, 2012-01-03 00:28:28,…
## $ image        <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ attach       <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ dollar       <dbl> 10, 0, 0, 0, 0, 0, 13, 0, 0, 0, 2, 0, 0, 0…
## $ winner       <fct> no, no, no, no, no, no, no, yes, no, no, n…
## $ inherit      <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ viagra       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ password     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ num_char     <dbl> 23.308, 1.162, 4.732, 42.238, 1.228, 25.59…
## $ line_breaks  <int> 477, 2, 127, 712, 30, 674, 367, 226, 98, 6…
## $ format       <fct> 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, …
## $ re_subj      <fct> 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, …
## $ exclaim_subj <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, …
## $ urgent_subj  <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ exclaim_mess <dbl> 12, 0, 2, 2, 2, 31, 2, 0, 0, 1, 0, 1, 2, 0…
## $ number       <fct> small, none, big, big, small, small, small…
```

```
glimpse(test_data)
```

```
## Rows: 785
## Columns: 21
## $ spam         <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ to_multiple  <fct> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, …
## $ from         <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
## $ cc           <int> 0, 1, 0, 1, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ sent_email   <fct> 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ time         <dttm> 2012-01-01 12:55:06, 2012-01-01 14:38:32,…
## $ image        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ attach       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, …
## $ dollar       <dbl> 0, 0, 5, 0, 0, 0, 0, 5, 4, 0, 0, 0, 21, 0,…
## $ winner       <fct> no, no, no, no, no, no, no, no, no, no, no…
## $ inherit      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, …
## $ viagra       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ password     <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, …
## $ num_char     <dbl> 4.837, 15.075, 18.037, 45.842, 11.438, 1.4…
## $ line_breaks  <int> 193, 354, 345, 881, 125, 24, 296, 13, 192,…
## $ format       <fct> 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, …
## $ re_subj      <fct> 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, …
## $ exclaim_subj <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ urgent_subj  <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ exclaim_mess <dbl> 1, 10, 20, 5, 2, 0, 0, 0, 6, 0, 0, 1, 3, 0…
## $ number       <fct> big, small, small, big, small, none, small…
```

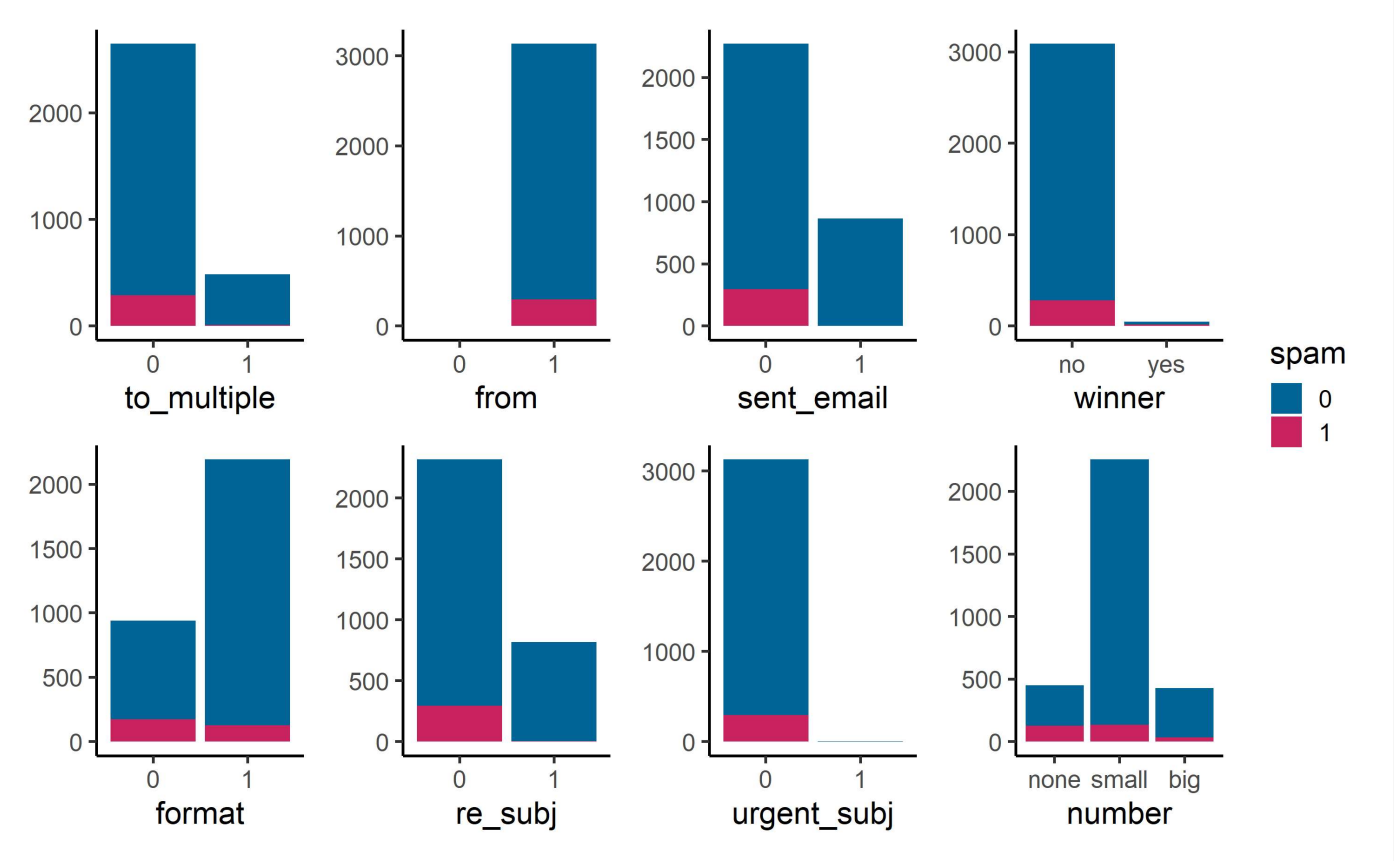
Flux de travail de la modélisation

Ajuster un modèle à l'ensemble de données d'entraînement

```
email_fit <- logistic_reg() %>%
  set_engine("glm") %>%
  fit(spam ~ ., data = train_data, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

Prédicteurs catégoriels



`from` et `sent_email`

- `from`: Si le message a été répertorié comme provenant de quelqu'un d'autre (cette option est généralement définie par défaut pour les courriers électroniques sortants).

```
train_data %>%
  count(spam, from)
```

```
## # A tibble: 3 × 3
##   spam  from      n
##   <fct> <fct> <int>
## 1 0     1      2837
## 2 1     0         3
## 3 1     1       296
```

- `sent_email`: Indicateur permettant de savoir si l'expéditeur a reçu un courrier électronique au cours des 30 derniers jours.

```
train_data %>%
  count(spam, sent_email)
```

```
## # A tibble: 3 × 3
##   spam  sent_email     n
##   <fct> <fct>      <int>
## 1 0     0           1972
## 2 0     1            865
## 3 1     0            299
```

Les prédicteurs numériques

```
##
## ── Variable type: numeric ──────────────────────────────────────────────────────
##    skim_variable spam n_missing complete_rate      mean        sd    p0    p25     p50    p75  p100
##  1 cc            0            0             1   0.393     2.62    0      0       0       0      68
##  2 cc            1            0             1   0.388     3.25    0      0       0       0      50
##  3 image         0            0             1   0.0536    0.503   0      0       0       0      20
##  4 image         1            0             1   0.00334   0.0578  0      0       0       0       1
##  5 attach        0            0             1   0.124     0.775   0      0       0       0      21
##  6 attach        1            0             1   0.227     0.620   0      0       0       0       2
##  7 dollar        0            0             1   1.56      5.33    0      0       0       0      64
##  8 dollar        1            0             1   0.779     3.01    0      0       0       0      36
##  9 inherit       0            0             1   0.0352    0.216   0      0       0       0       6
## 10 inherit       1            0             1   0.0702    0.554   0      0       0       0       9
## 11 viagra        0            0             1   0         0       0      0       0       0       0
## 12 viagra        1            0             1   0.0268    0.463   0      0       0       0       8
## 13 password      0            0             1   0.112     0.938   0      0       0       0      22
## 14 password      1            0             1   0.0201    0.182   0      0       0       0       2
## 15 num_char      0            0             1  11.4      14.9     0.003  1.97    6.83   15.7   190.
## 16 num_char      1            0             1   5.63     15.7     0.001  0.468   0.999   3.55  174.
## 17 line_breaks   0            0             1 247.      326.      2     42     138     318    4022
## 18 line_breaks   1            0             1 108.      321.      1     14      23      66.5  3729
## 19 exclaim_subj  0            0             1   0.0783    0.269   0      0       0       0       1
## 20 exclaim_subj  1            0             1   0.0769    0.267   0      0       0       0       1
## 21 exclaim_mess  0            0             1   6.68     50.2     0      0       1       5    1236
## 22 exclaim_mess  1            0             1   8.75     88.4     0      0       0       1    1209
```

Ajuster un modèle à l'ensemble de données d'entraînement

```
email_fit <- logistic_reg() %>%
  set_engine("glm") %>%
  fit(spam ~ . - from - sent_email - viagra, data = train_data, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
email_fit
```

```
## parsnip model object
##
##
## Call:  stats::glm(formula = spam ~ . - from - sent_email - viagra, family = stats::binomial,
##     data = data)
##
## Coefficients:
##  (Intercept)  to_multiple1            cc          time         image        attach        dollar
##   -9.867e+01    -2.505e+00     1.944e-02     7.396e-08    -2.854e+00     5.070e-01    -6.440e-02
##    winneryes       inherit      password      num_char   line_breaks       format1      re_subj1
##    2.170e+00     4.499e-01    -7.065e-01     5.870e-02    -5.420e-03    -9.017e-01    -2.995e+00
## exclaim_subj  urgent_subj1  exclaim_mess   numbersmall     numberbig
##    1.002e-01     3.572e+00     1.009e-02    -8.518e-01    -1.329e-01
##
## Degrees of Freedom: 3135 Total (i.e. Null);  3117 Residual
## Null Deviance:        1974
## Residual Deviance: 1447     AIC: 1485
```

Prédire les résultats sur l'ensemble des données de test

```
predict(email_fit, test_data)
```

```
## # A tibble: 785 × 1
##   .pred_class
##   <fct>
## 1 0
## 2 0
## 3 0
## 4 0
## 5 0
## 6 0
## # ℹ 779 more rows
```

Prédire les probabilités sur l'ensemble de données de test

```
email_pred <- predict(email_fit, test_data, type = "prob") %>%
  bind_cols(test_data %>% select(spam, time))

email_pred
```

```
## # A tibble: 785 × 4
##   .pred_0 .pred_1 spam  time
##     <dbl>   <dbl> <fct> <dttm>
## 1   0.993 0.00709 0     2012-01-01 12:55:06
## 2   0.998 0.00181 0     2012-01-01 14:38:32
## 3   0.981 0.0191  0     2012-01-02 00:42:16
## 4   0.999 0.00124 0     2012-01-02 10:12:51
## 5   0.988 0.0121  0     2012-01-02 11:45:36
## 6   0.830 0.170   0     2012-01-02 16:55:03
## # ℹ 779 more rows
```

Un examen plus approfondi des prévisions

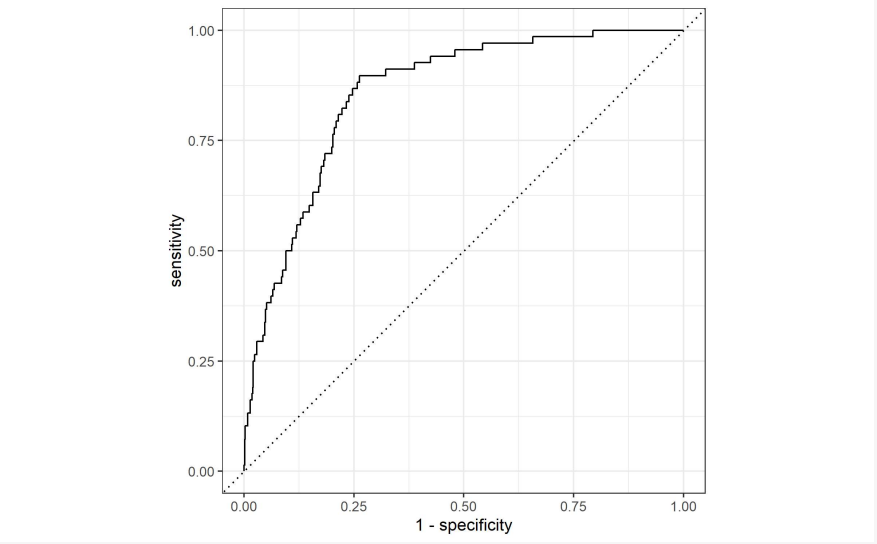
```
email_pred %>%
  arrange(desc(.pred_1)) %>%
  print(n = 10)
```

```
## # A tibble: 785 × 4
##    .pred_0 .pred_1 spam  time
##      <dbl>   <dbl> <fct> <dttm>
##  1  0.0972   0.903 1     2012-02-13 07:15:00
##  2  0.167    0.833 0     2012-01-27 15:05:06
##  3  0.175    0.825 1     2012-03-01 00:40:27
##  4  0.267    0.733 1     2012-03-17 06:13:27
##  5  0.317    0.683 1     2012-03-21 08:33:12
##  6  0.374    0.626 1     2012-02-08 03:00:05
##  7  0.386    0.614 0     2012-01-30 09:20:29
##  8  0.403    0.597 1     2012-01-07 11:11:49
##  9  0.462    0.538 1     2012-03-06 06:46:20
## 10  0.463    0.537 0     2012-02-17 17:54:16
## # ℹ 775 more rows
```

Évaluer la performance

La courbe de receiver operating characteristic (ROC)[+] qui représente le taux de vrais positifs vs. le taux de faux positifs (1 - spécificité)

```
email_pred %>%
  roc_curve(
    truth = spam,
    .pred_1,
    event_level = "second"
  ) %>%
  autoplot()
```



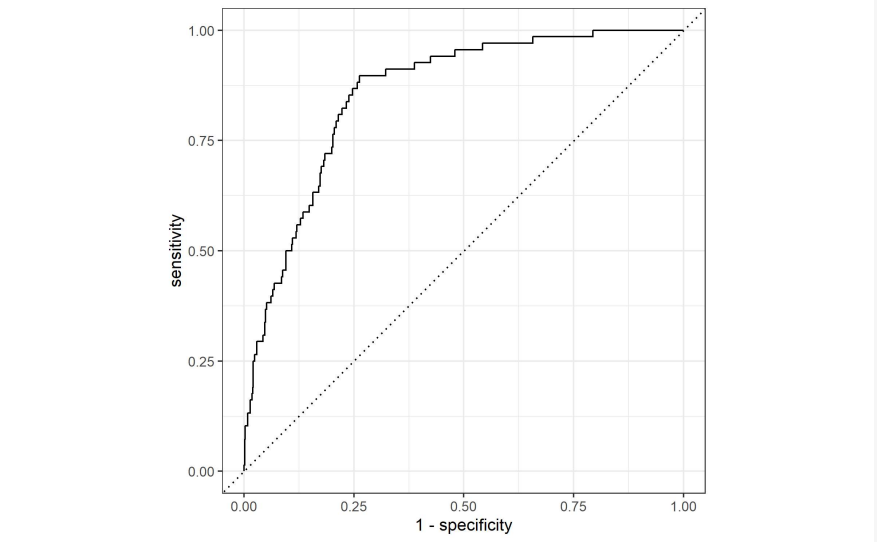
[+] Développé à l'origine pour les opérateurs de récepteurs radar militaires, d'où son nom.

Évaluer la performance

Trouver l'aire sous la courbe :

```
email_pred %>%
  roc_auc(
    truth = spam,
    .pred_1,
    event_level = "second"
  )
```

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.857
```



Compléter les exercices du Demo.