Reporte MARS

Adrian Tame, Miguel Calvo, Nelson Gil

11/28/2021

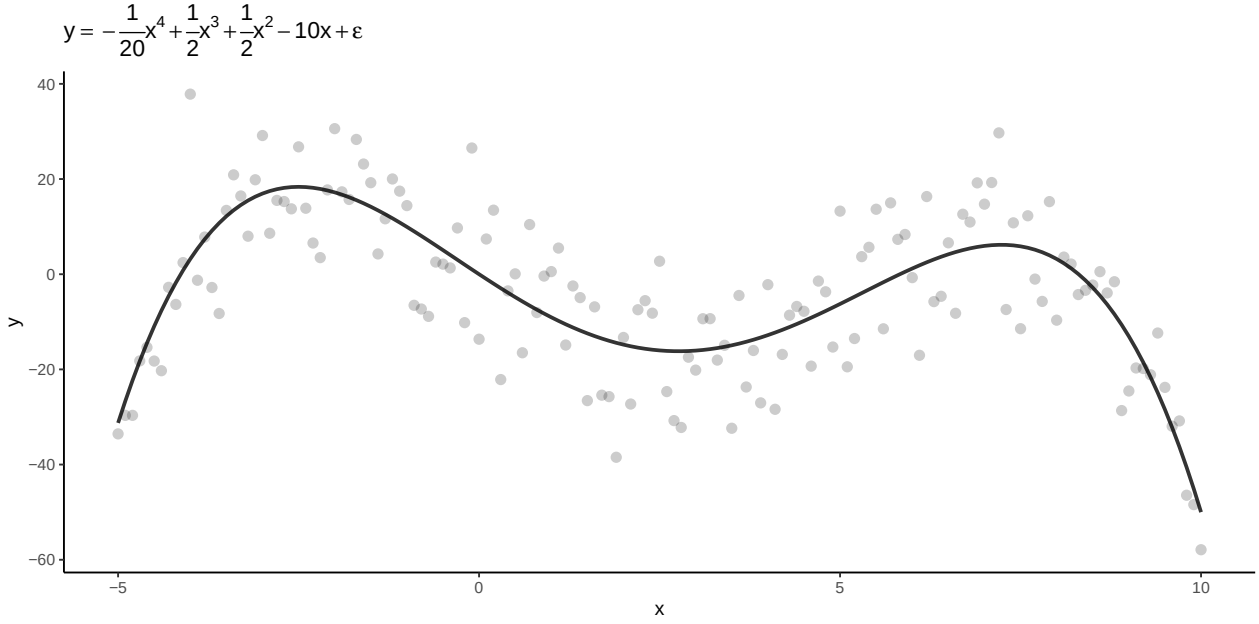
Multivariate Adaptive Regression Splines (MARS)

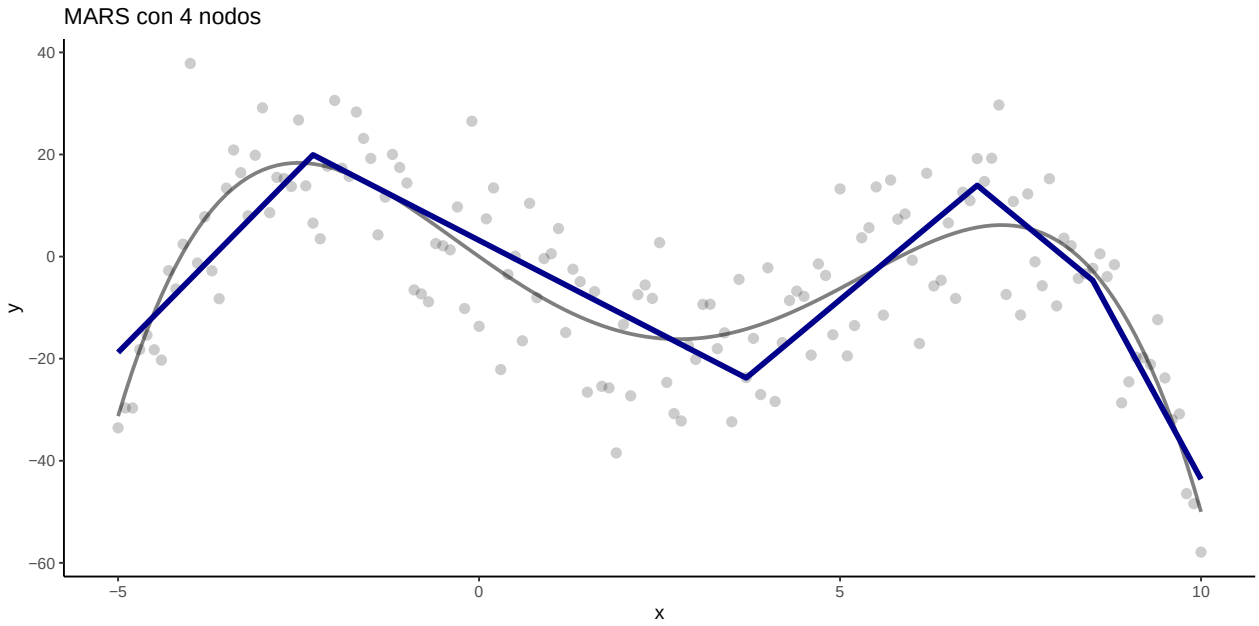
Introducción

El modelo MARS es un algoritmo de aprendizaje supervisado que funciona para regresión. En términos simples, es una versión modificada de regresión lineal, pero el modelo es más complicado ya que permite hacer diferentes pendientes para diferentes partes de la variable a estimar, y automaticamente modela términos no lineales e interacciones entre variables. Fue introducido originalmente en [1] por Friedman en 1991, y existen varias implementaciones del algoritmo, generalmente bajo el nombre de "Earth" [CITA] [CITA].

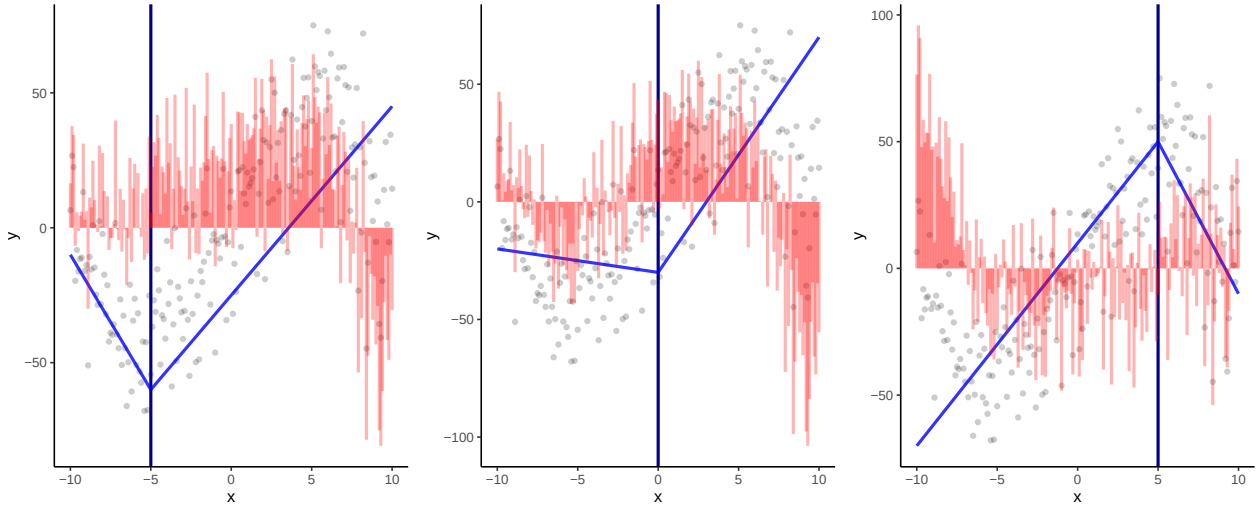
Comportamiento del Modelo

En términos simples, podemos considerar a MARS como un modelo de regresión lineal a pedazos. Como se aprecia en el ejemplo de abajo, MARS busca los puntos de corte y las pendientes óptimas para aproximar a la variable objetivo:

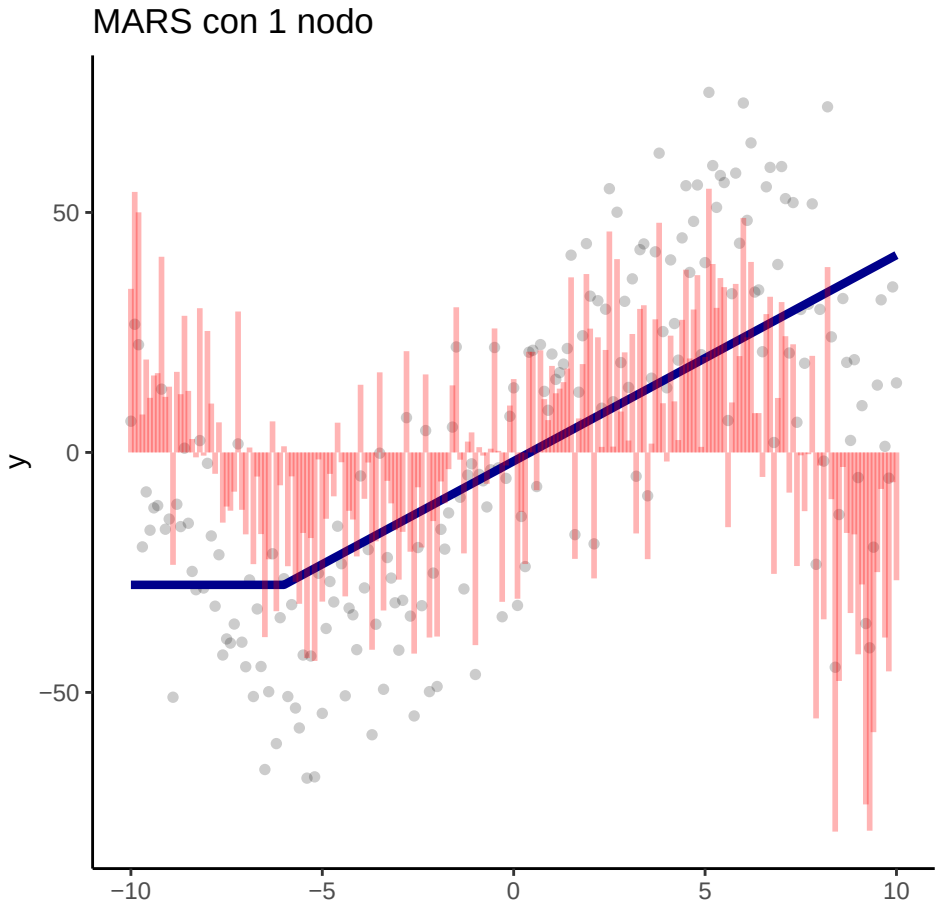




Como podemos ver, de darnos a la tarea de encontrar manualmente cada punto de corte y pendientes se puede traducir a minimizar alguna métrica como el error cuadrático medio:



Dados los valores anteriores, vemos el ajuste realizado por MARS:



Descripción del Modelo

MARS es un modelo de regresión no paramétrico que a través de funciones bisagra (*hinge functions*). Se puede considerar como una modificación al método CART con el fin de mejorar su desempeño en el contexto de regresión Es un modelo que es bueno para problemas de alta dimensionalidad. Para la siguiente sección, usamos la terminología y seguimos el desarrollo que se presenta en [1] y en [2].

MARS se define con el uso de funciones que son lineales por partes, de la forma $(x-t)_+$ y $(t-x)_+$. Estas funciones toman el máximo entre 0 y el valor dentro de la funcion, por lo tanto,

$$(x-t)_+ = \max\{0, x-t\} = \begin{cases} x-t & \text{si } x > t, \\ 0 & \text{en otro caso.} \end{cases}$$

Cada función es lineal a trozos con un cambio de pendiente en el valor t, lo cual las hace splines lineales, y a cada par dividido en el valor t se le llama un *par reflejado*.

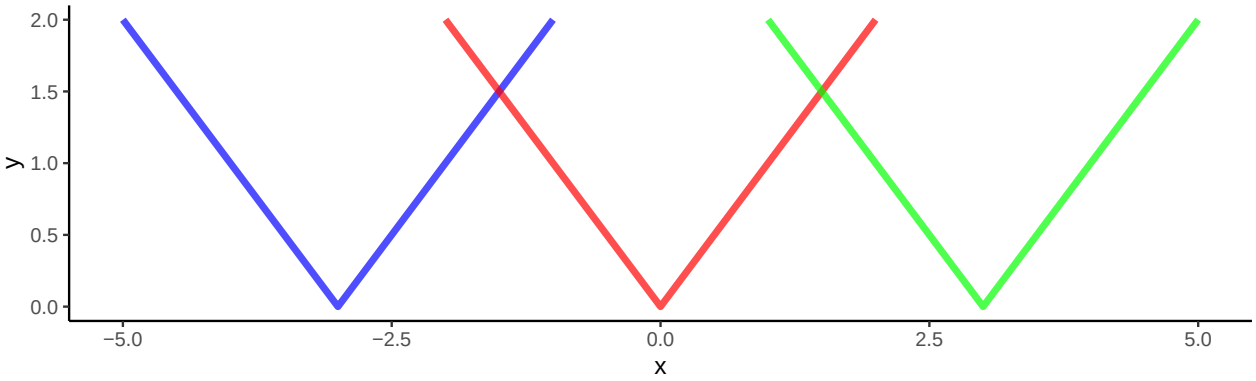
La idea del método es formar pares reflejados para cada variable de entrada X_j con cambios de pendiente en el spline en cada valor observado x_{ij}. Por lo tanto, para cada variable de entrada o variable predictiva, se define un spline distinto. Por lo tanto, para una variable X_j, la colección de funciones base es

$$\mathcal{C} = \{(X_j - t)_+, (t - X_j)_+\}, \text{ con } t \in \{x_{1,j}, x_{2,j}, \ldots x_{N,j}\}, \text{ y } \ j = 1, 2, \ldots, p.$$

Esto significa que si todos los valores de entrada son distintos, se tienen $2Np$ funciones base en total, y $2N$ divisiones en cada uno de los splines para cada variable. El modelo que utilizamos entonces para juntar todas las variables es uno aditivo, de forma parecida que se hace con regresión lineal:

$$f(X) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(X),$$

donde cada función $h_m(X)$ es elemento de $\mathcal{C}$ o una combinación lineal de estas funciones base, pero para esta explicación inicial, usaremos solamente funciones que no involucran interacciones. Este modelo sin interacciones funciona de forma muy parecida a los árboles de decisión CART. Visualmente, se ve algo parecido a la siguiente gráfica cada uno de los pares reflejados que se usan (notamos que estas están en $x = 0$ por facilidad de visualización):

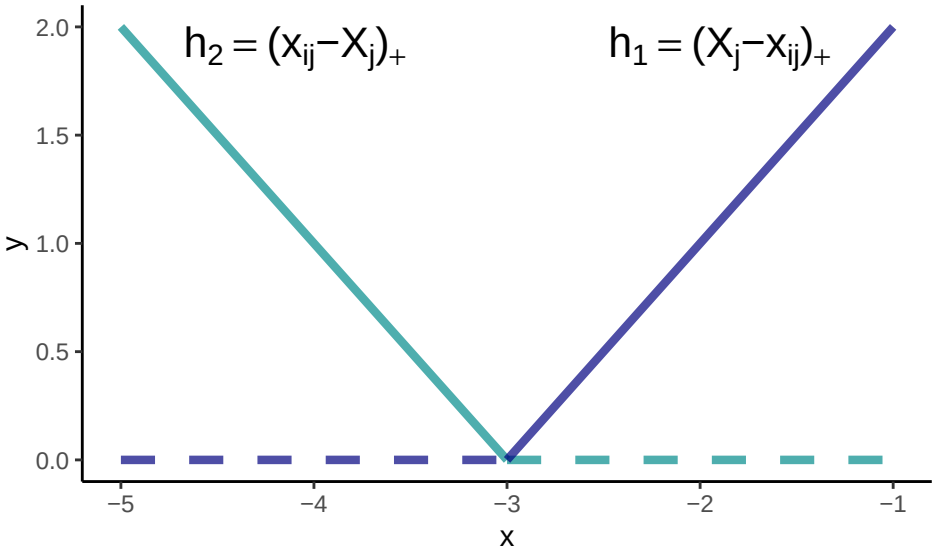


El proceso de construcción del modelo (denotado $\mathcal{M}$) empieza con el modelo base $\hat{f}(X) = \hat{\beta}_0 = h_0(X) = 1$, donde se define un término que funciona como intercepto al orígen, y se hace un procedimiento que se llama *forward*. Después de esto, todas las funciones en el conjunto $\mathcal{C}$ son candidatas de entrada a $\mathcal{M}$. Para cada observación x_{ij}, se genera un punto de corte descrito por un par reflejado:

$$h_1(X) = h(X_j - x_{ij})$$

$$h_2(X) = h(x_{i,j} - X_j)$$

en la siguiente gráfica, podemos ver una representación de como se verían estas funciones:



y con esto, se ajusta el nuevo modelo involucrando estas funciones:

$$\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 h_1(X) + \hat{\beta}_2 h_2(X).$$

Como esto es una forma lineal en términos de cada una de las funciones h_i, se hace el ajuste de cada parámetro $\hat{\beta}_i$ minimizando la suma de cuadrados.

En cada etapa de este procedimiento, se van agregando pares reflejados para cada una de las observaciones que se tienen, y eventualmente, se llega a un modelo final que incluye todos los posibles cortes definidos por

cada observación. Claramente, esto llevará a sobreajuste de los datos, pero esto es lo que se está buscando en esta parte del procedimiento. Ya teniendo $\mathcal{M}$ con todas las interacciones posibles, se empieza la segunda parte del ajuste del modelo, que es la parte de podarlo. En este caso, se van eliminando los términos $h_i(X)$ iterativamente, empezando por el que produce el menor incremento en el error cuadrático residual cuando se quita. Este procedimiento produce un mejor modelo para cada tamaño λ, donde este modelo lo denotamos $\hat{f}_\lambda$.

Hay varios procedimientos que se pueden utilizar para estimar el valor óptimo de λ, como validación cruzada o bootstrap, y probablemente el mejor de ellos sea análisis de *leave one out*, pero esto involucra un gran costo computacional. Para minimizar este problema de costo computacional, los modelos MARS generalmente utilizan un procedimiento de validación cruzada generalizada (GCV, por sus siglas en inglés). Este criterio se define como:

$$GCV(\lambda) = \frac{\sum_{i=1}^{N}(y_i - \hat{f}_\lambda(x_i))^2}{\left(\frac{1-M(\lambda)}{N}\right)^2},$$

donde el valor $M(\lambda)$ es el número de parámetros *efectivos* en el modelo, que depende del número de términos más el número de puntos de corte utilizados penalizado por un factor (2 en el caso aditivo que estamos explicando, 3 cuando hay interacciones).

Ya que describimos el modelo base, podemos regresar y considerar cuales serían las diferencias al incorporar términos de interacción. Si se tiene el modelo base sin podar, podemos seguir iterando y agregando términos, pero esta vez, se considera como nueva función base todos los productos de una cierta función h_m en el modelo $\mathcal{M}$ con uno de los pares reflejados que definimos en $\mathcal{C}$. Estos se agregan al modelo de la siguiente forma:

$$\hat{f}(X) = \hat{\beta}_0 + \sum_{m=1}^{M} \hat{\beta}_m h_m(X) + \hat{\beta}_{M+1} h_\ell(X) \cdot (X_j - t)_+ + \hat{\beta}_{M+2} h_\ell(X) \cdot (t - X_j)_+,$$

para cada $h_\ell \in \mathcal{M}$, y donde $t = x_{ij}$ cualquiera. Otra manera de ver esto es que el modelo base es interactuar cada una de los pares reflejados para una variable x_{ij} con el término incial del modelo, $h_0 = \hat{\beta}_0 = 1$.

Implementación

Para nuestra implementación, utilizamos una base de datos `spam`, tomada de [3]. Es una colección de palabras y caracteres que aparecen comúnmente en mensajes que son *spam*. La variable respuesta es una variable categórica que tiene dos niveles: `spam`, `no_spam`. Más información se puede ver en [4].

Presentamos una table de algunas de las variables predictoras y la variable respuesta.

```
## # A tibble: 6 x 5
##   wfyour wfaddress wfemail crlaverage spam
##    <dbl>     <dbl>   <dbl>      <dbl> <fct>
## 1   1.15      0.57    1.73       1.42 spam
## 2   1.24      0.41    0.82       3.17 spam
## 3   4.87      0       0          1    no_spam
## 4   0.48      0       0          3.32 spam
## 5   5.97      0       0          2.35 spam
## 6   0         0       0          1.94 no_spam
```

Podemos hacer predicciones con este modelo. Por ejemplo, en entrenamiento tenemos las predicciones de clase dan:

```
## # A tibble: 4 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary          0.96
## 2 sens     binary          0.97
## 3 spec     binary          0.94
```

```
## 4 roc_auc  binary          0.99
```

y en prueba:

```
## # A tibble: 4 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary          0.94
## 2 sens     binary          0.96
## 3 spec     binary          0.92
## 4 roc_auc  binary          0.98
```

Y notamos el ajuste entre prueba y entrenamiento, es bastante consistente por lo que tenemos un buen ajuste, bajo la métrica de `binary`, que es pérdida logarítmica. En la siguiente tabla, podemos ver los coeficientes que se usan para este modelo final (los primeros 10). Notamos que muchos de ellos son interacciones de variables, como se explicó en la descripción del modelo. El valor bajo `spam` es el coeficiente asociado β_i.

```
##                                          spam
## (Intercept)                         0.199122835
## h(0.257-cfexc)                     -0.245063537
## h(0.088-cfdollar)                   7.809551831
## h(wfremove-0.29)*h(0.088-cfdollar)  0.627252683
## h(0.29-wfremove)*h(0.088-cfdollar) -12.356605563
## h(wffree-0.41)*h(0.088-cfdollar)    0.238958028
## h(0.41-wffree)*h(0.088-cfdollar)   -4.768347722
## h(0.52-wfhp)*h(216-crltotal)       -0.001157403
## h(0.52-wfhp)*h(0.44-wfedu)          1.012531128
## h(0.52-wfhp)*h(wfgeorge-0.08)       0.016298821
```

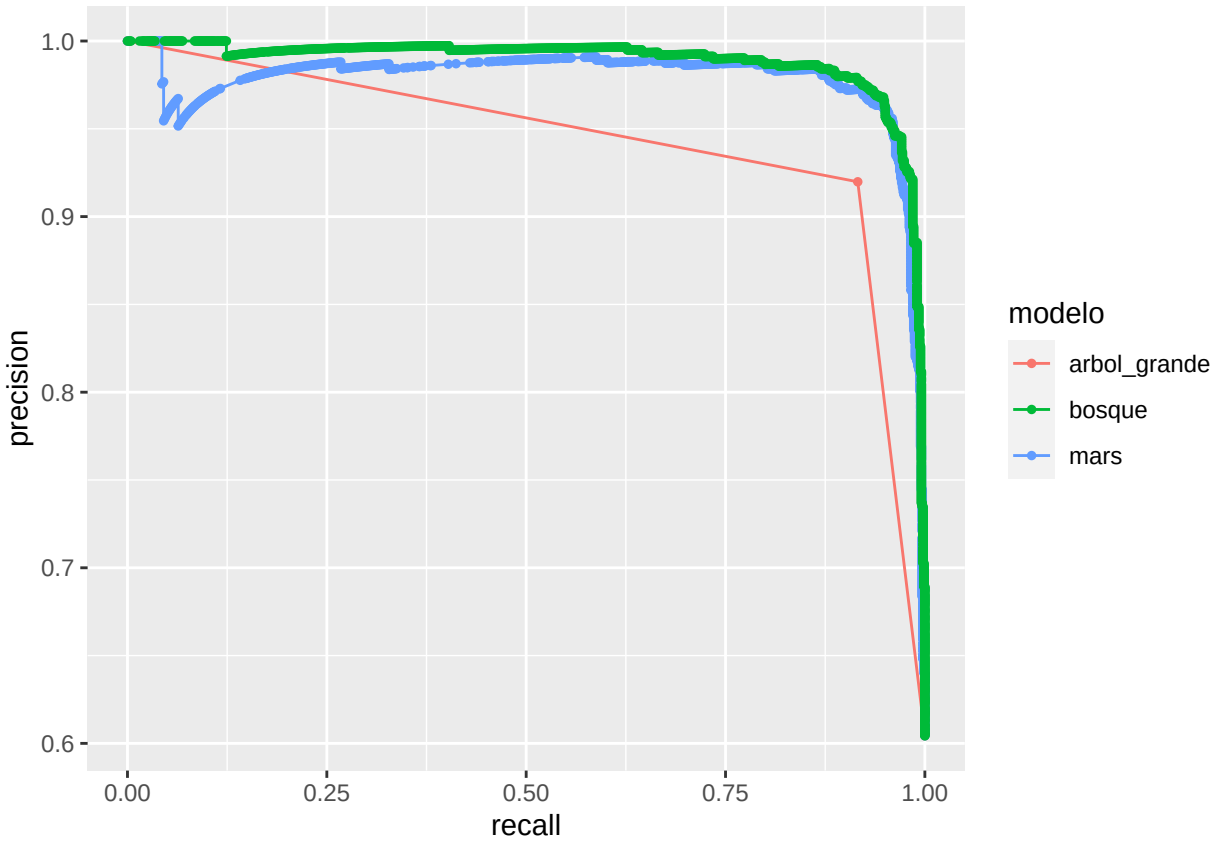
Comparación con Árboles y Bosques Aleatorios

Ajustamos otros dos modelos, en este caso un árbol de decisiones y bosques aleatorios, y podemos comparar el ajuste de MARS contra este nuevo modelo. Estas implementacióoes para estos datos la tomamos de [5].

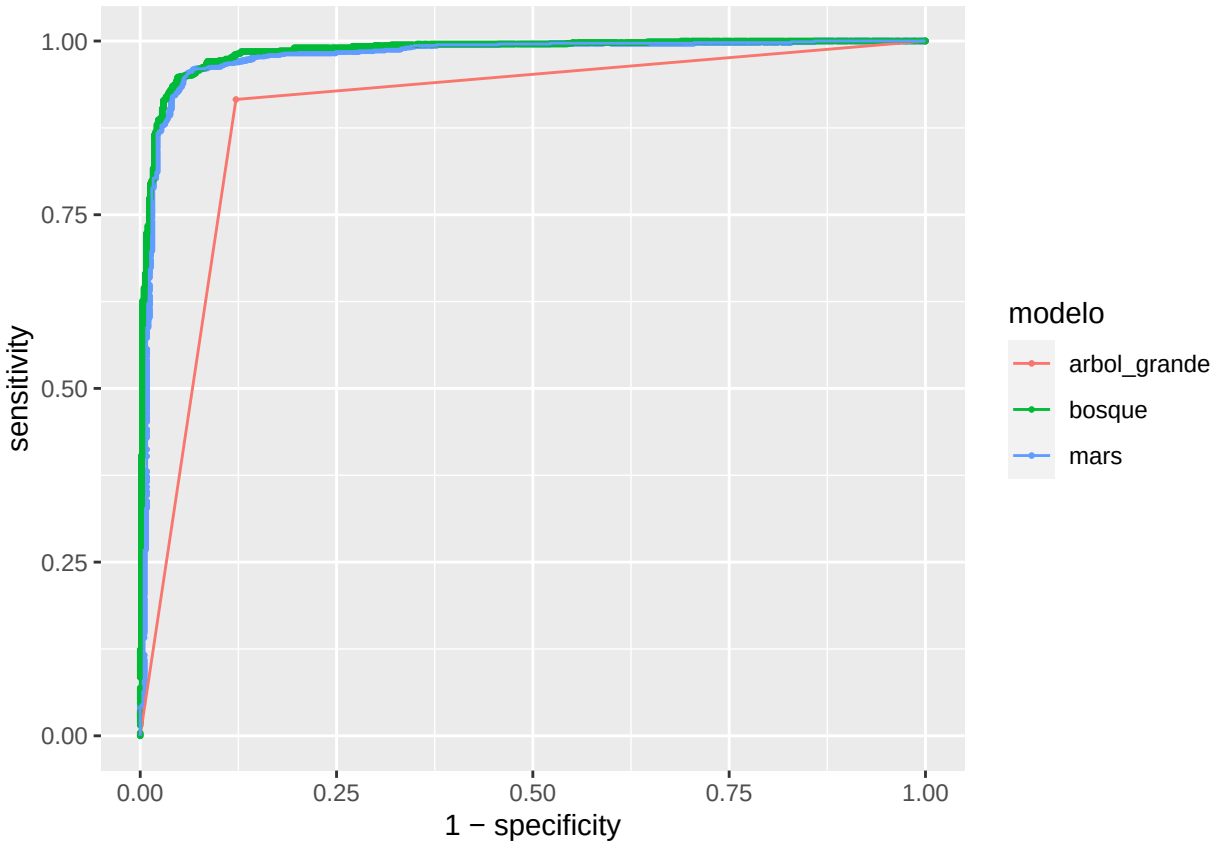
```
## # A tibble: 4 x 5
##   .metric  .estimator estimate_mars estimate_arboles estimate_bosques
##   <chr>    <chr>              <dbl>            <dbl>            <dbl>
## 1 accuracy binary              0.94             0.9              0.95
## 2 sens     binary              0.96             0.92             0.97
## 3 spec     binary              0.92             0.88             0.91
## 4 roc_auc  binary              0.98             0.9              0.98
```

Curvas Precision-Recall y ROC

Podemos visualizar esta diferencia con una gráfica de precision-recall:



O las curvas ROC:



En el caso de comparar con otros modelos, vemos desempeño comparable o un poco peor del modelo MARS contra el de bosques aleatorios.

Ventajas y Desventajas

Conclusiones

Spam malo porque clasificación.

Referencias

- [1] Friedman, J. H. (1991). "Multivariate Adaptive Regression Splines". *The Annals of Statistics*. **19** (1): 1–67.

https://projecteuclid.org/journals/annals-of-statistics/volume-19/issue-1/Multivariate-Adaptive-Regression-Splines/10.1214/aos/1176347963.full

https://towardsdatascience.com/mars-multivariate-adaptive-regression-splines-how-to-improve-on-linear-regression-e1e7a63c5eae

https://en.wikipedia.org/wiki/Multivariate_adaptive_regression_spline

[2] https://rubenfcasal.github.io/aprendizaje_estadistico/mars.html

https://web.stanford.edu/~hastie/Papers/ESLII.pdf

[3] https://archive.ics.uci.edu/ml/datasets/spambase

[4] https://lorrie.cranor.org/pubs/spam/

[5] https://aprendizaje-maquina-2021-mcd.netlify.app/m%C3%A9todos-basados-en-%C3%A1rboles.html