# Multivariate Adaptive Regression Splines (MARS)

Adrian Tame Jacobo, Miguel Calvo Valente, Nelson Gil Vargas

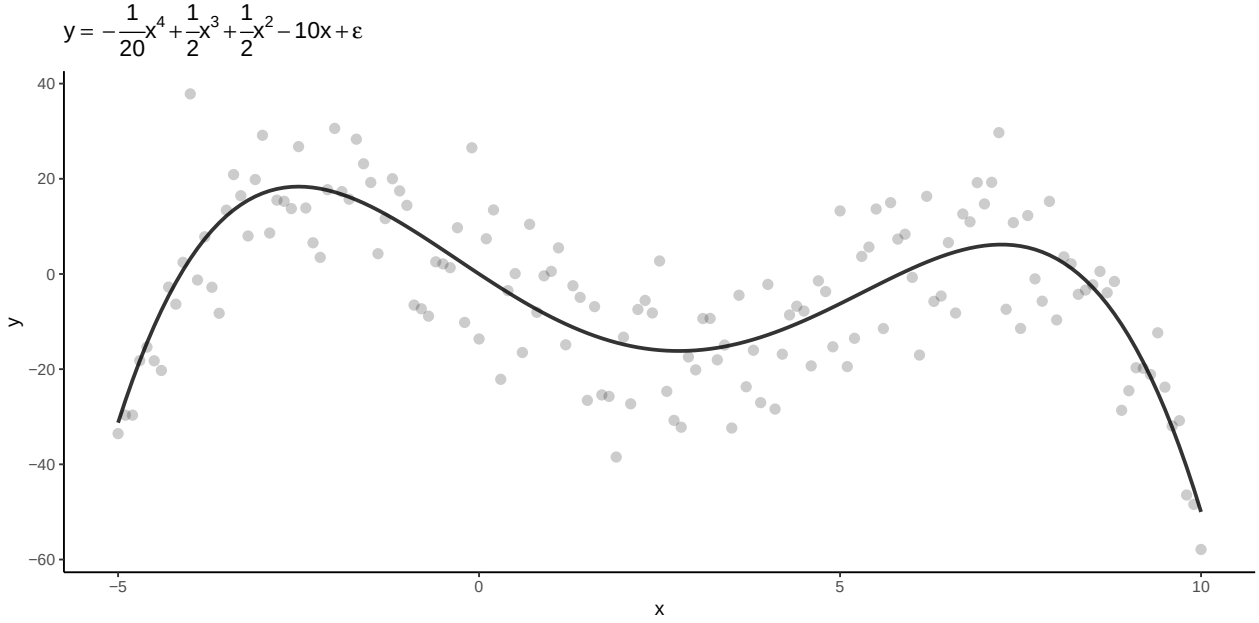
11/28/2021

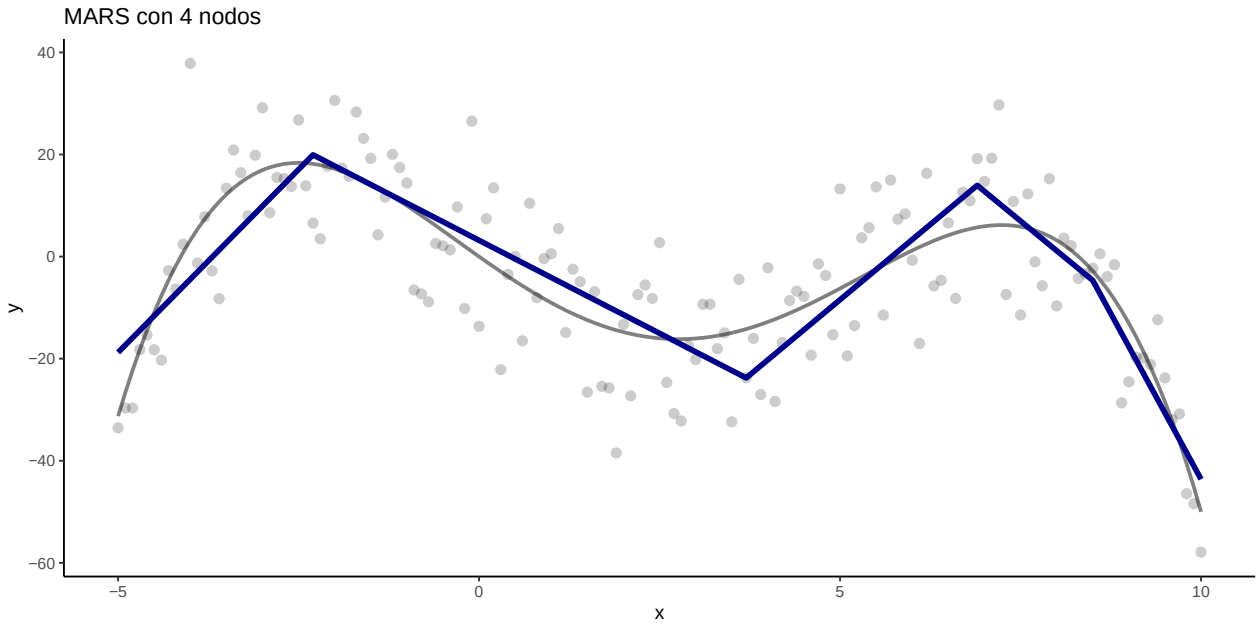
## Introducción

El modelo MARS es un algoritmo de aprendizaje supervisado que funciona para regresión y clasificación. Es una versión generalizada de regresión lineal a pedazos [6], en el cual el modelo permite hacer diferentes pendientes para diferentes partes de la variable a estimar en función de las variables predictoras, y automáticamente 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" [8].

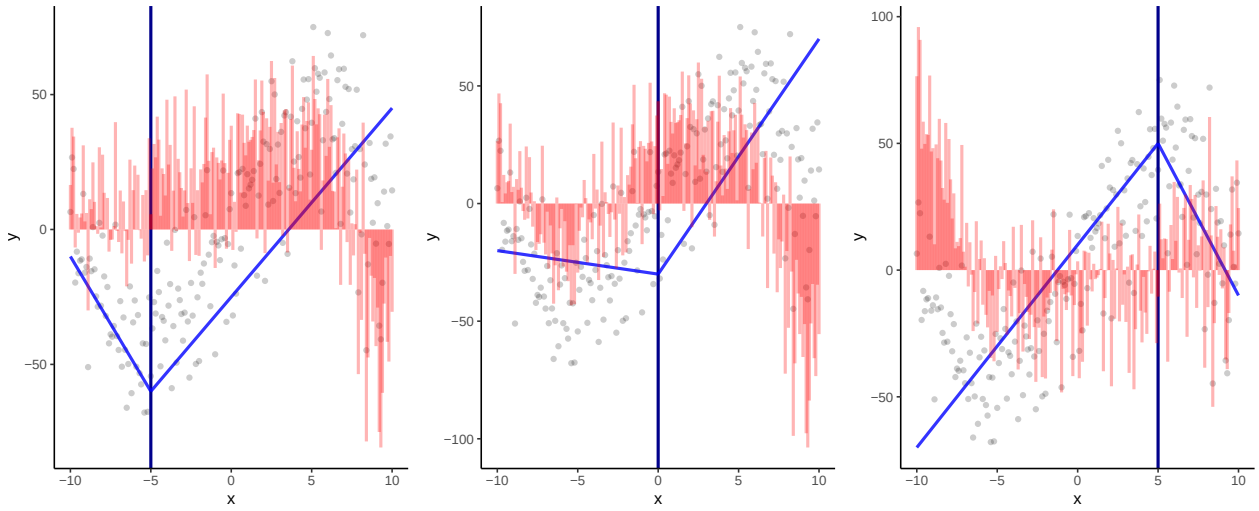
## Comportamiento del Modelo

Esta parte del reporte se basa en el desarrollo de [6] y [2]. Como se aprecia en el ejemplo de abajo, MARS busca los puntos de corte y las pendientes óptimas para aproximar a la variable objetivo. Se puede apreciar como en diferentes segmentos de la variable $x$ existen pedazos de funciones lineales con diferentes pendientes.

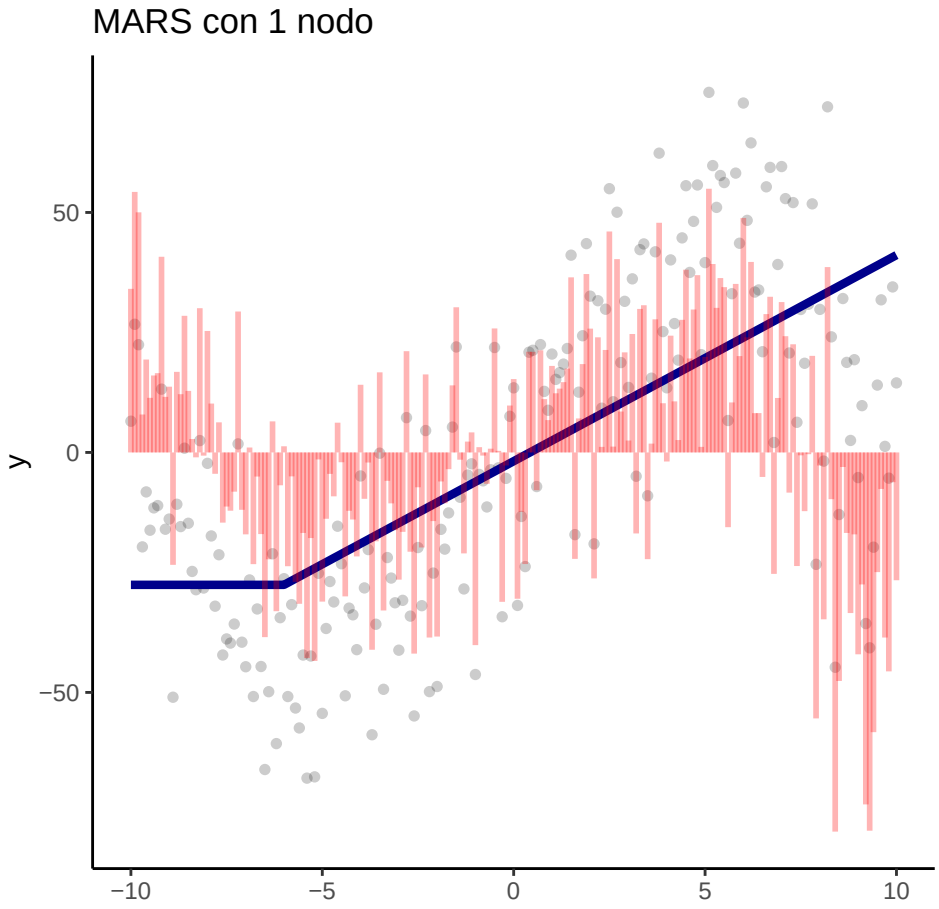





Como podemos ver, darnos a la tarea de encontrar manualmente cada punto de corte y pendientes óptimas se puede traducir a minimizar alguna métrica como el error cuadrático medio, o análoigamente, maximizar la $R^2$:




Lo que hace MARS es encontrar sistemáticamente los mejores puntos de corte y pendientes para minimizar el error de entrenamiento:




## Descripción del Modelo

MARS es un modelo de regresión no paramétrico que a través de funciones bisagra (*hinge functions*). Se puede considerar como una generalización de regresión lineal a pedazos o como modificación al método CART con el fin de mejorar su desempeño en el contexto de regresión [6]. 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,

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

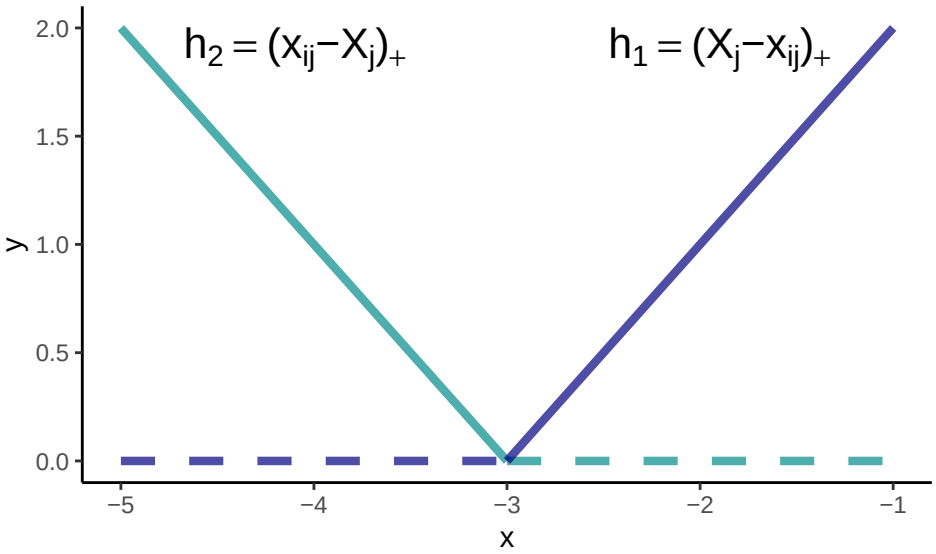
Cada función es lineal a trozos con un cambio de pendiente en el valor $t$, lo cual las hace splines lineales, y a cada par dividido en el valor $t$ se le llama un *par reflejado*.

La idea del método es formar pares reflejados para cada variable de entrada $X_j$ con cambios de pendiente en cada valor observado $x_{ij}$. Por lo tanto, para una variable $X_j$, la colección de funciones base es:

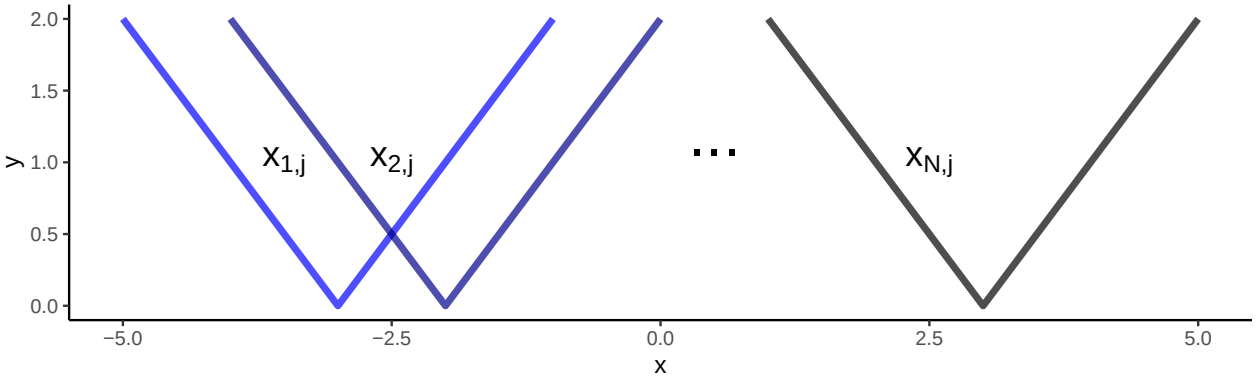
$$\mathcal{C} = \{(X_j - t)_+, (t - X_j)_+\}, \text{ con } t \in \{x_{1,j}, x_{2,j}, \dots x_{N,j}\}, \text{ y } \ j = 1, 2, \dots, p.$$

Cada par reflejado se ve de la siguiente forma:

$$h_1(X) = h(X_j - x_{ij})$$
$$h_2(X) = h(x_{i,j} - X_j)$$




Entonces, para cada $X_j$, el conjunto de los pares reflejados candidatos que habitan en cada uno de los $x_{ij}$ puntos de tal variable es:




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 en $\boldsymbol{\beta}$:

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

donde cada función $h_m(X)$ es elemento de $\mathcal{C}$ o una combinación lineal de estas funciones base. Para esta explicación inicial, usaremos solamente funciones que no involucran interacciones (el modelo con interacciones funciona de forma muy parecida a los árboles de decisión CART).

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. Para ir agregadno términos, todas las funciones en el conjunto $\mathcal{C}$ son candidatas de entrada a $\mathcal{M}$. Para cada observación $x_{ij}$, 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.

El par reflejado que se agrega es aquel que minimice el error de entrenamiento. Este proceso se repite para cada uno de los pares reflejados restantes, resolviendo OLS para determinar el nuevo $\boldsymbol{\beta}$, y agregando aquel

que continue minimizando el error. El criterio de paro puede ser ya sea que ninguno de los pares reflejados restantes reduzca suficiente el error (en términos absolutos o relativos), o hasta que tengamos un número determinado de variables en el modelo.

Este llevará a sobreajuste de los datos, pero esto es lo que se está buscando en esta parte del procedimiento al minimizar el error de entrenamiento. Ya teniendo $\mathcal{M}$, 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 $\lambda$, donde este modelo lo denotamos $\hat{f}_\lambda$.

Hay varios procedimientos que se pueden utilizar para estimar el valor óptimo de $\lambda$, como validación cruzada o bootstrap, pero esto involucra un gran costo computacional. Para minimizar tal costo, 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 es el caso aditivo en $X$ que estamos explicando, y 3 es cuando hay interacciones).

Ya que describimos el modelo base, podemos regresar y considerar cuales serían las diferencias al incorporar términos de interacción. En vez de solo agregar términos aditivamente, se pueden agregar los productos entre las funciones $h_\ell$ ya existentes en el modelo y entre cada para reflejado de cada $x_{ij}$. Nótese que este caso general abarca al anterior, ya que si se quiere agregar un par reflejado sin que interactue con las demás variables, simplemente se multiplica por $h_0 = 1$.

De esta forma, un modelo $\mathcal{M}$ con $M$ términos se actualizará de la siguiente forma tras haber encontrado a la combinación $\hat{\beta}_{M+1}h_\ell(X)\cdot(X_j - t)_+ + \hat{\beta}_{M+2}h_\ell(X)\cdot(t - X_j)_+$ que mejor minimice al error de entrenamiento:

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

donde $h_\ell \in \mathcal{M}$ y $t = x_{ij}$ cualquiera.

## Implementación

Para nuestra implementación, utilizamos una base de datos de `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 tabla 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 que las métricas entre prueba y entrenamiento son bastante consistentes 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 directamente bajo `spam` es el coeficiente asociado $\beta_i$.

```
##                                            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
```

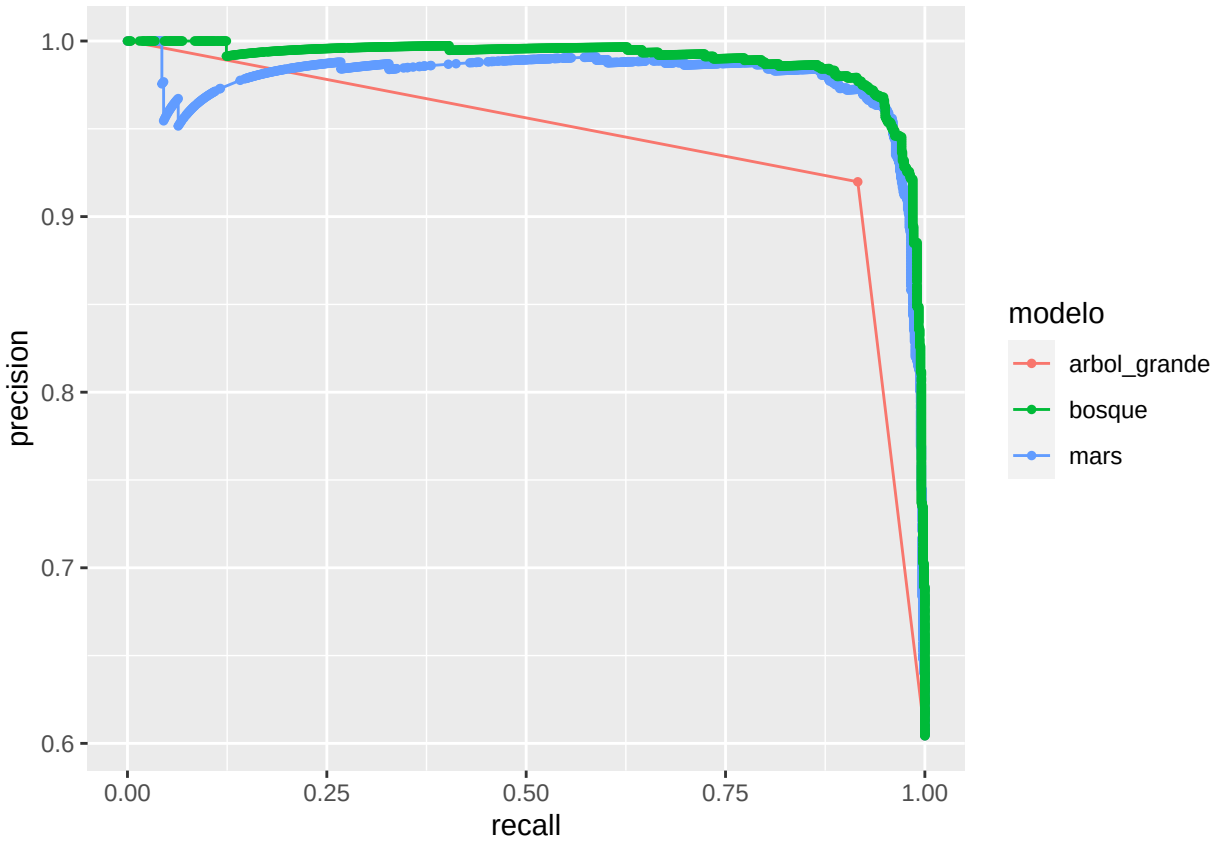
## Comparación con Árboles y Bosques Aleatorios

Ajustamos otros dos modelos, un árbol de decisiones y bosques aleatorios, y podemos comparar el ajuste de MARS contra estos modelos. Las implementaciones 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
```

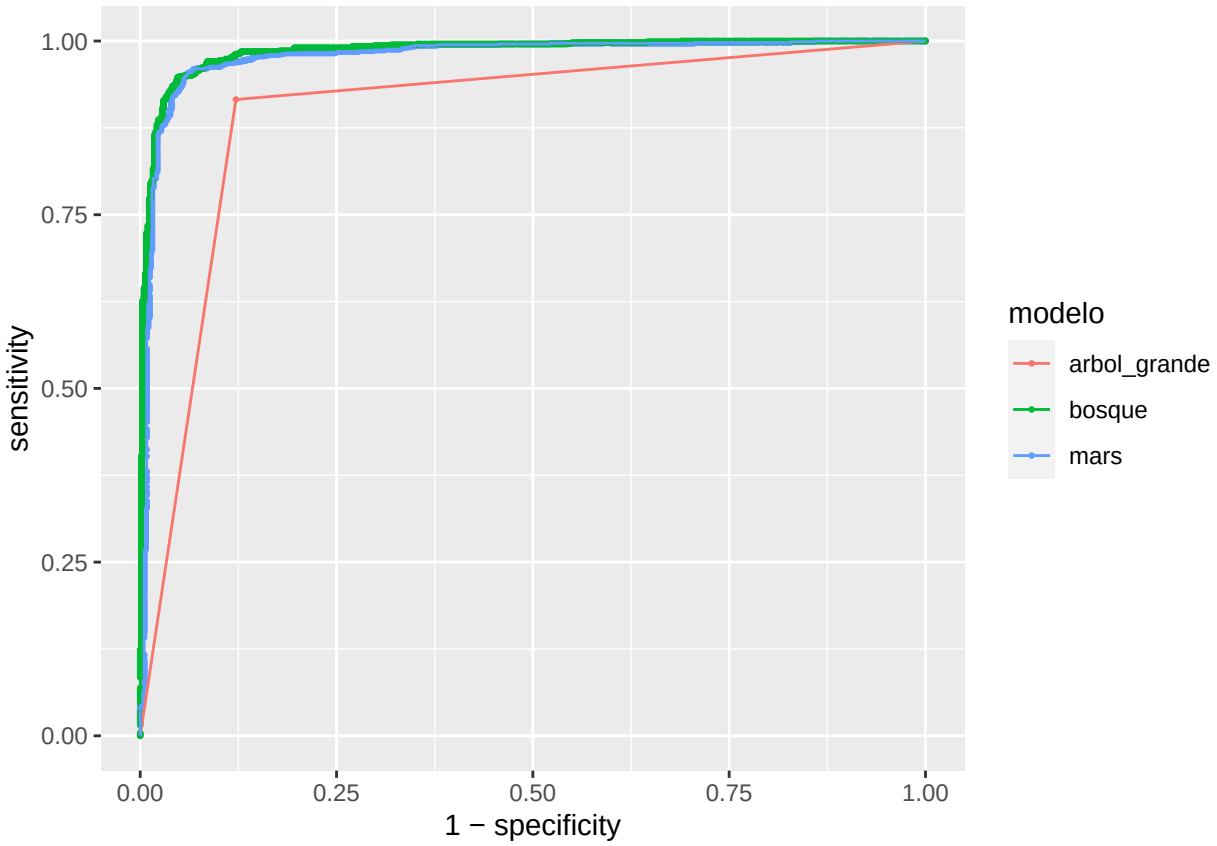
### Curvas Precision-Recall y ROC

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




En términos generales, vemos que MARS tiene un desempeño bastante similar a un modelo de bosques aleatorios. Para los valores de *recall* mayores a 0.5, la diferencia puede ser incluso sólo debido a la muestra. Para valores menores vemos que el desempeño de MARS se deteriora, y en particular por debajo de alrededor de 0.25 incluso llega a ser menor que un modelo de un sólo árbol.

O las curvas ROC:




En la curva ROC, el desempeño de MARS es prácticamente idéntico al de bosques, y ambos son mayores al de un árbol grande. Esto en general muestra que MARS, aún siendo un modelo lineal que no requirió una necesidad de computo excesiva, tiene un desempeño formidable a comparación de métodos que requieren un poco más de requerimientos computacionales (y que en general funcionan muy bien) como árboles.

## Ventajas y Desventajas

### Ventajas

- Al ser lineal, resulta ser relativamente parsimonioso; preserva bastante interpretabilidad de los coeficientes, aún en presencia de interacciones.
- Funciona bien tanto para baja como alta cardinalidad.
- No es computacionalmente costoso.
- Hace selección de variables automáticamente, tanto por si solas como interacciones.
- Preciso si localmente es correcto hacer aporximaciones lineales.

### Desventajas

- Existen modelos que tienen mejor desempeño predictivo; vimos arriba que bosques aleatorios funcionó casi igual, aunque en general este supera a MARS.
- Paquetes como earth no incluyen funciones de órdenes mayores (aunque, de acuerdo a [6], el hacer interacciones de un orden ayuda a la interpretabilidad al hacer superficies igual a 0 donde no queremos aproximar).
- Poco preciso si en general hacer las relaciones lineales, aún localmente, es incorrecto.

## Conclusiones

El modelo de MARS es un modelo bastante bueno y aplicable a varios tipos de problemas. Mientras que su uso principal es en problemas de regresión, presentamos un ejemplo en el que se aplica para clasificación. Esta es una extensión del modelo base, y no necesariamente el tipo de problema en el que se luce este algoritmo. Si se tienen interacciones relativamente lineales y una variable a predecir continua y no un problema de clasificación, entonces el algoritmo tiene muchas ventajas relativo a por ejemplo árboles o una regresión lineal simple. Existe una extensión llamada PolyMARS [7] de el algoritmo base que está diseñado específicamente para problemas de clasificación, pero queda fuera del enfoque de este trabajo.

En términos de la relación que existe entre MARS y CART, si se cambia la forma de las funciones base a $I(x - t > 0)$ y $I(x - t < 0)$ y también se especifica que si un término ya se usó para interacción entonces ya no lo podemos volver a usar entonces obtenemos el modelo generador de árboles CART [1, sec. 9.4.3]. Es términos prácticos, perdemos la habilidad de representar a el método a través de un árbol binario, pero ganamos la abilidad de capturar efectos aditivos a través de interacciones.

Algunas extensiones o ideas de futuro trabajo basado en lo que se presenta aquí es ajustar el modelo de PolyMARS a el dataset de *spam*. Podemos comparar el desempeño de MARS base contra este otro y ver en un dataset relativamente sencillo si perdemos mucho al usar el modelo base, o esto también lo podemos hacer a través de simulación para obtener mejores resultados. Además de esto, sería interesante comparar el modelo de MARS contra el de *random forest* en un contexto un poco distinto, y usarlos en un problema de regresión, ya que para este tipo de problemas está diseñado MARS.

## Referencias


- [1] Friedman, J. H. (1991). "Multivariate Adaptive Regression Splines". The Annals of Statistics. *19* (1): 1–67.
- [2] Notas sobre MARS
- [3] Base de datos Spam
- [4] Datos de Spam
- [5] Árboles aleatorios y bosques
- [6] Hastie, T., et al. "The Elements of Statistical Learning", Springer Series in Statistics, ISBN 0172-7397, 745 pp. (2009): 291.
- [7] Stone, C. J., et al. "Polynomial splines and their tensor products in extended linear modeling: 1994 Wald memorial lecture." The Annals of Statistics *25.4* (1997): 1371-1470.
- [8] Stephen Milborrow. Derived from mda:mars by Trevor Hastie and Rob Tibshirani. Uses Alan Miller's Fortran utilities with Thomas Lumley's leaps wrapper. (2021). earth: Multivariate Adaptive Regression Splines. R package version 5.3.1. https://CRAN.R-project.org/package=earth
- [9] Video del cual basamos algunas gráficas
- [10] Multivariate adaptive regression splines
- [11] How to improve a linear regression with Mars
- [12] Wikipedia de MARS