Boosting

María De Arteaga

Quantil

30 de enero de 2014

Contenido

Objetivo de Boosting

- Método de aprendizaje de máquinas supervisado.
- Combina clasificadores débiles para producir un clasificador fuerte.

Cómo funciona

- Un algoritmo de clasificación débil se aplica de forma iterativa a versiones modificadas de los datos, de modo que se produce una secuencia de clasificadores débiles $G_m(x), m = 1, 2, 3, ..., M$.

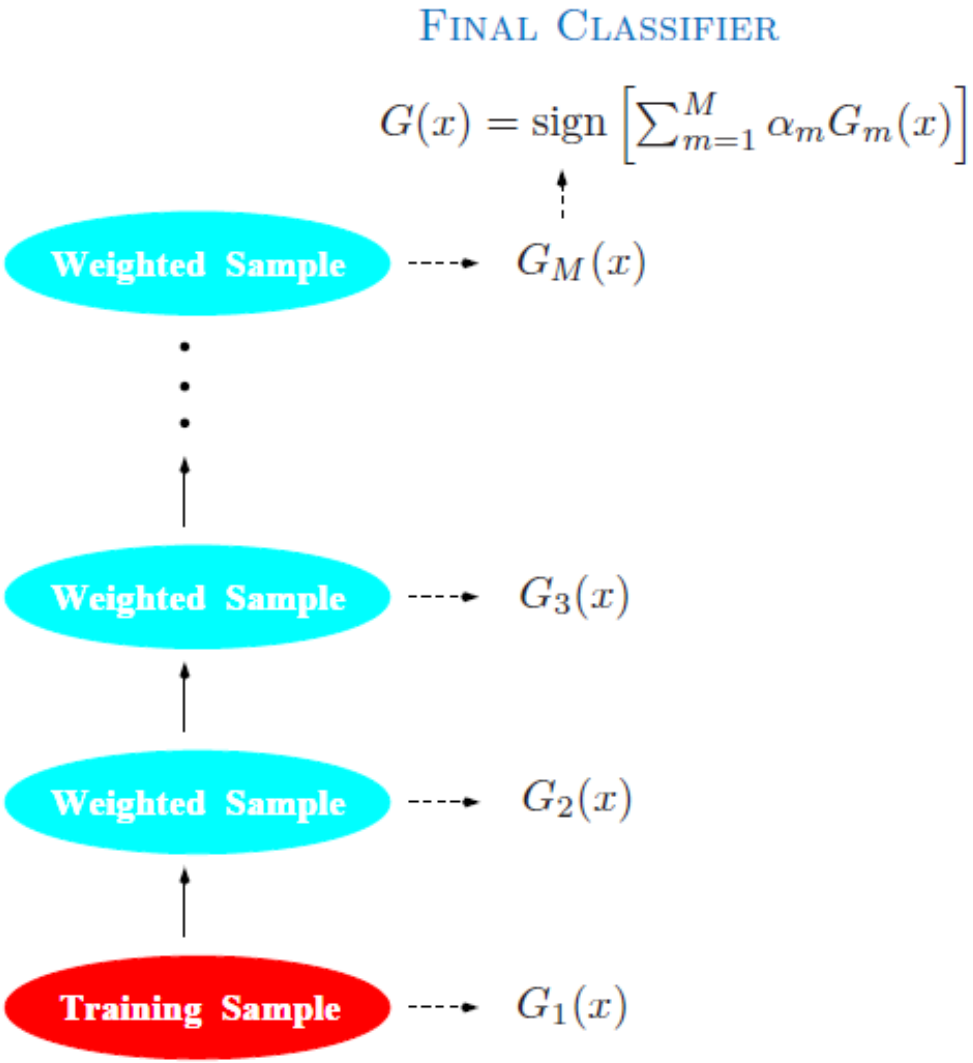
- La modificación de los datos en cada iteración depende del desempeño del clasificador anterior. En los datos utilizados para generar el clasificador G_m se da mayor peso a las observaciones clasificadas erróneamente por el clasificador G_{m-1}.

- Los M clasificadores se combinan por medio de una votación ponderada, donde el peso asignado a cada clasificador depende de su tasa de error.

AdaBoost.M1

- Algoritmo de *Boositng* más popular, desarrollado por Freund y Schapire en 1997.
- Vector de variables predictivas X.
- Considera un problema de dos clases, donde la variable de salida se codifica como $Y \in \{-1, 1\}$.
- Si el clasificador base no da un resultado binario, y por el contrario entrega una probabilidad entre $[-1, 1]$, se puede modificar facilmente. El algoritmo es conocido como *Real AdaBoost*.

Esquema AdaBoost.M1



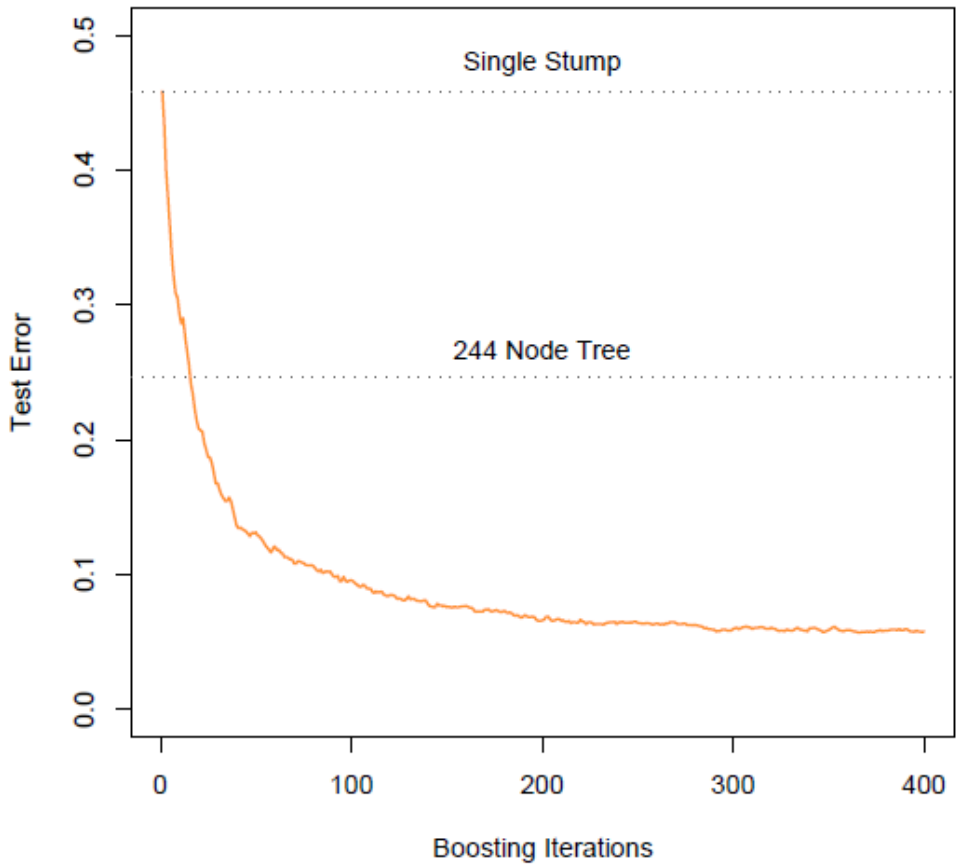
Algoritmo AdaBoost.M1

1. Asignar peso inicial de $w_i = \frac{1}{N}$ a cada una de las observaciones de entrenamiento.

2. Para $m = 1, 2, 3, ..., M$:
   - (a) Obtener clasificador $G_m(x)$ con los datos de prueba, utilizando los pesos w_i.
   - (b) Calcular la tasa de error:
   $$err_m = \frac{\sum_{i=1}^{N} w_i \cdot I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$$
   - (c) Obtener el peso del clasificador débil G_m en el clasificador fuerte:
   $$\alpha_m = \log(\frac{1-err_m}{err_m})$$
   - (d) Reasignar el peso a las observaciones de entrenamiento:
   $$w_i \leftarrow w_i \cdot e^{\alpha_m \cdot I(y_i \neq G_m(x_i))}, \ i = 1, 2, ..., N.$$

3. Clasificador final: $G(x) = \text{sign}[\sum_{m=1}^{M} \alpha_m G_m(x)]$.

Disminución de tasa de error



AdaBoost como modelo aditivo

- Los modelos aditivos son aquellos que toman la forma:

$$f(X) = \sum_{j=1}^{p} f_j(X_j)$$

- AdaBoost es un modelo aditivo, que toma como función base $G_m(x) \in \{-1, 1\}$.

Stagewise Additive Model

En cada iteración m se resuelve para la función base óptima y su correspondiente coeficiente β_m. Se suman funciones base de forma secuencial, sin ajustar los parametros y coeficientes de aquellas que ya se han sumado.

1. $f_o(x) = 0$.
2. Para $m = 1, 2, ..., M$:
   - **i.** $(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$
   - **ii.** $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.

Función de pérdida (costo)

- Una función de clasificación $f(X)$ para predecir Y a partir de los valores de entrada X tiene asociada una función de pérdida (costo) $L(Y, f(X))$, la cual penaliza los errores de predicción.
- Función de mínimos cuadrados $L(Y, f(X)) = (Y - f(X))^2$:
  - La más popular.
  - Conveniente para el análisis.
- Función del error absoluto $L(Y, f(X)) = |Y - f(X)|$.
- Función de pérdida exponencial.
  - **Más robusta que las usadas comúnmente**.

AdaBoost y pérdida exponencial

- AdaBoost.M1 es equivalente a un modelo aditivo por etapas (forward stagewise aditive model) utilizando la función de pérdida exponencial: $L(y, f(x)) = e^{y \cdot f(x)}$
- Esto sólo se descubrió cinco años después de haber sido creado el método.

AdaBoost y pérdida exponencial

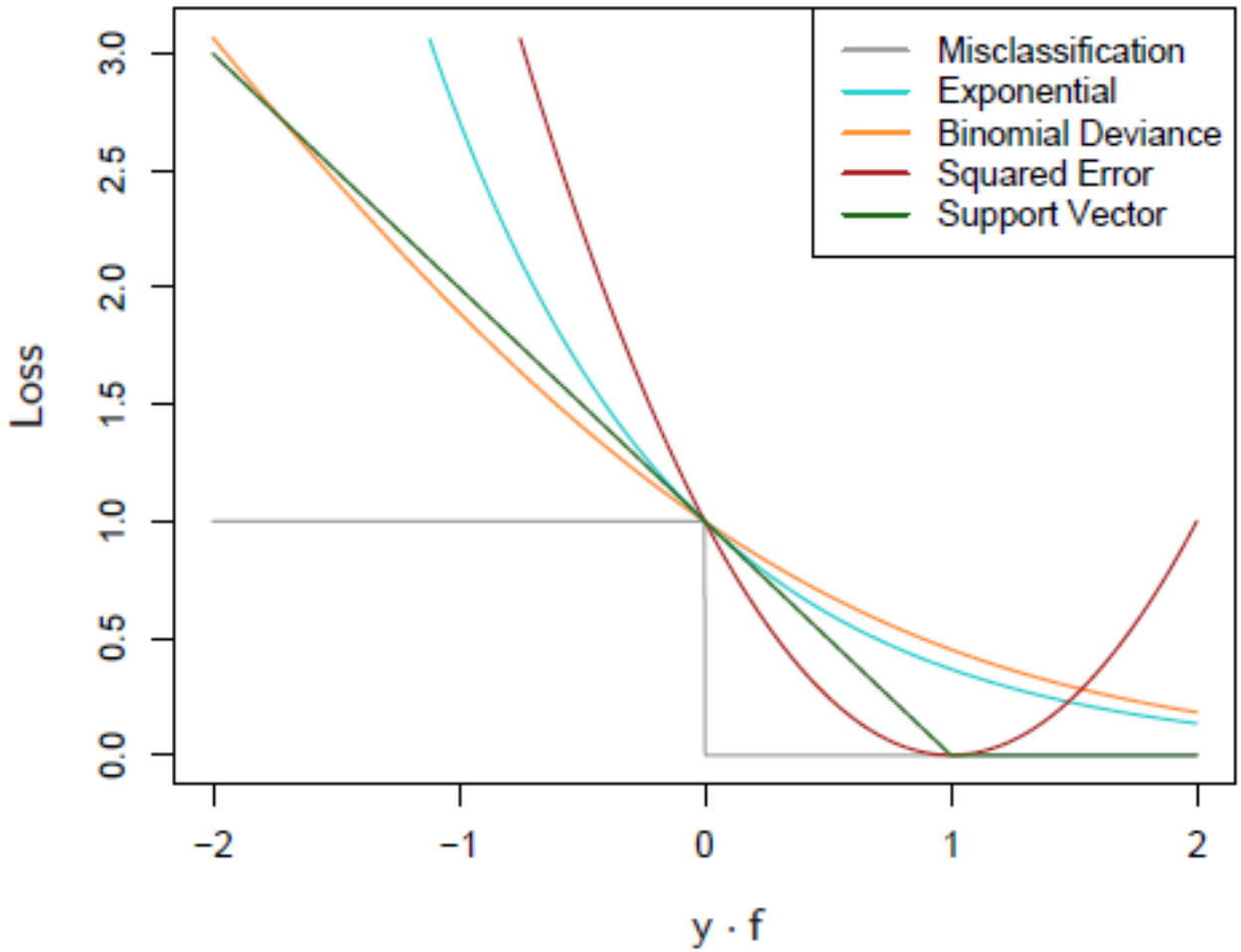
- Aproximación monótona continua a la pérdida por clasificación errónea, la cual es igual a 1 si la clasificación es incorrecta y a cero si es correcta. Por lo tanto, cumple con el objetivo de penalizar más a las observaciones clasificadas de incorrectamente que a las clasificadas de forma correcta.
- Le da una importancia demasiado grande a observaciones con márgenes negativos muy grandes. Esto hace que AdaBoost sea un método muy sensible al ruido.

Exponencial vs. Mínimos cuadrados

- Si el objetivo es la clasificación en clases, la exponencial funciona mejor, pues la de mínimos cuadrados penaliza la clasificación correcta de algunas observaciones, disminuyendola influencia relativa de aquellas clasificadas incorrectamente.

- Existe una modificación de la función de mínimos cuadrados, conocida como "Huberized" que no tiene las ventajas de ambas funciones.

Comportamiento de funciones de pérdida



Boosting de árboles

- Formalmente, un árbol se puede expresar como:
  $T(x;\Theta) = \sum_{j=1}^{J} \gamma_j I(x \in R_j)$,

  con parámetros $\Theta = \{R_j, \gamma_j\}_1^J$.
  medskip
- El boosting de árboles se puede expresar como:

  $f_M(x) = \sum_{m=1}^{M} T(x;\Theta_m)$.
- En cada iteración del boosting, el objetivo es resolver:

  $\hat{\Theta} = \arg min_{\Theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i;\Theta_m))$

Gradient Boosted Model (GBM)

- Se utiliza una aproximación de análisis numérico para la optimización de $\hat{\Theta}$.
- Como el objetivo es minimizar $L(f) = \sum_{i=1}^{N} L(y_i, f(x_i))$, un método basado en *steepest descent* toma el gradiente de $L(f)$.
- El gradiente sólo se puede calcular sobre los datos de entrenamiento. Para que el método sea aplicable a datos no-marcados, se construye un árbol que se aproxime al gradiente negativo.

Gradient Tree Boosting Algorithm

1. $f_0(x) = \text{arg}min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. Para $m = 1, 2, ..., M$:
   - **i.** Para $i = 1, 2, ..., N$, calcular:

     $r_{im} = -[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}]_{f=f_{m-1}}$
   - **ii.** Ajustar un árbol de decisión a los r_{im}, de modo que se obtengan regiones terminales R_{jm}, $j = 1, 2, ..., J_m$.
   - **iii.** Para $j = 1, 2, ..., J_m$ se calcula:

     $\gamma_{jm} = \text{arg}min_\gamma \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$.
   - **iiii.** $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. $\hat{f}(x) = f_M(x)$.

Comparación de métodos de aprendizaje

TABLE 10.1. *Some characteristics of different learning methods. Key:* ▲ *= good,* ◆ *=fair, and* ▼ *=poor.*

Characteristic	Neural Nets	SVM	Trees	MARS	k-NN, Kernels
Natural handling of data of "mixed" type	▼	▼	▲	▲	▼
Handling of missing values	▼	▼	▲	▲	▲
Robustness to outliers in input space	▼	▼	▲	▼	▲
Insensitive to monotone transformations of inputs	▼	▼	▲	▼	▼
Computational scalability (large N)	▼	▼	▲	▲	▼
Ability to deal with irrelevant inputs	▼	▼	▲	▲	▼
Ability to extract linear combinations of features	▲	▲	▼	▼	◆
Interpretability	▼	▼	◆	▲	▼
Predictive power	▲	▲	▼	◆	▲

Ventajas de los árboles

1. Se contruyen relativamente rápido.
2. Los resultados se pueden interpretar.
3. Incluye variables categóricas y numéricas. También puede lidiar con datos faltantes.
4. Selecciona las variables relevantes como parte del algoritmo, por lo que no es sensible a la inclusión de variables irrelevantes.

El problema: Falta de precisión.

Un método *"off-the-shelf"*

- Un método *"off-the-shelf"* es aquel que puede aplicarse directamente a los datos, sin necesidad de un pre-procesamiento muy extenso ni de muchas modificaciones al algoritmo de aprendizaje.
- Por sus características, los árboles de decisión se consideran los más cercanos a un método "off-the-shelf" para minería de datos, pero su baja precisión impide que estos sean utilizados como tal.

GBM como método *"off-the-shelf"*

- AdaBoost mejora la precisión, pero sacrifica:
  - Rapidez.
  - Facilidad para interpretar los resultados.
  - Más sensible a datos de entrenamiento marcados de forma incorrecta y a superposición de distribuciones de las clases.
- GBM Generaliza el boosting de árboles y mitiga los problemas que surgen en el AdaBoost.

Selección de variables con *Boosting*

- Para un árbol de decisión, Breiman definió la importancia de la variable predictiva X_ℓ como:

$$\mathcal{I}_\ell^2(T) = \sum_{t=1}^{J-1} \hat{i}_t^2 \, I(v(t) = \ell)$$

  Teniendo en cuenta que en el nodo t se selecciona la variable $v(t)$, la cual tiene una mejora de $\hat{i}_t^2$.

- Se puede generalizar por medio del promedio de la variable para todos los árboles:

$$\mathcal{I}_\ell^2 = \frac{1}{M} \sum_{m=1}^{M} \mathcal{I}_\ell^2(T_m).$$

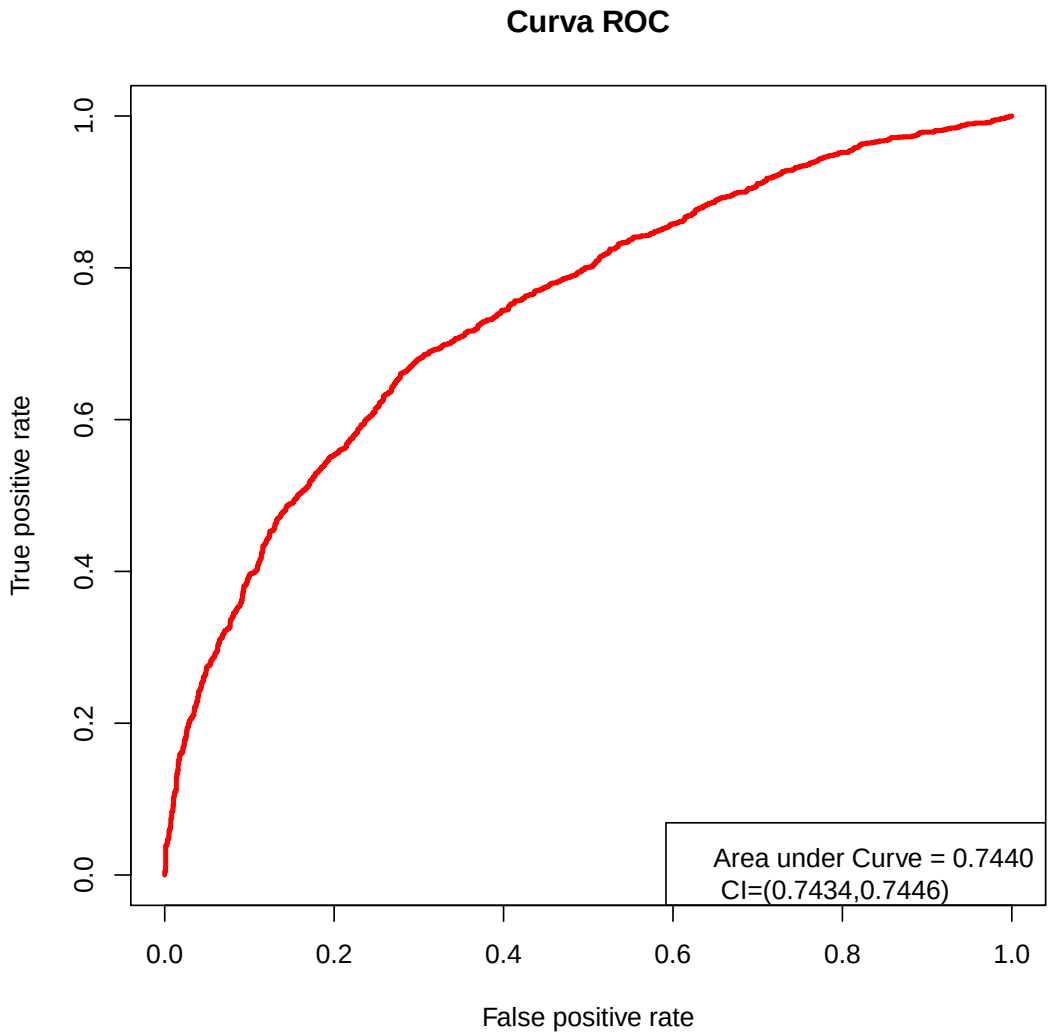
Boosting en Quantil

En el proyecto de la ANDJE:

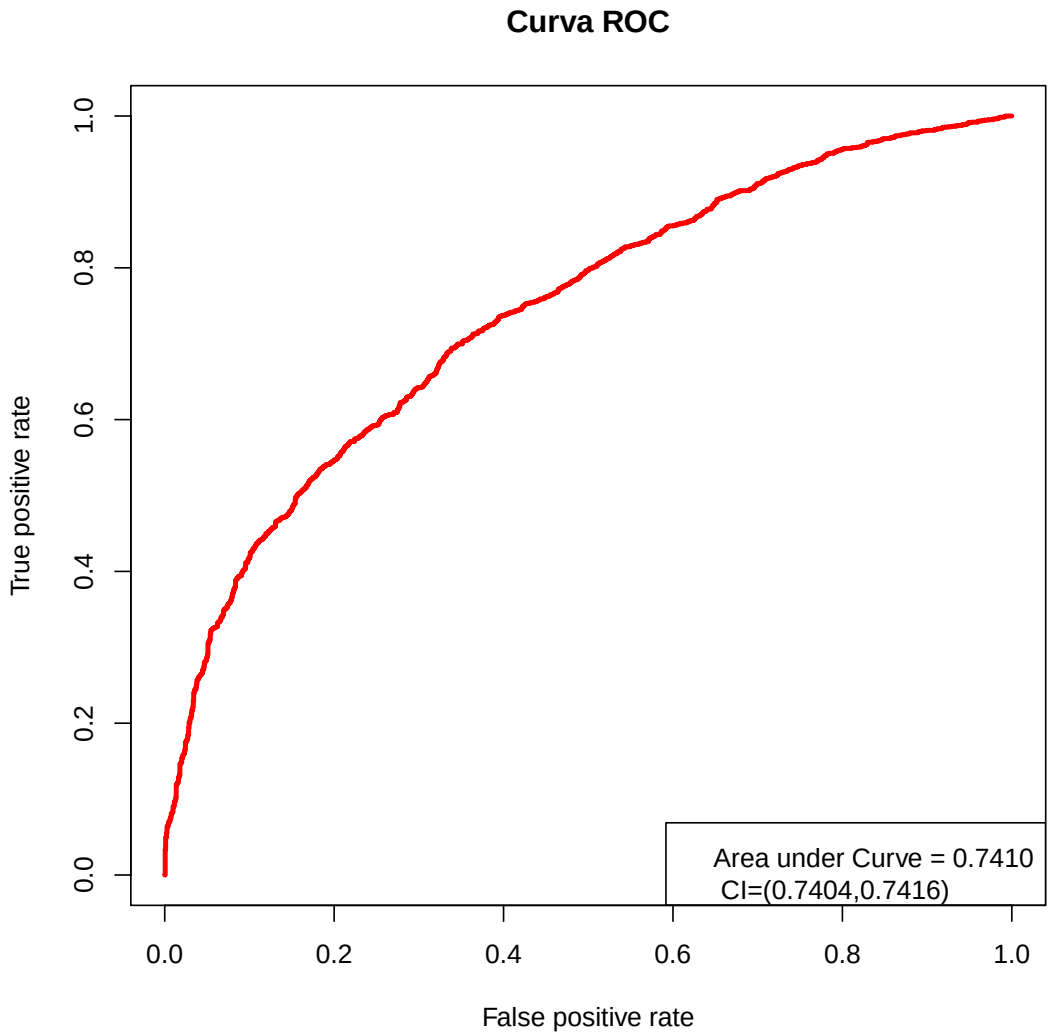
1. Se utilizó GBM para seleccionar variables. Esto logró disminuir las variables de más de 3.000 a alrededor de 60.

2. Después de probar varios métodos de clasificación (regresión logística, redes neuronales, métodos semi-supervisados) el mejor desempeño fue el de GBM.

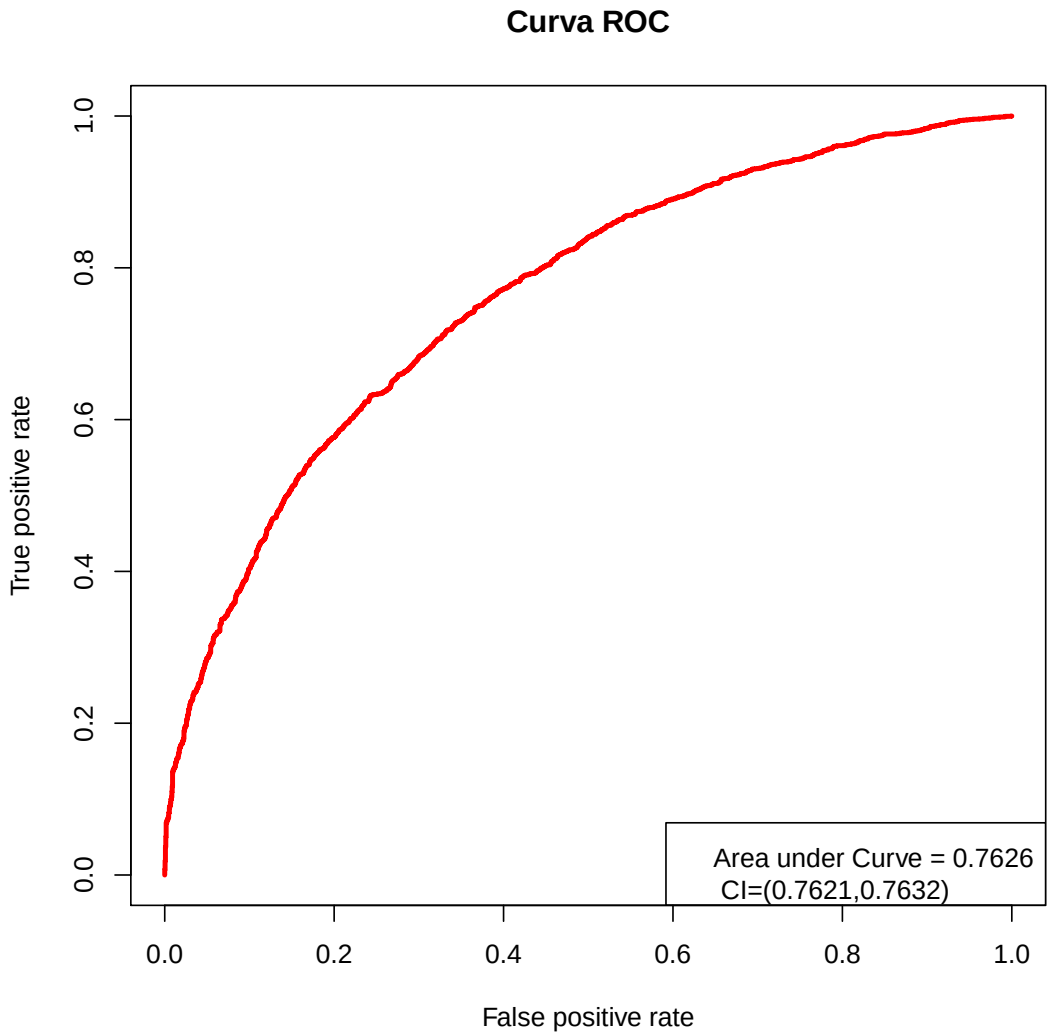
Regresión logística para la ANDJE



Redes neuronales para la ANDJE



GBM para la ANDJE



Entrenamiento vs. validación

Cuadro 5: Comparación de *auc* para los tres métodos de clasificación.

Modelo	área ROC - Entrenamiento	área ROC - Validación
Logit	0.755	0.744
Boosting	0.8054	0.7626
Redes neuronales	0.7691	0.741

Bibliografía

Trevor. Hastie, Robert. Tibshirani, and J. Jerome H. Friedman. The elements of statistical learning. Vol. 1. New York: Springer, 2001.