# Clasificación de usuarios de Twitter

*Informe TP1 - Aprendizaje Automático - Grupo 6*

Ignacio Chiapella, Juan Knebel, Christian Marcusa

*Maestría en Data Mining, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires*
Entregado el 27 de Mayo de 2019

**1. Abstract** **3**

**2. Introducción** **4**

**3. Datos** **5**

3.1 Dataset 5

3.2 Preprocesamiento 5

**4. Metodología** **7**

4.1 Particionado de datos 7

4.2 Primeras pruebas 7

**5. Resultados** **10**

5.1 5-fold Cross Validation 10

5.2 Comparación 13

5.3 Datos faltantes y tolerancia a ruidos 13

5.3.1 Reemplazo de datos faltantes 14

5.3.2 Incorporación de ruido 16

**6. Conclusiones** **18**

**7. Trabajos futuros** **19**

**8. Referencias bibliográficas** **19**

Link al Dataset 19

**9. Anexo** **19**

## 1. Abstract

Con la evolución de las redes sociales, internet, y las tecnologías que se apoyan sobre ambos, cada día es más difícil predecir si la información consumida es real o no. Sobre esta premisa existen además de las “Noticias falsas” (Fake News), perfiles falsos que se dedican a generarla y transmitirla. Estos perfiles pueden ser a su vez personas reales (los hoy llamados “**trolls**”) o robots (“**bots**”), que en muchos casos son programados para re transmitir información y generar tendencia en las redes sociales de **forma automática**. En algunos casos resulta evidente para una persona darse cuenta si la información fue efectivamente escrita por otra persona, o no. Si bien una persona puede clasificar a los usuarios, al tratarse de una cantidad inmanejable se necesitan procesos que puedan “aprender” a estas técnicas, por tal motivo presentaremos dos técnicas usuales de **aprendizaje automático** o **machine learning** que pueden ser utilizadas para clasificar usuarios de redes sociales.


*Keywords:* Aprendizaje automático, robot, bot, publicaciones, trolls

## 2. Introducción

El dominio del problema es crear un producto que pueda detectar si un usuario de la red social **Twitter** es efectivamente un **bot** o una persona real presentado como un un problema de clasificación muy común en los días que corren.

El objetivo de este informe es presentar una descripción de los datos utilizados, y los resultados obtenidos a partir de la creación de clasificadores para los mismos, con **Árboles de decisión** y la utilización del método probabilístico **Naive Bayes**.

El informe se encuentra organizado primero con una breve introducción a los datos y su fuente de origen, luego se presentan y justifican las decisiones tomadas en el pre procesamiento del conjunto de datos. A continuación se detallan los métodos de aprendizaje automático utilizados y sus respectivos hiperparametros. En otra sección se verán los resultados obtenidos del punto anterior la comparación de de los mismos. Para finalizar daremos las conclusiones que obtuvimos luego de los experimentos y propondremos mejoras para trabajos futuros.

## 3. Datos

### 3.1 Dataset

Se utilizó una base de datos obtenida del sitio kaggle que contiene un conjunto de usuarios de la red social Twitter y los mismos se encuentran clasificados en bot o no bot. Los datos originalmente contenían 19 columnas que no todas son necesarias para la clasificación y otras que requieren de otro tipo de análisis (como por ejemplo text mining) que no serán cubiertos en este trabajo práctico. A continuación mostramos a modo de contexto con qué atributos comenzamos el trabajo de análisis exploratorio y cuáles fueron las decisiones que realizamos hasta obtener un conjunto de datos en mejores condiciones.

### 3.2 Preprocesamiento

Los datos tenían condiciones de baja calidad en algunos atributos, que se pre-procesan para su adecuada utilización en el proceso de aprendizaje.

En principio se tuvieron que limpiar las filas que tenían caracteres separadores de más, que impedían reconocer con exactitud los atributos de cada una de esas filas.

En segundo lugar se discretiza la columna “description” (número 4). Se reemplazó el contenido de la misma por el valor Has content y por Has no content cuando es vacío. Esto es porque era información amorfa y que no sumaba al problema de clasificación concretamente el contenido del mismo, sino su presencia o ausencia.

En tercer lugar en la columna “url” (número 5) reemplazó el contenido por Url cuando hay un url presente, Empty cuando el contenido es vacío y None cuando el mismo es None o null. Esto es similar a lo ocurrido con la descripción.

En cuarto lugar se normaliza la columna created_at y graba la cantidad de segundos desde la fecha indicada hasta el 1/1/2000. Había tres tipos de fechas: 2/25/2015 20:11, 25/2/2015 20:11 o Sat Jun 07 19:34:43 +0000 2014. Se hizo un parser para llevar a un formato estándar de fecha, y luego se transformó en un campo numérico para su uso en el aprendizaje.

Además se quitaron los registros cuya clase no estaba presente. Este es el campo “bot” (número 19), ya que eran registros que no aportan al aprendizaje.

Luego de la depuración quedaron 2770 registros listos para usar en los algoritmos. (De unos 2797 registros originales).

Todo el preprocesamiento aquí descrito puede verse en la notebook entregada con nombre *clean_data.ipynb*.

## 4. Metodología

### *4.1 Particionado de datos*

Antes de comenzar con la experimentación de los métodos se particionó el conjunto de datos, previamente des ordenados de manera aleatoria, en conjunto de **entrenamiento** y otro de **test**, usando una proporción del 70/30. Los datos elegidos para test solamente fueron utilizados para calcular la precisión final de los algoritmos.
Con este criterio se obtuvieron 1938 datos de entrenamiento y 831 seleccionados para medir la efectividad final de los algoritmos.
A su vez al conjunto de datos de entrenamiento a su vez se lo volvió a separar en dos, **entrenamiento** y **validación** pero en este caso con una proporción de 80/20.
Finalmente quedaron 1550 datos de entrenamiento y 388 de validación para hacer los ajustes necesarios.

### *4.2 Primeras pruebas*

Antes de comenzar el análisis, se realizaron unas simples pruebas con los valores por default de los hiper parámetros. La finalidad de la prueba fue obtener valores de **accuracy** e identificar la forma de la curva **ROC** para comenzar a analizar los resultados.

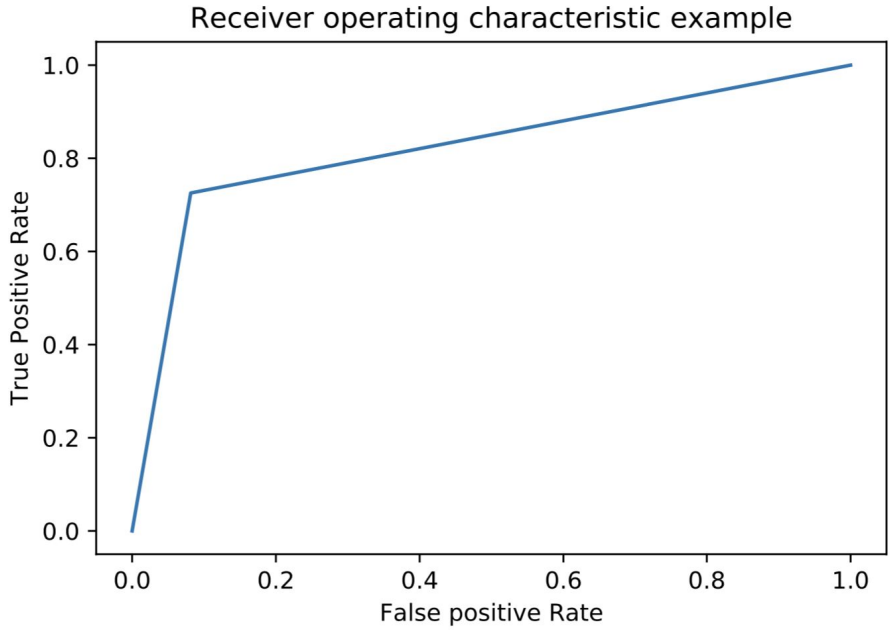


**Figura 1:** Performance piloto con Árboles de decisión

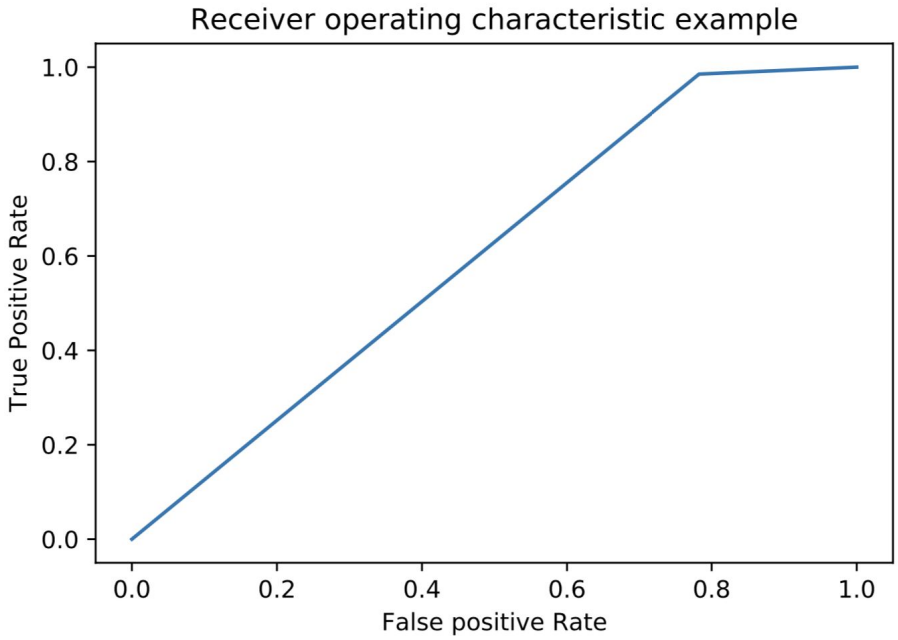


**Figura 2:** Performance piloto con Naive Bayes

En estas primeras pruebas ya se puede apreciar una tendencia favorable a los árboles de decisión, en las próximas secciones se podrá observar que esta tendencia sigue siendo la misma.

*Grid search*

Para poder probar todas las combinaciones posibles de los hiperparametros se construyó un método que itera sobre la profundidad del árbol entre 3, 6 y sin límite cada una de estas alturas se combina con con las funciones de información **Gini Gain** e **Information Gain**.

*Cross Validation*

Se utilizó el método K-fold Cross Validation con K=5 para garantizar la independencia de los datos de entrenamiento y datos de validación y de esta manera evitar el sobreajuste (**overfitting**) sobre los datos utilizados. A continuación podemos observar el gráfico superpuesto de la curva **ROC** para cada uno de los folds involucrados en cada uno de las pruebas de los hiperparametros.

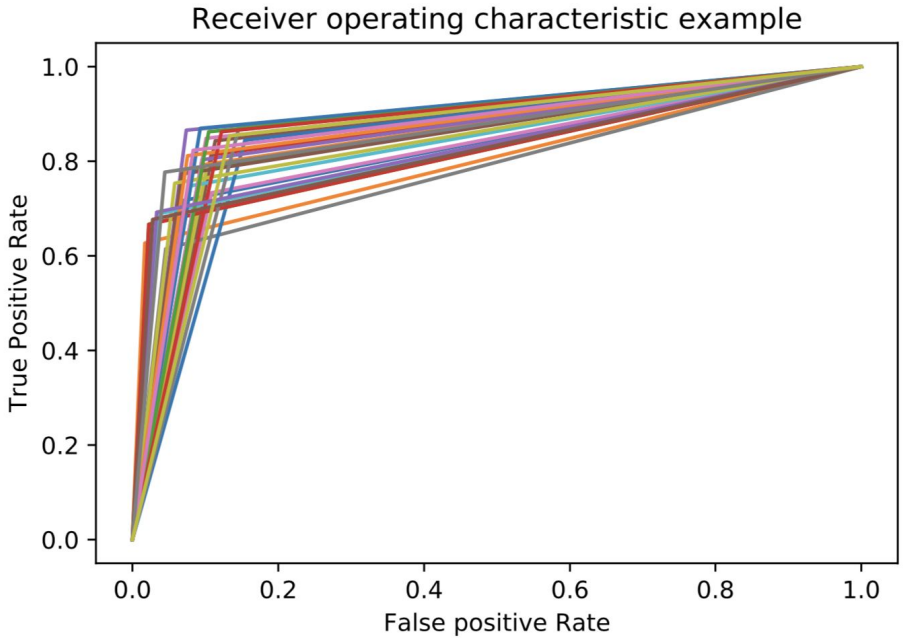


**Figura 3:** ROC en Cross Validation

Las ejecuciones y el código que se utilizó para las experimentaciones pueden encontrarse en la notebook de nombre *fileHandler.ipynb*.

## 5. Resultados

### *5.1 5-fold Cross Validation*

Luego de la ejecución de utilizar la técnica 5-fold cross validation con una búsqueda exhaustiva (o **grid search**) de los hiperparametros se puede visualizar que se obtienen valores superiores con respecto a la exactitud (**accuracy**) utilizando árboles sin restricción de profundidad y la utilización de la función de **Gini Gain**.

A continuación mostramos un gráfico comparando el promedio del accuracy en función de la altura sobre las funciones Gini gain vs Information Gain.

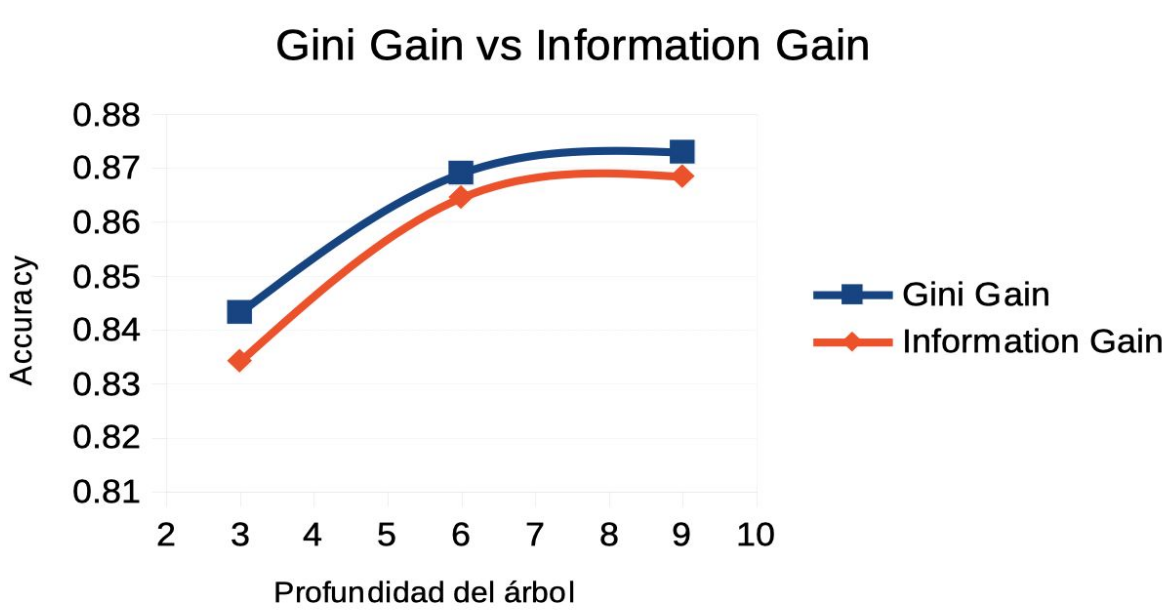


**Figura 4:** La profundidad con valor 9, indica que el árbol generado no tiene restricción de altura.

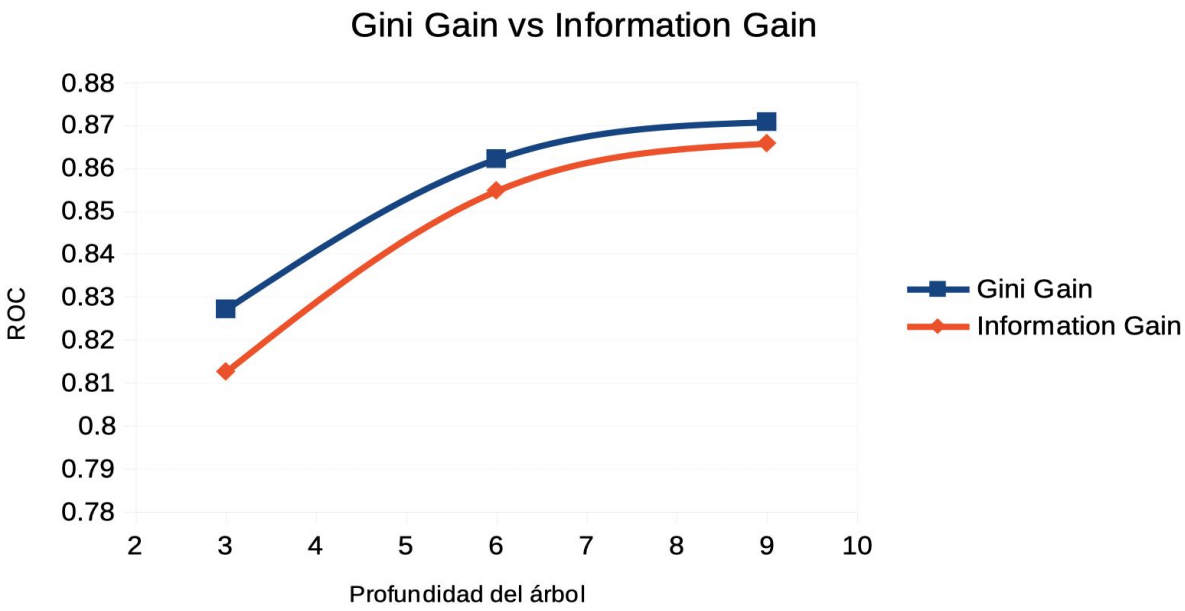


**Figura 5:** La profundidad con valor 9, indica que el árbol generado no tiene restricción de altura.

Para las pruebas del método Naive Bayes se realizó el cálculo de la curva ROC para cada uno de los cinco folds, a continuación mostramos la curva superpuesta y un gráfico con la evolución del accuracy el área bajo la curva para cada uno de los fold.

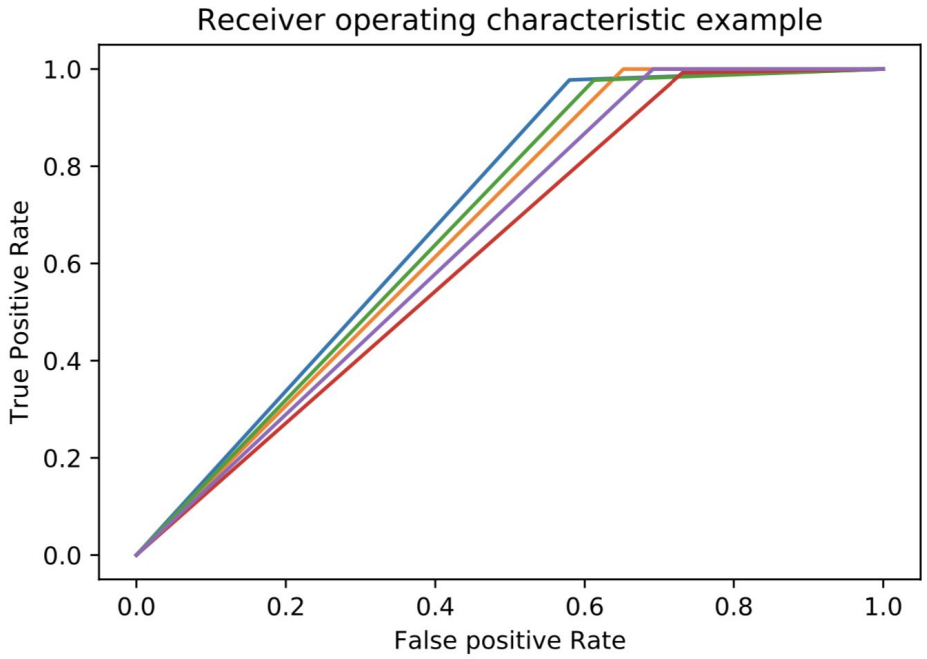


**Figura 6:** ROC en Cross Validation Naive Bayes

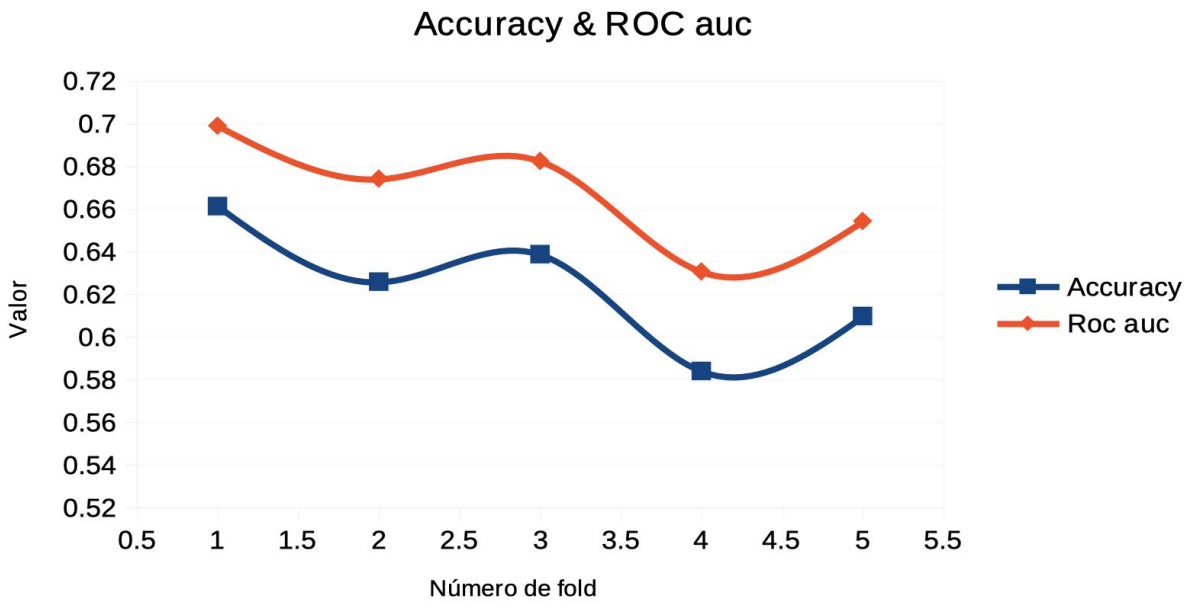


**Figura 7:** Valores del accuracy y ROC auc para los 5 folds.

Lo que se puede observar es que el valor promedio del accuracy aplicando 5-fold Cross Validation y el mismo valor sin realizar la separación son muy similares. En cambio en el área bajo la curva se puede apreciar un diferencia cercana al 10%.

| | Metrica | Valor |
|---|---|---|
| **Valor medio 5-fold Cross Validation** | Accuracy | 0.6238709677 |
| | Roc auc | 0.6681029719 |
| **Sin Cross Validation** | Accuracy | 0.6211340206 |
| | Roc auc | 0.601342711 |

### *5.2 Comparación*

Después de las pruebas realizadas previamente descritas, se llevó adelante el entrenamiento final de los algoritmos tanto de árboles de decisión con altura sin límite y la función Gini Gain, como del clasificador probabilístico Naive Bayes.
El entrenamiento fue realizado con el conjunto original mencionado en el punto 2 y fue testeado con el set de datos que hasta el momento no utilizamos en ninguna prueba. Los resultados en cuanto a su **accuracy** fueron los siguientes.

| | Metrica | Valor |
|---|---|---|
| **Árbol de decisión** | Accuracy | 0.8387484958 |
| **Naive Bayes** | Accuracy | 0.6341756919 |

Puede observarse que el clasificador basado en árboles de decisión es significativamente mejor que el predictor probabilístico para este conjunto de datos.

### *5.3 Datos faltantes y tolerancia a ruidos*

En esta sección describimos los resultados obtenidos luego de completar datos faltantes sobre el conjunto de datos original para los datos de entrenamiento y validación.
También se quiere analizar la tolerancia a ruido (o datos “raros”) de los algoritmos, por eso se implementó una función que genera ruido sobre los conjuntos de datos para luego entrenar los árboles con estos datos.

En todos los casos que se muestran a continuación los entrenamientos de los árboles de decisión fueron realizados con una **altura sin límite** y con la función de **information gain**, que fueron los mejores hiperparametros que obtuvimos luego de realizar el *grid search*.

Un tema de discusión fue que atributo tomar para empezar a probar con la introducción de ruido o reemplazo de datos. Lo que hicimos en esta instancia, fue estudiar las modas por atributo que teníamos y encontramos en muchos casos las modas eran 1 o 0 ya que teníamos muchos atributos booleanos que se traducen a estos. Fue por ello que tomamos la sexta columna que se corresponde a cuántos seguidores tiene un usuario determinado. Esta columna presentaba dos características que eran favorables para la instrucción de ruido o para el reemplazo de datos faltantes. Una de ellas era que tenía un rango de valores grandes, con lo cual no había muchos repetidos razón por la cual al introducir ruido, lo estaríamos haciendo en un gran espectro de valores diferentes, y la otra que al tener esta variedad de datos al reemplazar por un valor fijo de la moda, tanto la de toda la columna como la que se corresponde a la clase 0 o la clase 1 estaríamos cambiando sustancialmente los valores del atributo.

### *5.3.1 Reemplazo de datos faltantes*

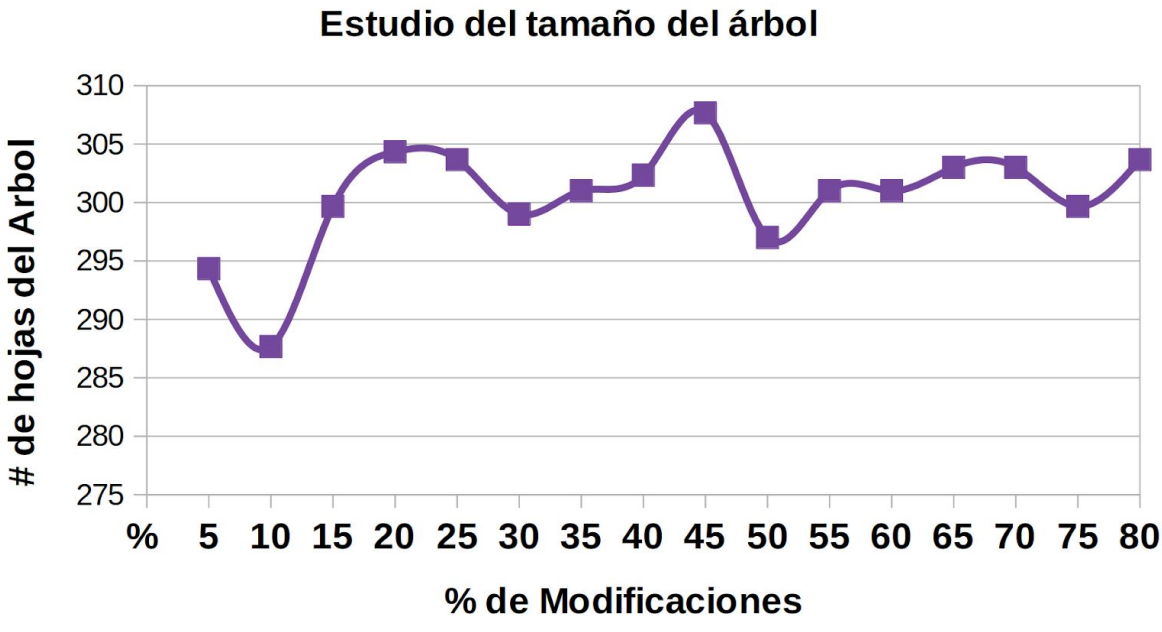


**Figura 6:**Analizamos al ir aumentando el porcentaje de datos cuál fue el promedio de hojas

de los árboles haciendo un promedio entre las 3 clases.

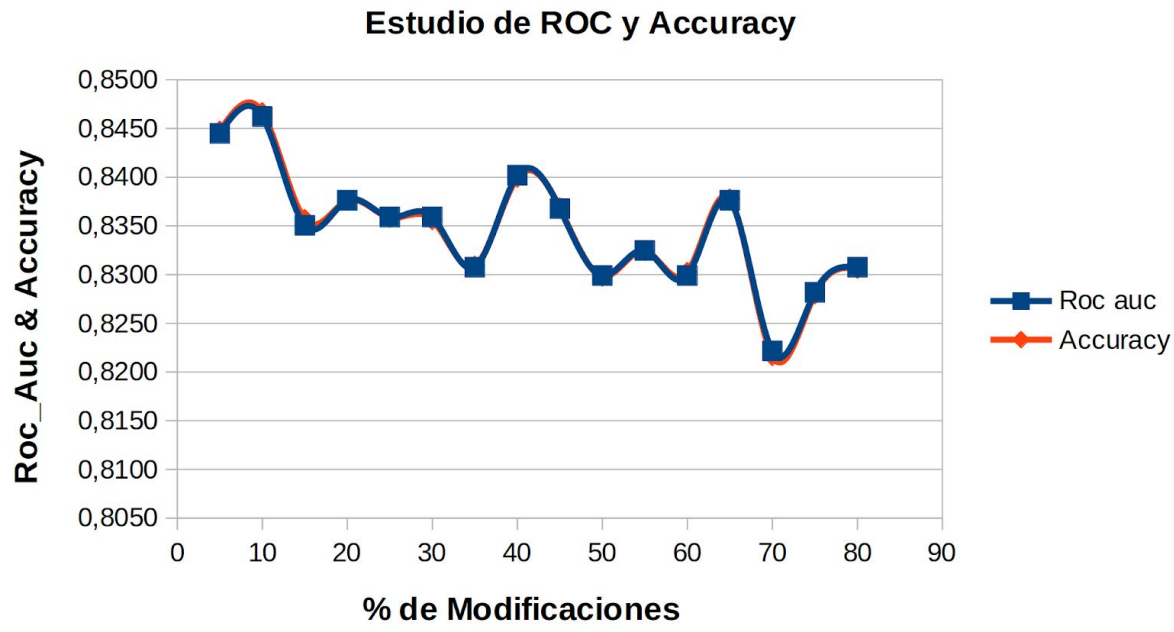


**Figura 7:**Analizamos al ir aumentando el porcentaje de datos cuál fue el promedio medidas de los árboles haciendo un promedio entre las 3 clases.

### *5.3.2 Incorporación de ruido*

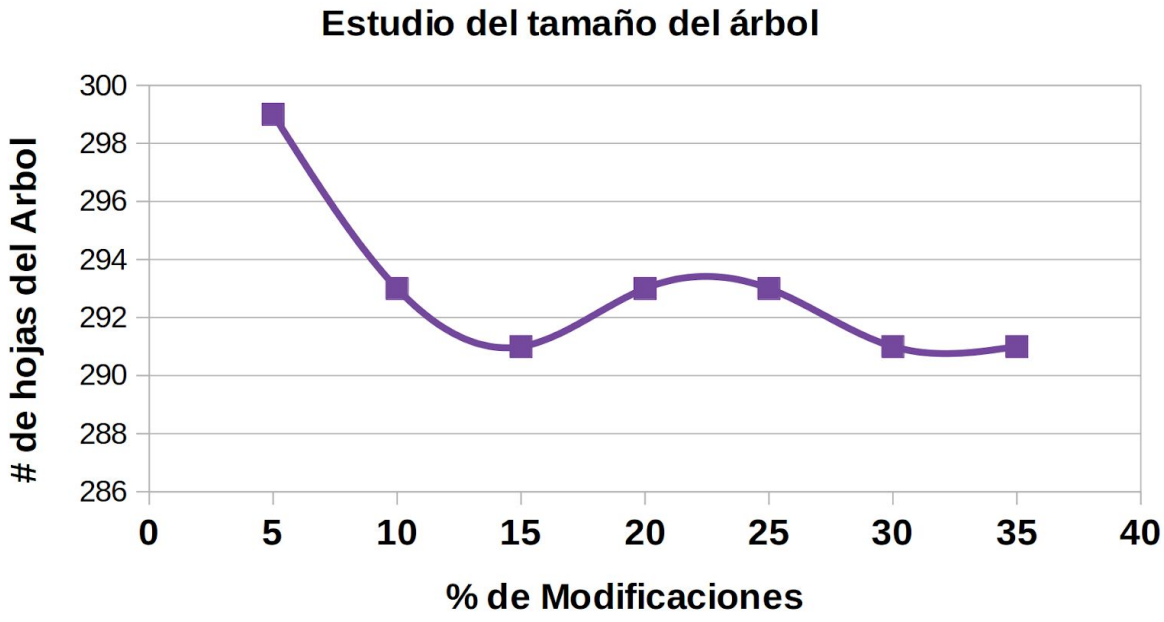


**Figura 8:** Analizamos al ir aumentando el porcentaje de datos a los que le incorporamos ruido, si había o no varianza en el tamaño del árbol medido por la cantidad de hojas del mismo.

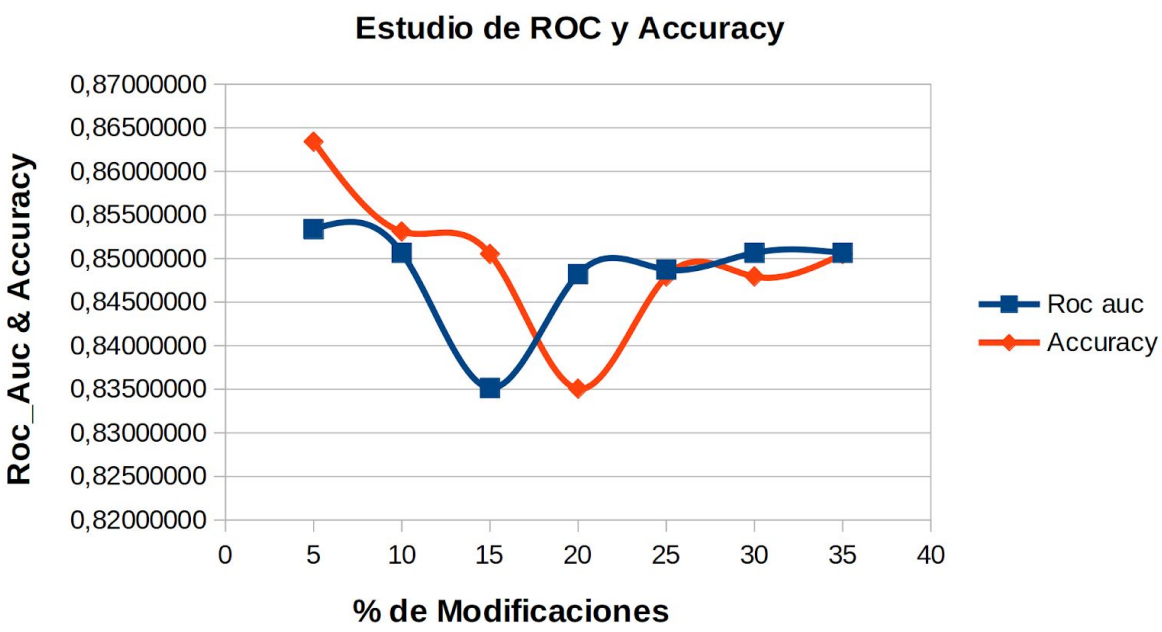


**Figura 9:** Analizamos al ir aumentando el porcentaje de datos a los que le incorporamos ruido, como se modifica el Accuracy el área bajo la curva ROC.

## 6. Conclusiones

El primer punto que se debe resaltar es que antes de comenzar cualquier tipo de pruebas con los métodos que se desea probar, se debe realizar un detallado análisis exploratorio de los datos con los que se cuenta e identificar que atributos aportarán a los mecanismos de **machine learning** y cuáles no. En este punto también resaltamos la importancia de un preprocesamiento de los datos para limpiar datos mal cargados con secuencias delimitadores de string repetidos por ejemplo, o categorizar atributos en grupos pequeños ya que sus valores no aportan mucha información.

En este conjunto de datos particular el método Naive Bayes no fue tan eficaz como hubiéramos esperado, ésta baja eficacia puede deberse a que los atributos esconden una independencia más fuerte de lo que espera el algoritmo. Por tal motivo las curvas **ROC** obtenidas no fueron las esperadas y si también observamos el área bajo la curva que se encuentra cercana al 50% podemos concluir que para el ejemplo aquí tratado, bajo estas condiciones no es adecuado. Utilizando otras técnicas como text mining para darle más importancia a los atributos que tienen texto, tal vez mejore el clasificador.

Los árboles de decisión que se obtuvieron fueron muy parecidos entre todos tanto para su **accuracy** como para el área bajo la curva. Si bien con altura indefinida e Gini Gain se obtienen los mejores resultados, se recomienda utilizar una altura máxima de 6 ya que visualmente es más simple de observar e interpretar (ver árboles generados) con sus 60 nodos en promedio, vs los 250 en los árboles sin límite. En cuanto a la función a utilizar, se recomienda Gini Gain ya que no hace uso de la función logaritmo (cara en términos de complejidad computacional) y además dio mejores resultados.

Como era de esperarse en los árboles decisión, las pruebas muestran que son métodos robustos a faltantes de datos y datos ruidosos. Las métricas muestran que disminuyeron, pero no significativamente como para buscar alternativas.

## 7. Trabajos futuros

Se podría utilizar un espectro más amplio de algoritmos para hacer un análisis completo del problema, pero a primera aproximación el resultado es satisfactorio. Se cuentan con atributos que contienen gran cantidad de texto que deben ser tratados con técnicas de *text mining* o *sentiment analysis* para obtener un mejor enfoque al aquí propuesto.

Se podría experimentar introduciendo ruido no solo a una atributo en particular, sino a un conjunto de ellos, lo mismo con el cambio por la moda.

Las redes sociales tienen atributos similares en cada una de ellas, por ejemplo ubicación, perfil, seguidores o amigos, publicaciones o twitter realizados, contenido re publicado, etc. Teniendo ésto último en cuenta se podría ver si este análisis puede funcionar predecir robots para otras redes sociales además de twitter.

## 8. Referencias bibliográficas

*Link al Dataset*
https://www.kaggle.com/charvijain27/detecting-twitter-bot-data

## 9. Anexo

Se entrega ademas del codigo:

- Set de datos a procesar.
- Planilla donde cada hoja representa el conjunto de datos a partir de los cuales se hicieron los análisis.
- Carpeta con imágenes de los árboles construidos y curvas ROC, tanto de las mejores corridas en cada caso (mejor Accuracy y ROC Auc) como todas las curvas ROC consolidadas en cada corrida.