Proyecto Final Aprendizaje Estadístico: Clasificación automática de noticias

Mario Becerra Contreras 124362 Felipe Domínguez García 127041

12 de diciembre de 2014

Resumen

En el siguiente trabajo se comparan los métodos Random Forest, Extremely Randomized Trees y Gradient Boosting en la clasificación de noticias. Se señalan algunas dificultades al emplear estos métodos y se presentan los resultados más relevantes de 41 modelos probados en total. En particular el estudio es sobre la clasificación de noticias en las categorías de finanzas, seguridad nacional y otros a partir de textos provenientes de periódicos y revistas.

Keywords: NLP, Procesamiento del Lenguaje Natural, Clasificación de texto, bolsa de palabras, Random Forest, Extra Trees, Gradient Boosting.

1. Introducción

Existe una gran cantidad de documentos de texto en formato electrónico, y este número aumenta cada día; por lo que resulta natural buscar una forma de clasificar estos textos de manera automática sin intervención humana. Esta es una de las tareas de las cuales se encarga el procesamiento del lenguaje natural.

El procesamiento del lenguaje natural (de ahora en adelante abreviado NLP por sus siglas en inglés) es un problema de gran interés actual debido a la variedad de aplicaciones que tiene, como en traducción automática de texto, reconocimiento del lenguaje, extracción de información de textos voluminosos, resumir textos, sistemas automáticos de diálogo, corrección de textos, entre otros.

El NLP presenta una enorme variedad de dificultades, principalmente por el hecho de que existen ambigüedades en el lenguaje y palabras que tienen varios significados, y, en el caso de reconocimiento de voz, se tiene el problema de palabras o frases casi homófonas. En este artículo, la atención se centra en la clasificación automática de texto en categorías especificadas previamente; en particular, clasificar noticias extraídas de revistas en *Seguridad Nacional*, *Finanzas* u *Otro*.

En este artículo se comparan distintos algoritmos de clasificación utilizando el modelo de bolsa de palabras (*bag-of-words model*). Se profundiza en los algoritmos de ensemble basados en árboles de decisión, esto debido a su mejor desempeño en comparación con los otros métodos.

2. Desarrollo

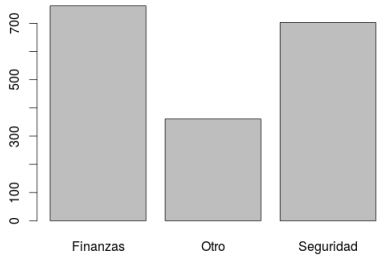
2.1. Datos

Los datos que se usaron son noticias de periódicos y revistas previamente clasificados como "Finanzas", "Seguridad Nacional" u "Otro". Se trató de que la distribución de las noticias fuera equilibrada para evitar sesgo hacia alguna clase en el entrenamiento; sin embargo no se pudo lograr esto y se tuvieron menos noticias de "Otro", esto se puede ver en la Tabla 1 y en la Figura 1. Esto podría traer problemas más adelante a la hora de la clasificación, pues se ha visto que si en un problema de clasificación las clases no están uniformemente distribuidas en el conjunto de entrenamiento, el clasificador final puede tener sesgo hacia la(s) clase(s) predominante(s).

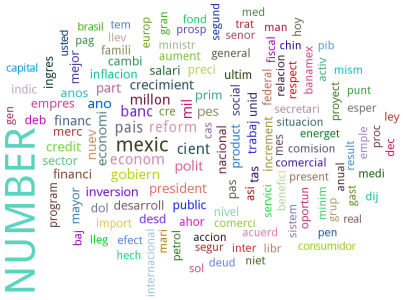
Tabla 1: Distribución de noticias

	y
Finanzas	762
Otro	361
Seguridad	703

Figura 1: Distribución de noticias

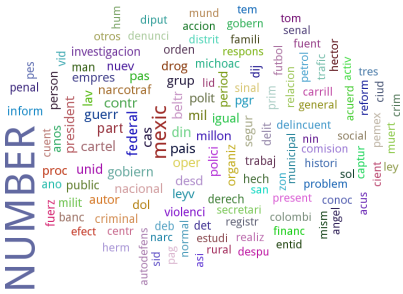


A continuación, en la figura 2, se presentan tres *wordclouds*, cada uno correspondiente a cada clase del problema que nos ocupa. La intención de estas nubes de palabras es notar qué palabras son màs mencionadas en qué contexto; y tal vez estas mismas palabras sean las que los clasificadores utilicen para distinguir entre clases.



(a) Wordcloud Finanzas

(b) Wordcloud Otros



(c) Wordcloud Seguridad

Figura 2: Wordclouds

2.2. Preprocesamiento

Lo primero que se hizo fue preprocesar el texto para ser representado mediante el modelo de bolsa de palabras. Para esto, en principio se toman por separado cada una de las palabras en todo el texto (*tokenizing*); después se convierten a minúsculas; se quitan signos de puntuación; se sustituyen los números, URLs y otros símbolos por palabras clave que los representan; y se eliminan las palabras que no aportan información al modelo, como *stop-words* y palabras poco frecuentes (que aparecen menos de dos veces en todo el texto).
El siguiente paso es lematizar las palabras que quedan, esto es, dejar solo el lema o raíz de las palabras. Por ejemplo, agrupar las palabras "biblioteca" y "bibliotecario" en una sola palabra "bibliotec". Esto se hace mediante el algoritmo de Porter adaptado para español.
Todo este proceso se hace para reducir la dimensionalidad del problema y quitar redundancias que puedan existir en las variables, así como para reducir la posibilidad de errores numéricos; de manera que se crea una matriz término-documento (TDM, por sus siglas en inglés).
Cabe mencionar que se forman dos TDMs, una para el conjunto de entrenamiento y otra para el conjunto de prueba. Esta última se debe hacer de tal forma que contenga únicamente las columnas que aparecen en la TDM de entrenamiento, y así tengan la misma dimensión. Esta es una desventaja del modelo de bolsa de palabras, pues no puede tomar en cuenta palabras que no "conoce" el modelo entrenado.
Después de esto, se hace una transformación a los datos: *term frequency - inverse document*

frequency (tf-idf). El tf-idf es un estadístico numérico que pretende reflejar la importancia de cada palabra en un documento dentro de una colección de documentos o *corpus*. Intuitivamente, lo que hace esto es aumentar la proporcionalidad de acuerdo al número de veces que aparece cada palabra en un documento pero compensar por la frecuencia de la palabra en todo el *corpus*. La definición técnica del tf-idf es el producto de dos estadísticos, el tf y el idf, los cuales están definidos como:

$$tf(t,d) = 0.5 + \frac{0.5 \times f(t,d)}{max\{f(w,d) : w \in d\}} \tag{1}$$

$$idf(t,D) = log\left(\frac{N}{1 + |\{d \in D : t \in d\}|}\right) \tag{2}$$

donde N es el número total de documentos en el *corpus* y $|\{d \in D : t \in d\}|$ es el número de documentos donde aparece el término t. Se le suma el 1 pues si el término no está en el *corpus* entonces se haría una división entre cero.
Finalmente, el tf-idf se calcula al hacer el profucto de tf con idf, i.e., $tfidf(t,d,D) = tf(t,d) \times idf(t,D)$. Así, un valor grande en la matriz td-idf se logra si se tiene una frecuencia alta en el documento correspondiente y una frecuencia baja en toda la colección de documentos, y al contrario para un valor bajo; es por esto que esta transformación puede filtrar los términos muy comunes que no aportan información.
Como ya se mencionó, todo este proceso se hace por separado en el conjunto de entrenamiento y el conjunto de prueba. De esta forma, se tienen dos matrices listas para ser analizadas y probadas; la matriz del conjunto de entrenamiento tiene 6174 variables y 1827 observaciones, mientras que la de prueba tiene el mismo número de variables pero 784 observaciones.

2.3. Clasificadores

Se utilizaron métodos de ensemble para este proyecto, esto principalmente porque debido a que se tenía un problema de alta dimensionalidad, estos métodos resultan en buenos clasificadores. En particular se utilizó *Random Forest* (RF), *Extremely Randomized Trees* (XT) y *Gradient Boosting* (GB). A continuación se da una breve descripción de cada método:

- *Random Forest*: Este método está basado en la idea de promediar árboles de clasificación, esto porque los árboles son muy ruidosos, y al promediarlos este ruido se reduce. Cada árbol del bosque es creado con muestras *bootstrap* de la muestra original utilizando el método de *bagging* (*Boostrap aggregating*), y además cada árbol se hace utilizando solo un subconjunto de las variables, así el corte que se hace en cada árbol no es el óptimo; con esta aleatoriedad la varianza se reduce al disminuir la correlación entre los árboles sin aumentar la varianza, sin embargo, se corre el riesgo de aumentar el sesgo.

- *Extremely Randomized Trees*: En este método, la aleatoriedad va un paso más allá en la forma en que se hacen los cortes. Como en RF, los cortes se hacen con subconjunto de las variables, pero en este método los cortes son escogidos aleatoriamente para cada variable candidato y se escoge el mejor de cada uno de estos cortes generados aleatoriamente como regla de decisión.

- *Gradient Boosting*: Este método, como su nombre lo indica, está basado en *boosting*. La motivación del *boosting* es una combinación de clasificadores débiles para producir

uno fuerte. Desde esta perspectiva, *boosting* es como *bagging*, sin embargo, todo el proceso es estructuralmente diferente. El propósito de *boosting* es aplicar el algoritmo de clasificación débil secuencialmente a versiones modificadas de los datos repetidamente, produciendo una secuencia de M clasificadores débiles $G_m(x)$, $m = 1, 2, \ldots, M$. Las modificaciones de los datos en cada paso consisten en aplicar pesos a cada uno de las observaciones de entrenamiento, dándole más peso a las observaciones que no fueron clasificados de forma correcta. De esta forma, cada nuevo clasificador está forzado a concentrarse en las observaciones que no clasificó bien el paso anterior. *Gradient boosting* es básicamente una implementación de esto pero mejorado para que la rapidez aumente y sea más robusto; sin embargo, aún con esto, en comparación con la rapidez de RF o XT, GB se queda muy atrás. Una desventaja de este método es que tiende a sobreajustar las muestras de entrenamiento, por eso es importante tener un conjunto de prueba para validar.

También se debe mencionar que para estos clasificadores (y prácticamente todos) necesitan diversos parámetros para poder ser entrenados, de hecho, el fin del entrenamiento es estimar estos parámetros; en el caso de los árboles se quiere estimar los cortes óptimos. Pero además de los parámetros "naturales" que se tienen que estimar, existen otros, como la profundidad máxima de cada árbol, el número de árboles que se van a promediar, el parámetro de regularización, etc. Así, que, naturalmente, se hicieron diversas estimaciones para distintos parámetros.
Los parámetros particulares que se probaron fueron:

- *Random Forest*:
  - Profundidad máxima de cada árbol: 5, 10, 50, 78 y 100.
  - Número de árboles: 100, 1000, 5000 y 10,000.
- *Extremely Randomized Trees*:
  - Profundidad máxima de cada árbol: 5, 10, 50, 78 y 100.
  - Número de árboles: 100, 1000, 5000 y 10,000.
- *Gradient Boosting*:
  - Profundidad máxima de cada árbol: 1.
  - Parámetro de regularización: 0.01.
  - Número de árboles: 1000.

Esto da un total de $4 \times 5 \times 2 + 1 = 41$ modelos distintos. Los resultados más representativos se enuncian en la siguiente sección.

3. Resultados

Se mencionó que se entrenaron 41 modelos distintos; y los resultados de todos son muy parecidos. Esto es demasiado para incluir en este artículo, así que solo se enuncian los modelos más importantes o que clasificaron mejor. El parámetro para decidir si un modelo era bueno o malo fue la kappa del modelo. La kappa es un estadístico calculado a partir de

la matriz de confusión que intuitivamente dice qué tan bueno es un clasificador comparado con el azar. Si la kappa es 1, entonces el clasificador es perfecto, si es 0, entonces es igua a adivinar la clasificación con base en la probabilidad de cada clase, y si la kappa es menor a 0, entonces el clasificador es peor que el azar.
Todos los clasificadores entrenados tuvieron una kappa mayor a 0.6 y una precisión mayor a 0.7. En el anexo se pueden ver a detalle varios de las matrices de confusión y resultados OOB de los clasificadores. Llama la atención en las matrices de confusión, como en las figuras 2 en adelante del anexo, que el clasificador no distingue muy bien entre las clases "Otro" y "Seguridad Nacional". Esto puede deberse a la distribución de las clases en el conjunto de entrenamiento no estaba muy bien balanceada. Para corregir esto se podría proseguir de varias formas: *oversampling* de la clase minoritaria (en este caso "Otro"), *undersampling* de la clase mayoritaria ("Finanzas" y "Seguridad Nacional") o técnicas de remuestreo como SMOTE (*Synthetic Minority Over-sampling TEchnique*); las cuales no se usaron porque salen del contenido del curso.
En la Figura 5 se aprecian las matrices de confusión del GB. Se puede ver en la figura arriba a la derecha la gráfica de la pérdida devianza de este clasificador, y se que este clasificador podría disminuir la devianza con más árboles de entrenamiento, sin embargo, esto no se creyó necesario pues la devianza de prueba seguramente iba a bajar, pero la confusión que el clasificador tiene con "Seguridad Nacional" y "Otro" no iba a aclararse. Esto debido a que el problema viene desde la muestra de entrenamiento y no es error del clasificador en sí.
Es de interés ver en las Figuras 5 en adelante que las variables importantes de cada clasificador son muy parecidas para los tres clasificadores; y además son palabras que tiene sentido que se usen para distinguir los temas. Palabras como "reforma", "financiero", "economía", etc. También se puede ver que las gráficas de importancia de variables son prácticamente las mismas para XT y para RF. Esto no es sorpresa pues los métodos son muy parecidos, de hecho XT es solo una versión modificada de RF.

4. Conclusiones

A pesar de que se probaron 41 modelos distintos para clasificar las noticias, los resultados de todos ellos fueron muy parecidos y consistentes. Consistentes en el sentido de que todos parecían tener los mismos tipos de errores y que consideraban las mismas palabras clave para distinguir entre los tres temas que se quiere distinguir.
Un error consistente en todos los clasificadores fue que había una confusión entre las clases "Seguridad Nacional" y "Otro". Este problema viene desde la muestra así que al clasificador no se le puede hacer mucho para mejorar esto.
Se puede concluir que para este problema en particlar, los tres métodos de ensemble que se probaron se desempeñan de formas muy parecidas y que si se quiere mejorar la precisión se debe conseguir una muestra más grande y mejor balanceada.

5. Anexo

5.1. Gradient Boosting: 1000 árboles

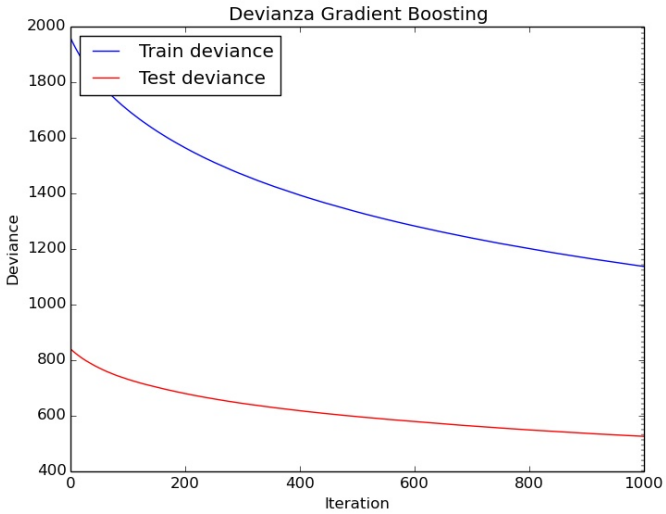


Figura 3: Devianza prueba y entrenamiento

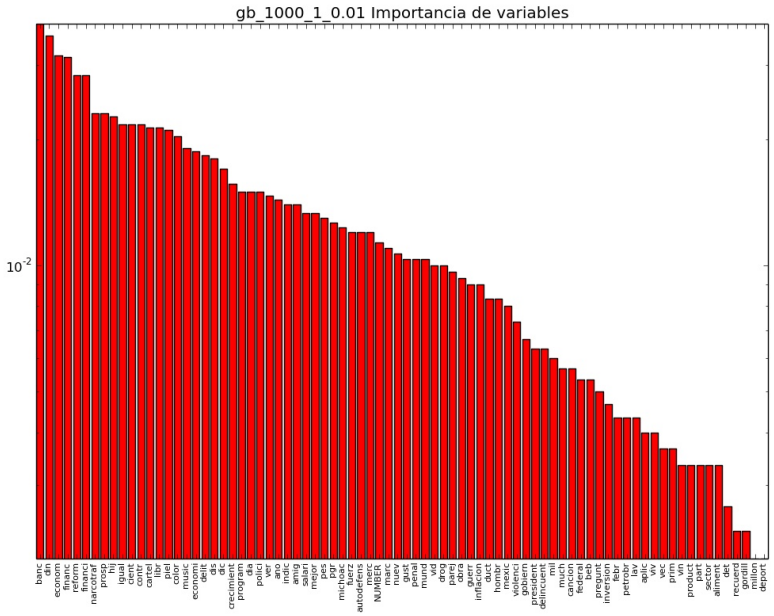
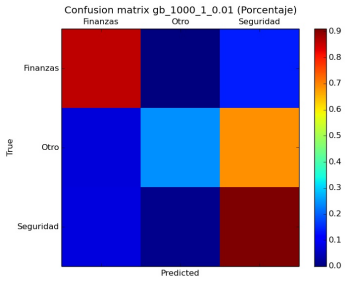
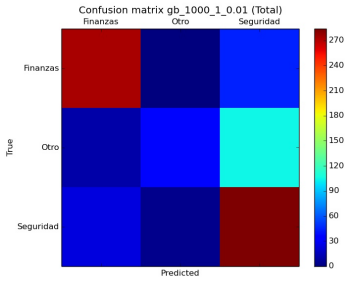


Figura 4: Importancia de variables



(a) Matriz de confusión OOB



(b) Matriz de confusión en conjunto de prueba

Figura 5: Gradient Boosting 1000

		Predicted		
True	274	0	45
	11	37	105
	24	4	284

Tabla 2: Matriz de confusión

Kappa	0.6054
Test score	0.7589
Número de arboles	1000
Profundidad máxima	1
Parámetro de regularizacion	0.01

Tabla 3: Matriz de confusión

5.2. Random Forest: profundidad máxima 5, 1000 árboles

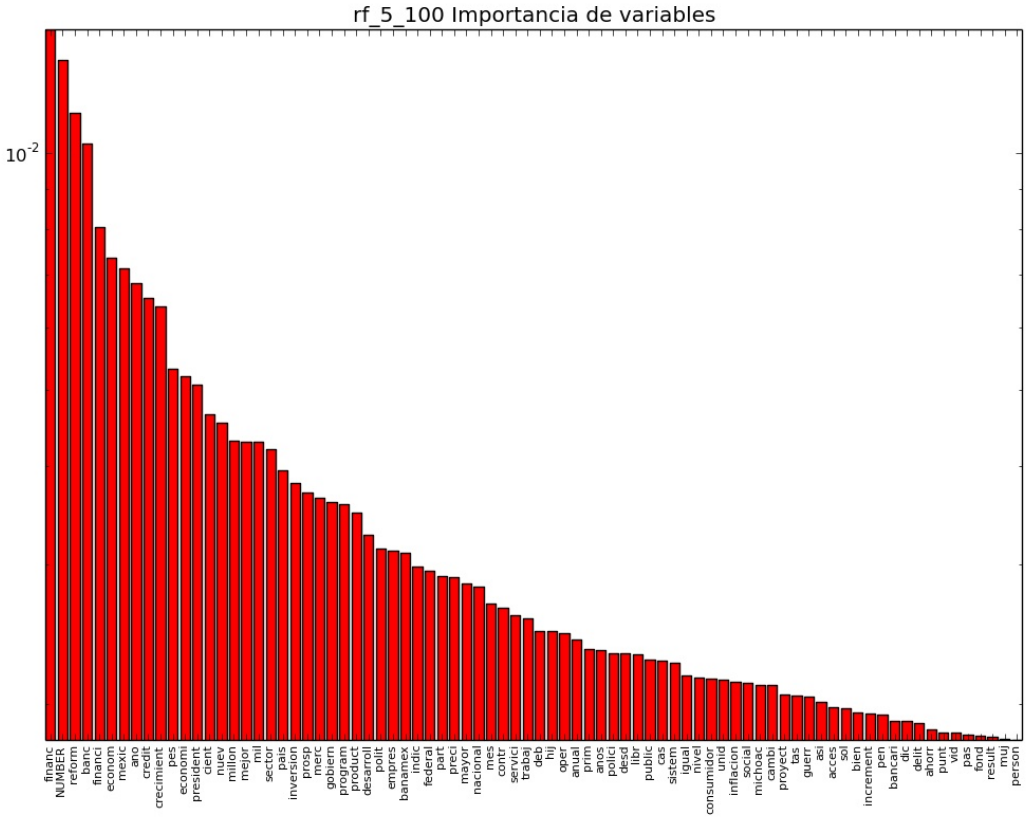
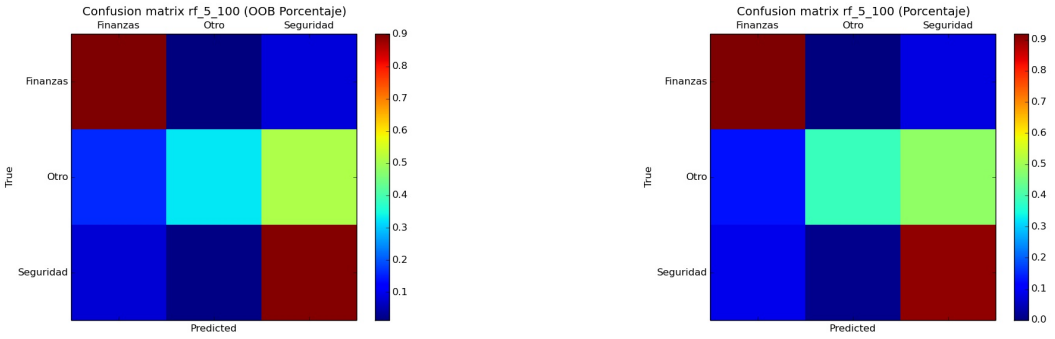


Figura 6: Importancia de variables



(a) Matriz de confusión OOB

(b) Matriz de confusión en conjunto de prueba

Figura 7: Random Forest 5 1000

		Predicted		
True	293	0	26
	20	59	74
	27	4	281

Tabla 4: Matriz de confusión RF

		Predicted		
True	687	10	65
	58	118	185
	58	14	632

Tabla 5: Matriz de confusión OOB

Kappa RF	0.6873
Test score RF	0.8073
OOB score RF	0.7865

Tabla 6: Resultados

5.3. Random Forest: profundidad máxima 50, 10,000 árboles

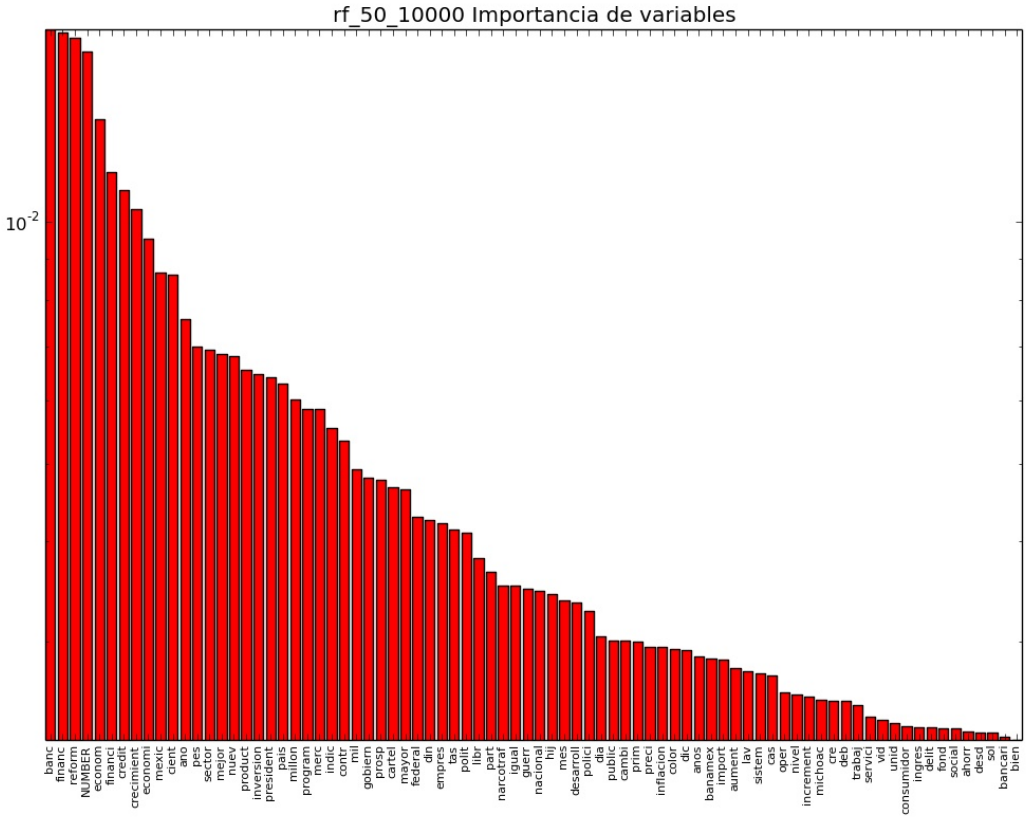
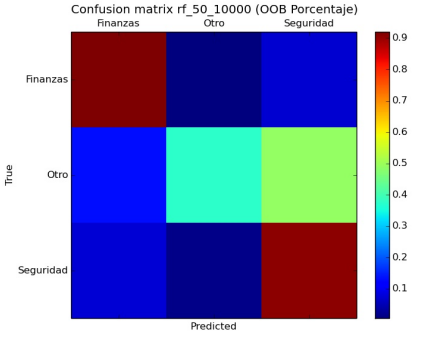
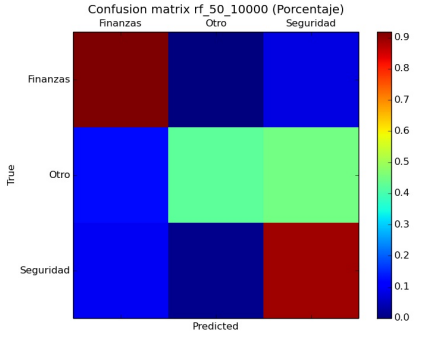


Figura 8: Importancia de variables



(a) Matriz de confusión OOB



(b) Matriz de confusión en conjunto de prueba

Figura 9: Random Forest 50 10000

		Predicted		
True	293	0	26
	19	65	69
	30	4	278

Tabla 7: Matriz de confusión RF

		Predicted		
True	701	5	56
	48	136	177
	53	12	639

Tabla 8: Matriz de confusión OOB

Kappa RF	0.6942
Test score RF	0.8112
OOB score RF	0.8078

Tabla 9: Resultados

5.4. Extremely Randomized Trees: profundidad máxima 5, 100 árboles

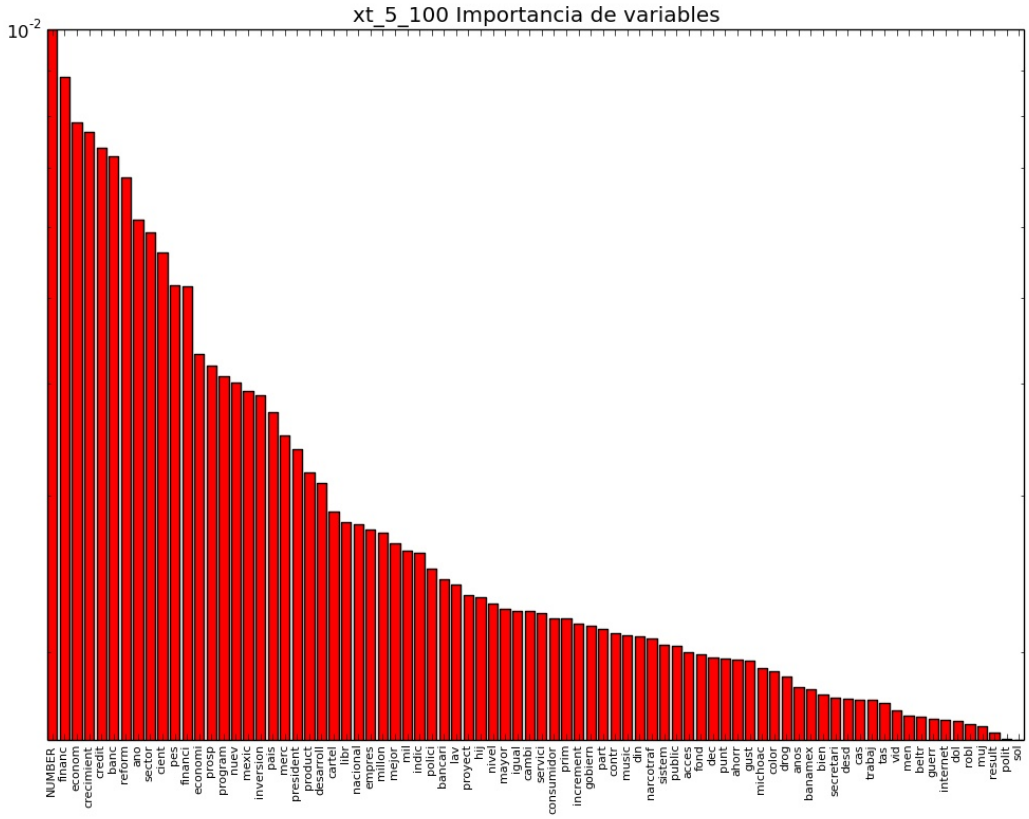
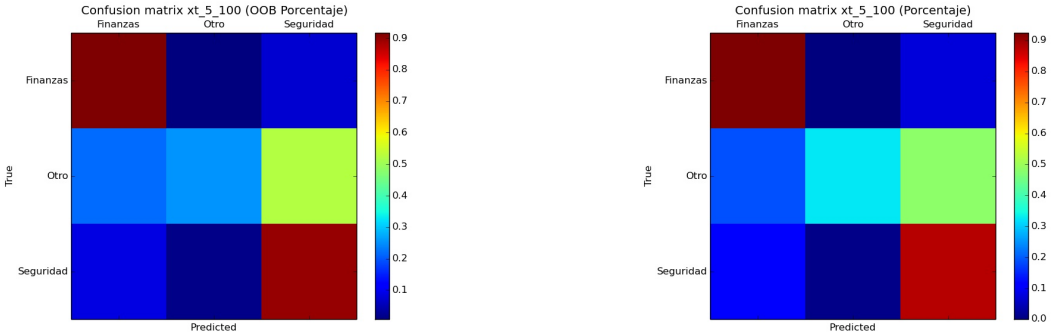


Figura 10: Importancia de variables



(a) Matriz de confusión OOB

(b) Matriz de confusión en conjunto de prueba

Figura 11: Extremely Randomized Trees 5 100

		Predicted		
True	295	0	24
	29	50	74
	35	3	274

Tabla 10: Matriz de confusión XT

		Predicted		
True	699	6	57
	79	92	190
	62	11	631

Tabla 11: Matriz de confusión OOB

Kappa XT	0.6562
Test score XT	0.7895
OOB score XT	0.7783

Tabla 12: Resultados

5.5. Extremely Randomized Trees, profundidad máxima 50, 10,000 árboles

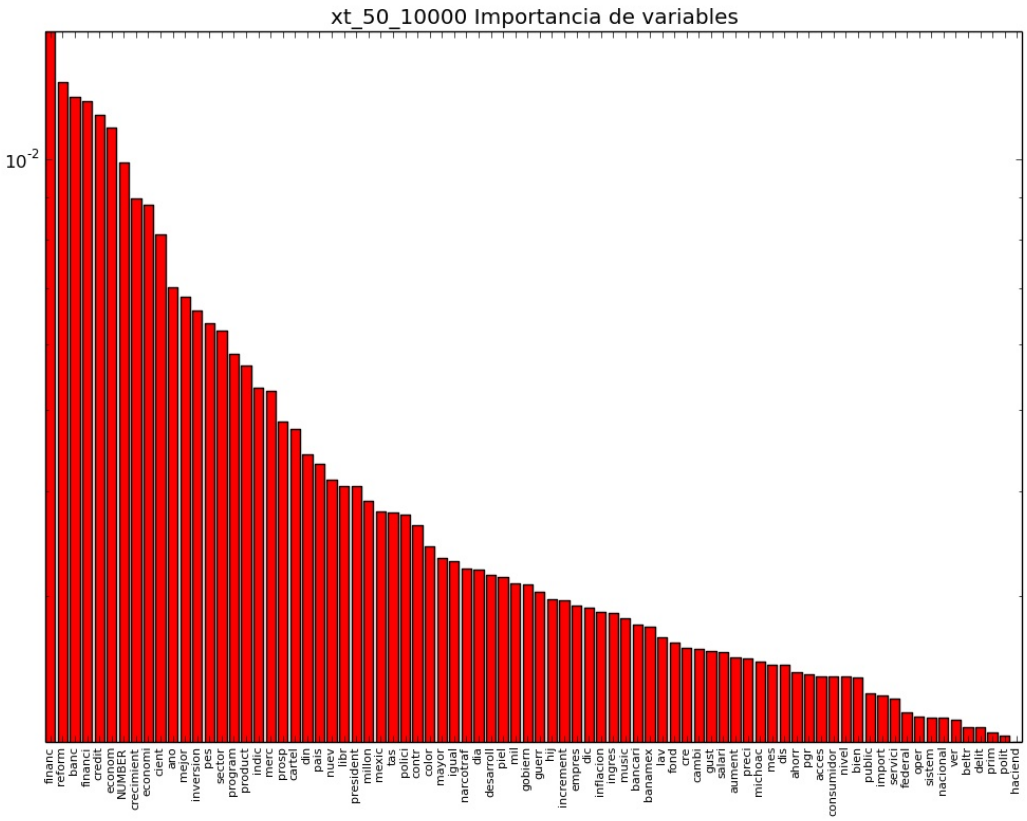
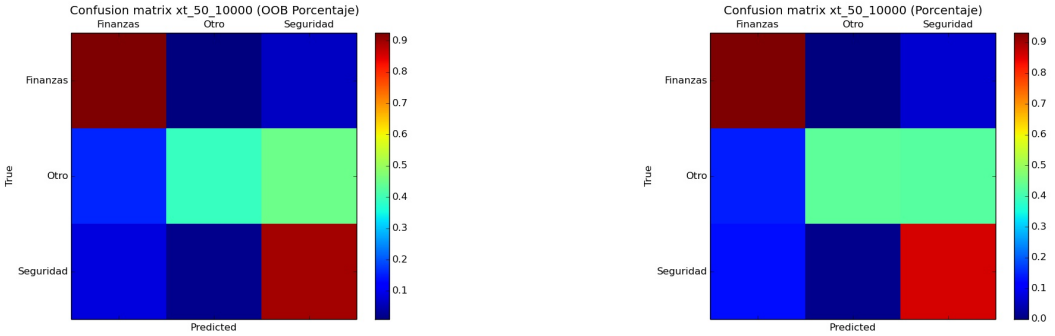


Figura 12: Importancia de variables



(a) Matriz de confusión OOB

(b) Matriz de confusión del conjunto de prueba

Figura 13: Extremely Randomized Trees, profundidad máxima 50, 10,000 árboles

		Predicted		
True	297	0	22
	22	66	65
	40	4	268

Tabla 13: Matriz de confusión OOB

		Predicted		
True	705	7	50
	57	141	163
	60	15	629

Tabla 14: Matriz de confusión OOB

Kappa XT	0.6838
Test score XT	0.8048
OOB score XT	0.8073

Tabla 15: Resultados

Referencias

[1] Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze. *Introduction to Information Retrieval*, Cambridge University Press. 2008.

[2] Trevor Hastie, Robert Tibshirani & Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.*, 2nd Ed. Springer 2009.

[3] Wikipedia, la enciclopedia libre. *Stemming*. Fecha de consulta: 10 de diciembre de 2014. Disponible en `http://es.wikipedia.org/wiki/Stemming`

[4] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. *Scikit-learn: Machine Learning in Python*. JMLR 12, pp. 2825-2830, 2011. Disponible en `http://scikit-learn.org/stable/modules/ensemble.html`