DEPARTAMENTO DE COMPUTACION

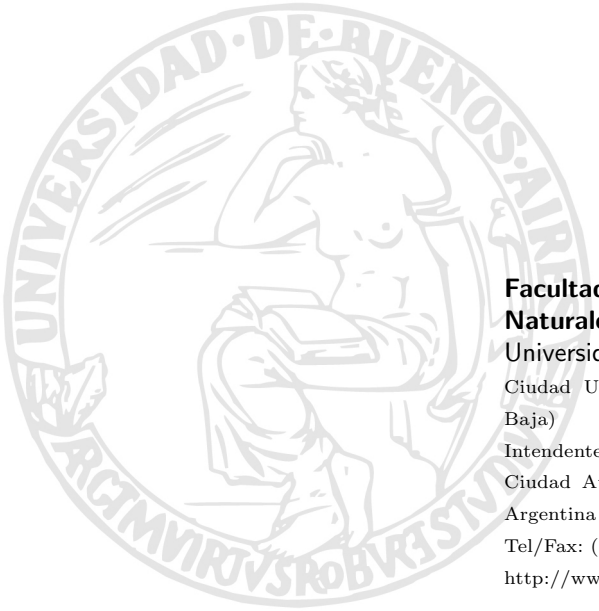
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 1

Detección de Spam

27 de septiembre de 2016 — Aprendizaje Automático

Integrante	LU	Correo electrónico
Fernández, Gonzalo	836/10	gpfernandezflorio@gmail.com
Damian, Aleman	377/10	damianealeman@gmail.com
Matías, Pizzagalli	257/10	matipizza@gmail.com



Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2160 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (54 11) 4576-3359

http://www.fcen.uba.ar

Índice

Introducción

1. Metodología

Cargamos los mails de los archivos json. Extrajimos los atributos de la base de mails y los guardamos como matrices. Separamos los datos de entrenamiento de los de test utilizando la función train_test_split y almacenamos cada conjunto en archivos npy.

2. Extracción de atributos

Para saber qué palabras identifican mejor a los mails de spam, implementamos un script que cuenta la cantidad de ocurrencias por palabra, relativa entre los mails de spam contra los de ham. Es decir, para cada palabra en los archivos json este script devuelve la cantidad de apariciones de dicha palabra en el archivo de spam menos la cantidad de apariciones de la misma palabra en el archivo ham. Este script está implementado en el archivo `wordCounter.py`. En el archivo `words-100.txt` se pueden ver las 100 palabras con mayor cantidad de ocurrencias relativa.

En principio usamos como atributos la cantidad de apariciones de estas palabras en cada mail. Por otro lado agregamos los atributos:

- Cantidad de palabras en mayúsculas.
- Cantidad de palabras compuestas por números.
- Cantidad de palabras escritas con la primera letra en mayúscula.
- Cantidad de letras promedio por palabra.
- Cantidad de letras máxima por palabra.

Se pudo observar que a médida que ibamos agregando atributos, el modelo predictivo obtenía un valor más alto de accuracy y rápidamente se pudo llegar al 97 % de accuracy.

3. Modelos

Utilizamos los siguientes algoritmos de aprendizaje y seteamos los hiperparámetros a partir de un `gridSearch`:

Decision Tree max_depth: 14 16 o None, min_samples_split: 3, criterion: entropy

Random Forest max_depth: None, min_smaples_split: 4,5, criterion: entropy, max_features: 95 , n_estimators: 80

Naive Bayes

Vecinos Más Cercanos(KNN) n_neighbors: 4, weights: distance

Support Vector Machines (SVM)

4. Reducción de dimensionalidad

Sobre técnicas de reducción de dimensionalidad usamos:

PCA Utiliza la descomposición SVD de las matrices para quedarse con los vectores singulares más significativos.

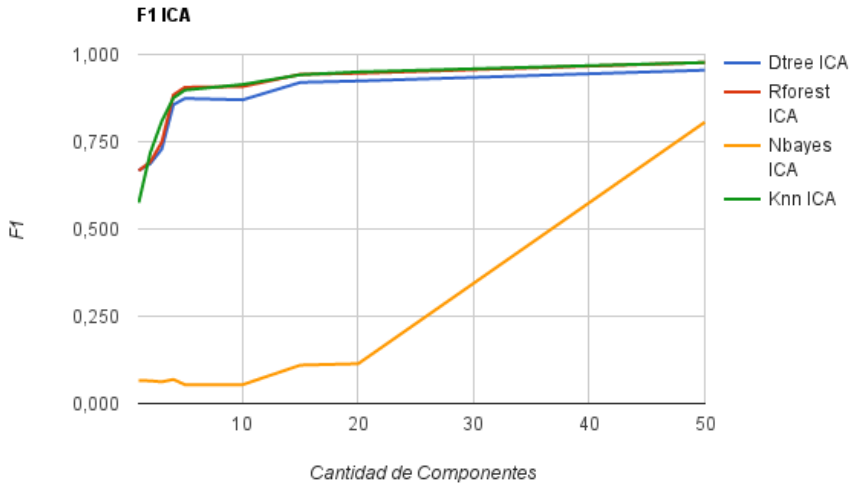
ICA Las dimensiones resultantes de la transformación de la base son estadísticamente independientes.

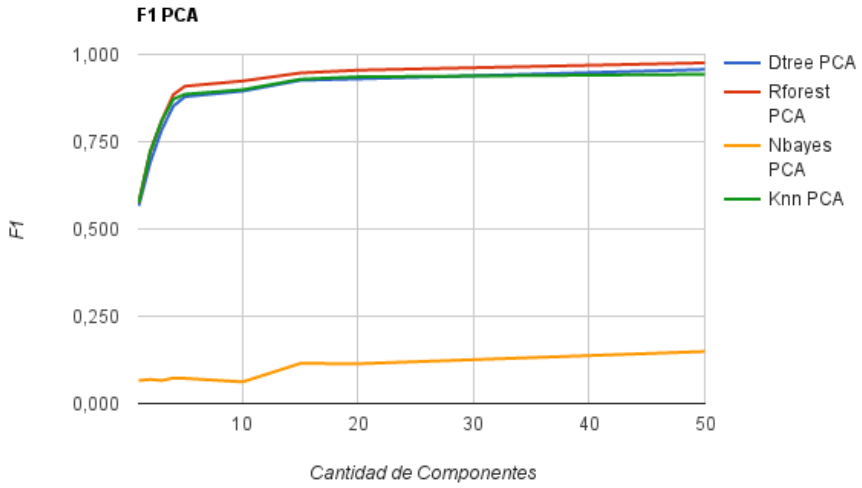
iPCA Una implementación más eficiente de PCA

REFCV Con esta ténica vimos los atributos que ofrecen mayor poder predictivo. Hace una especie de Greedy Backward elimination.

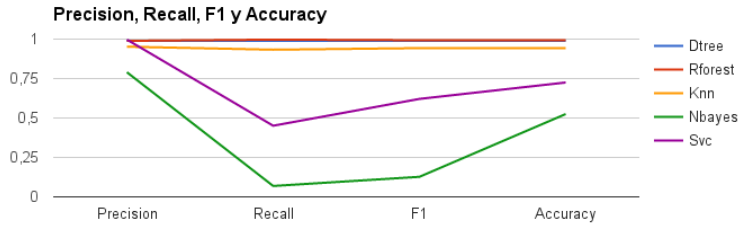
5. Resultados

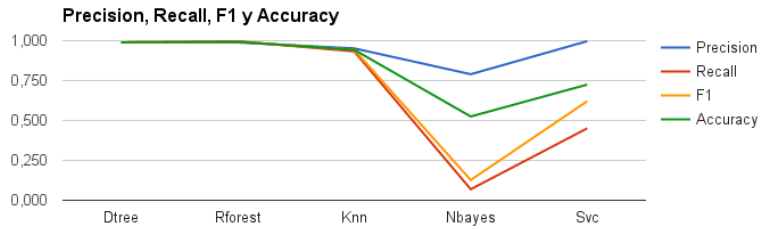
Vimos como varía la médida de F1 a médida que incrementamos la cantidad de componentes.



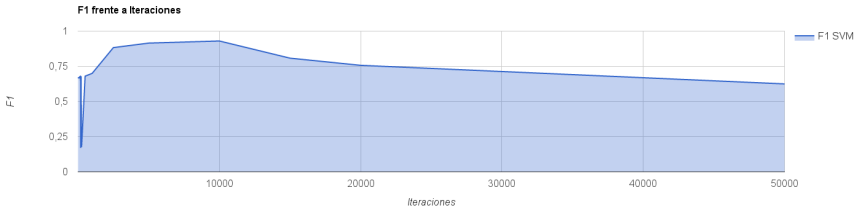


Comparamos los modelos según varias médidas de performance sobre los datos de test separados previamente (teniendo al mismo set de datos de test siempre para realizar las mediciones de forma adecuada):





A continuación mostramos un gráfico que muestra la médida de F1 en función de la cantidad de iteraciones del modelo de Support Vector Machine:



6. Discusión

A continuación podemos ver los atributos con mayor poder predictivo seleccionados con el metodo de RFCV haciendo cross validation con k = 5 y step = 1 usando como método un árbol de decisión. RFECV lo que hace es ir reduciendo la cantidad de atributos teniendo en cuenta un peso asociado que se obtiene a partir del estimador usado:

`count_ESMTP`

`count_upper`

`count_guionba`

`count_dol`

`count_arr`

`count_and`

`count_equal`

`count_dotc`

count_com

count_menor

count_menos

count_html

count_0600

count_2002

count_your

count_0800

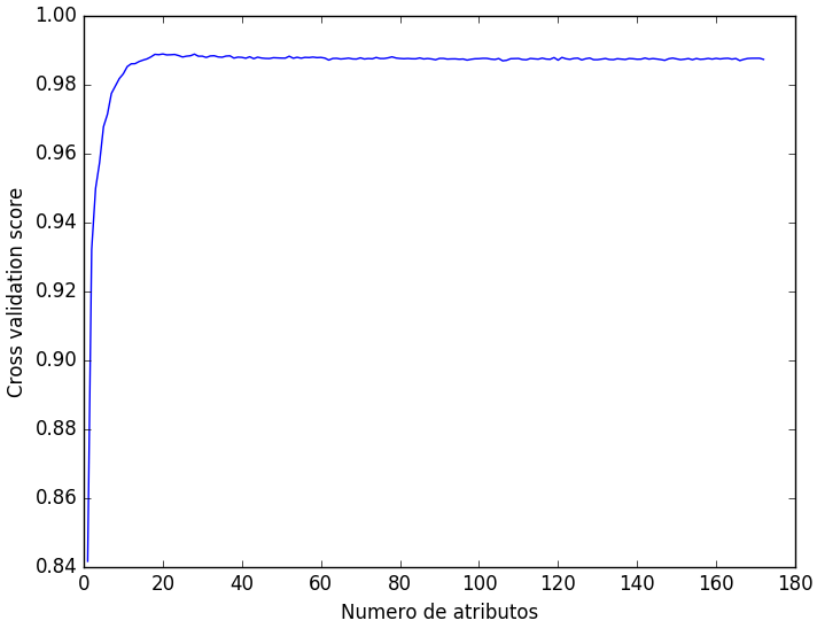
count_align

count_0500

count_smtp

count_cec

En el siguiente gráfico mostramos que con 20 atributos se puede llegar al score óptimo de cross validation.



Notamos en el gráfico que compara F1 en función de la cantidad de componentes que con pocas componentes (5 o 6) ya se puede obtener un valor igual a 0,90 y después las demás componentes (de la nueva base transformada) no mejoran demasiado la médida de performance en cuestión.

Se puede observar que el mejor modelo es el de Random Forest, teniendo todas las médidas de performance en 99 %. Igualmente el modelo basado en arboles de decisión es muy cercano en cuanto a performance, no así los modelos implementados de Naive Bayes y de Support Vector Machines. Por último el modelo de Vecinos más cercanos tiene un 94 % de accuracy. Dado que sabemos que Random Forest tuvo mejores médidas (incluso teniendo en cuenta el desvío estandard) lo elegimos como el modelo para ser usado en la competencia.

Finalmente es importante destacar que el modelo SVM mejora hasta una cierta cantidad de iteraciones (aproximadamente 10000) y luego la performance empieza a disminuir.