DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico Número 1: Spam Filter

27 de Septiembre de 2016 — Aprendizaje Automático

Integrante	LU	Correo electrónico
Abdala, Leila Yasmín	950/12	abdalaleila@gmail.com
Costa, Manuel José Joaquín	035/14	manucos94@gmail.com
Bertero, Federico Alberto		federico.bertero@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

1. Extracción de atributos

Para esta primer etapa se planteó dos tipos de atributos, automáticos y no automáticos. Los atributos no automáticos son 4. HTML, de tipo binario que indica si un mail contiene código HTML. SUBJ, de tipo binario que indica si un mail contiene el campo subject. LEN, de tipo entero que retorna la cantidad de caracteres del mail. Y finalmente SPACES, de tipo entero cuenta la cantidad de espacios de la forma " " que parecen en el mail.

Por otro lado, los atributos automáticos se obtuvieron haciendo uso de la librería SKLearn[1]. Con la misma y haciendo uso de los datos de entrada, se creó un vocabulario de palabras frecuentes, utilizando como métrica la cantidad de apariciones de cada una de ellas en ese mismo mail. Con el fin de eliminar las palabras repetidas en los headers y los conectores de lenguaje (por ejemplo The, He, Why, To, ect.), se hizo la suposición de que no es posible que una *keyword* aparezca en mas del 60 % de los mails[2].

Motivados por el fuerte impacto del código HTML en la extracción de atributos automáticos, se planteó el implementar un procedimiento para preprocesar los datos de entrada. Finalmente, dado el elevado tiempo de computo que requería esta tarea, que las herramientas que encontramos para quitar el código HTML presentaban errores en algunos casos y, la despreciable diferencia hallada entre el accuracy de los mails que fueron preprocesados y los que no, es que se determino omitir esta tarea.

En un principio se consideró un set de atributos diferente al finalmente utilizado, pero dado que se obtuvieron resultados peores en todos los casos, se decidió descartarlo. Este primer set de atributos consideraba como métrica la aparición de una palabra en el mail, usando para ello un vocabulario extraído de manera no automática[3] y clasificando como relevantes las λ palabras con mas apariciones en la base de datos de Spam.

2. Modelos

A continuación, se enuncian como items, los algoritmos de aprendizaje utilizados para experimentar y como subitems, los hiper parametros elegidos para cada uno de ellos.

- Decision Tree
  - $max\_leaf\_nodes = 100$ ($max\_leaf\_nodes \neq$ None $\rightarrow$ anula max_depth)
  - $max\_features =$ None
  - $max\_leaf\_nodes = 100$
  - $min\_samples\_split = 2$
  - $criterion =$ gini

[1] http://scikit-learn.org/stable/modules/feature_extraction.html

[2] Se llegó a este porcentaje mediante experimentación manual que no agregamos porque consideramos que no aporta al objetivo de este informe.

[3] Se llama *no automático* a todo proceso realizado íntegramente por nosotros, es decir, sin recurrir a librerías tales como SKLearn.

- Gaussian Naive Bayes
- Multinomial Naive Bayes
  - *alpha* = 0.25
  - *fit_prior* = True
- Bernoulli Naive Bayes
  - *binarize* = 0.0
  - *alpha* = 0.25
  - *fit_prior* = False
- K Nearest Neighbors
  - *n_neighbors* = 1
  - *weights* = uniform
  - *leaf_size* = 15
  - *algorithm* = kd_tree
- Support Vector Machines
- Random Forest
  - *kernel* = rbf
  - $C = 1$
  - *gamma* = 1.0

Para todos estos algoritmos se utilizó la implementación de la librería **sklearn** mientras que, para la elección de los mejores hiper parámetros, se hizo uso de la técnica denominada **Grid Search**, definiendo las grillas con valores que se consideraron adecuados para cada caso en particular.

Como se puede ver, dos clasificadores no tienen hiper parámetros asociados. En el primer caso, Gaussian Naive Bayes, se debe a que la implementación utilizada del mismo no soporta parametros. En el caso de Support Vector Machines, no se pudo realizar la experimentación completa debido a una mala implementación inicial y la gran tiempo que esta requiere.

3. Reducción de dimensionalidad

Para esta etapa, y con el fin de reducir la cantidad de atributos y mejorar la performance de los modelos, se optó por utilizar las técnicas **Select K Best** y **Principal Component Analysis**.

Select K Best es un método univariable para la selección de atributos. Es decir, es un método que luego de elegir una métrica de evaluación, aplica esta sobre cada

atributo de forma individual, genera un rating y se queda con los k mejores. Se utilizó la implementación del algoritmo de *sklearn*[4] y como parámetros de la misma, se opto por iterar sobre la cantidad de atributos a seleccionar, k, y tomar como metrica de evaluación la función *chi*2[5] ya que soporta una matriz sparse como entrada.

Con **Principal Component Analysis** se buscó proyectar los datos de entrada en un espacio de dimensión menor, maximizando la varianza de los datos proyectados. Para esta técnica, al igual que para la anterior, se hizo uso de su respectiva implementación en la librería *sklearn*[6]. El parámetro de mayor importancia en este caso es $n_components$, es decir, el numero de componentes a mantener y para su elección, se hizo uso e implemento el *criterio de Kaiser*, el cual se describe a continuación. El *criterio de Kaiser* consiste en retener solo las componentes principales (nuevos atributos formados como combinación lineal de los atributos originales) con autovalor mayor a 1.

4. Resultados

Para la evaluación de los métodos, se utiliza como métrica la media armónica con $\beta = 0,5$, $f_{0,5}$, ya se desea enfatizar la precisión. Esto es porque, al estar clasificando Spam, se considera un grave error que un mail que no entra en dicha categoría sea clasificado como tal, mientras que clasificar un Spam como Ham es aceptable. Es decir, se busca penalizar con mayor rigurosidad los casos de falsos positivos sobre los casos de verdaderos negativos.

Al comenzar, se dividió el conjunto de datos de entrada en dos conjuntos: uno para entrenamiento (training set), con el 80 % de los datos de entrada, y uno para evaluación (test set), con el 20 % de los datos de entrada. En la tabla 1 que se encuentra a continuación se puede ver el valor que se obtiene como score para cada uno de los clasificadores con los hiper parametros enunciados en la sección 2 y sin aplicar reducción de atributos, al ser entrenados sobre el training set y prediciendo sobre el test set.

Scores over test set	
Algoritmo	$f_{0,5}$
Decision Tree	0.998397578563
Gaussian Naive Bayes	0.563328795442
Multinomial Naive Bayes	0.646444592629
Bernulli	0.990909507236
KNN	0.94193989071
Random Forest	0.999622322936

Tabla 1: Valor de la función score aplicada sobre los diferentes clasificadores

[4] http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

[5] http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html

[6] http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

En la figura 1, se puede ver como varia el accuracy obtenido al aplicar diferentes clasificadores sobre datos con diferentes k atributos, seleccionados por el algoritmo Select K Best.

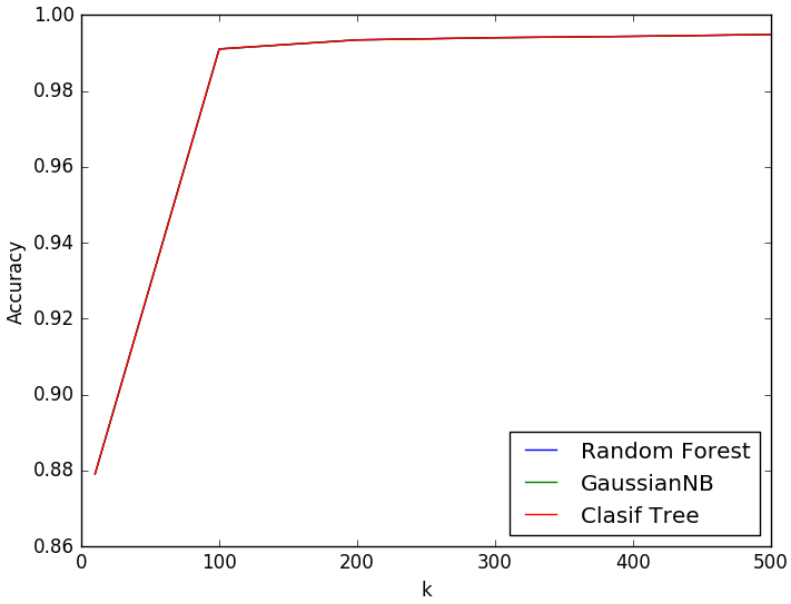

Figura 1: Variación del accuracy en relación a la selección de k atributos.

Al aplicar el criterio de Kaiser, se obtuvo que el mejor número de componentes para utilizar como parámetro del algoritmo PCA es 142.

En la figura 2, se puede comparar como varia el score $f_{0,5}$ al aplicar las dos técnicas de reducción de dimensionalidad con los parámetros que resultaron mejores y sobre cada uno de los clasificadores.

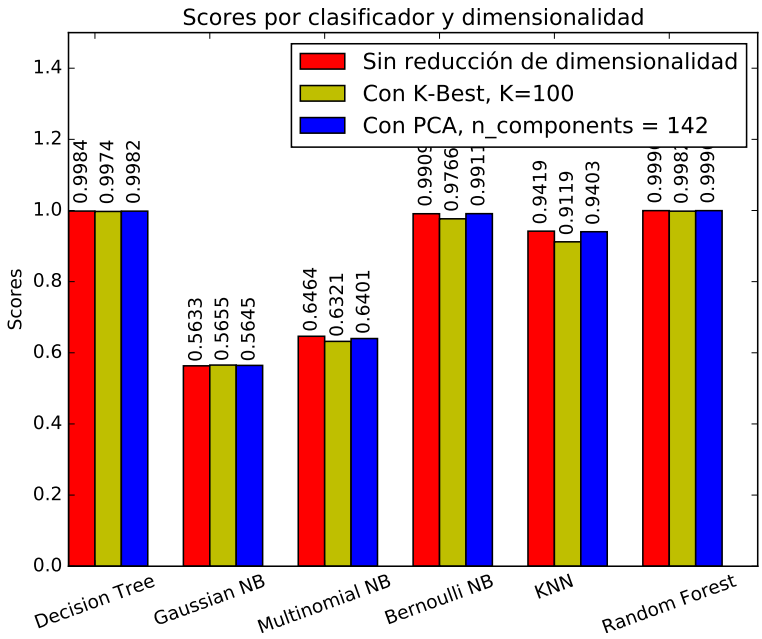

Figura 2: Variación de $f_{0,5}$ en los diferentes clasificadores al aplicar las técnicas de reducción de dimensionalidad.