Nomao Challenge

Trabajo Integrador de Especialización en Data Mining - año 2014

Moncarz, Gabriel

Revisor: Soria, Marcelo

***Resumen*—Nomao es un motor de búsqueda de lugares, donde la gente utiliza diferentes medios de comunicación (celulares, tablets, computadoras portatiles, etc) para guardar información de distintos destinos (restaurants, hoteles, bares, etc). Cada medio tiene sus características particulares y en ocaciones el mismo lugar es almacenado con datos distintos, similares o equivalentes (por ejemplo, "av." o "avenida" o "Avenue"), como también con datos erróneos o faltantes. El Desafío Nomao consiste en identificar si los datos pertenecientes a 2 destinos geográficos se refieren al mismo lugar o no. El presente trabajo hace un análisis de distintos clasificadores, para terminar proponiendo un ensamble con una capacidad predictiva superior al 98 %.**

***Index Terms*—clasificador supervisado data mining maquinas vector soporte discriminante Fisher vecinos cercano random forest regresión logística**

I. Introducción

I-A. Nota aclaratoria

El trabajo desarrollado es una ampliación del trabajo final de la materia AID de la Maestría en Data Mining en la UBA. En éste se desarrollan nuevos clasificadores: random forest de arboles CART y regresionles logísticas. También se hace un nuevo ensemble de mayor complejidad.

Ambos trabajos son desarrollos propios del mismo autor.

I-B. Sobre ALRA/Nomao Challenge

Nomao Challenge fue una competencia de Data Mining organizada por ALRA (Active Learning in Real-World Applications) y Nomao en el año 2012. Nomao [1] es un motor de búsquedas de lugares, que colecta información de diferentes fuentes (web, celulares, tablets, gps, etc). Esta información es almacenada en una base de datos interna. Cuando se realiza una consulta al motor de búsqueda, éste debe retornar una respuesta unificada. Una de las complejidades radica en el proceso de deduplicacion de datos. Este proceso es el encargado de detectar si la información de 2 fuentes distintas es asociada a un mismo lugar o no. Por ejemplo, la tabla I muestra las salidas que responden a la consulta "La poste" en Francia. El proceso de deduplicación debe identificar que el sitio 2 y 3 se refieren al mismo lugar, pero el número 1 no.

Con el propósito de clarificación se definen 2 conceptos básicos: un lugar y los atributos asociados a este. Un *lugar* hace referencia a un sitio geográfico de interes, como puede ser un cine, teatro, bar, shopping, etc. Por otro lado, los *atributos de un lugar* hacer referencia al conjunto de datos que identifícan a este sitio, por ejemplo el nombre, dirección o teléfono. Si se toma como ejemplo la tabla I, cada fila referencia a un lugar, mientras que cada columna representa un atributo de un lugar.

[1] www.nomao.com

Cuadro I
Posibles lugares a retornar por una consulta

ID	Nombre	Dirección	Teléfono
1	La poste	13 Rue De La Clef 59000 Lille France	3631
2	La poste	13 Rue Nationale 59000 Lille France	3631
3	La poste lille	13 r. nationale 59000 lille	0320313131

Los atributos de lugares provisto por los usuario son almacenados internamente. Se guarda información sobre el nombre, dirección, geolocalizacion, página web, teléfono, fax, etc. Uno de los inconvenientes es que como estos datos provienen de fuentes distintas (inclusive a veces son tipeados manualmente), 2 lugares iguales pueden tener atributos distintos, como también distintos lugares pueden tener algunos atributos iguales, como por ejemplo el nombre.

El objetivo del desafío Nomao Challenge 2012 es utilizar algoritmos de aprendizaje automático para identificar si 2 conjuntos con atributos de lugares distintos, se refieren al mismo lugar o no, teniendo en cuenta que los atributos pueden provenir de fuentes diferentes.

Las instrucciones oficiales del desafío pueden leerse en *http://fr.nomao.com/labs/challenge*.

I-C. Sobre los datos provistos por Nomao

El conjunto de datos provisto por Nomao se encuentra en *https://archive.ics.uci.edu/ml/datasets/Nomao*.

Nomao no presenta los datos crudos como estan almacenados en la base de datos ni como fueron ingresado por los usuario, **sino que cada instancia representa una comparación de 2 lugares**.

La tabla VI del apéndice A detalla todas las variables entregadas por Nomao, como también su tipo de datos. Todas las variables reales estan comprendidas en el rango de 0 a 1 inclusive. Las variables categóricas pueden tener 3 posibles valores: *'n'*, *'s'* o *'m'*. Ni la organización del desafío ni Nomao especifican el significado de las variable del dataset ni de los valores que estas pueden tomar (no se sabe que significa 'n', 's' o 'm').

El dataset contiene unas 34.465 instancias con un 28 % de datos faltantes. Los datos faltantes se debe a las limitaciones de cada fuente de datos. Por ejemplo, cuando se ingresa una dirección manualmente, el usuario no tiene capacidad de ingresar la información de GPS.

Los datos originales son transformados por Nomao y representados en 118 variables, de las cuales 89 son continuas y

29 son nominales. Además se entrega una variable adicional de identificación (id) y otra con la clase, que identifica si los atributos de ambas instancias referencian a un mismo lugar o no.

II. Materiales y métodos

El objetivo del Challenge es clasificar correctamente si 2 conjuntos de atributos de lugares referencian al mismo lugar. Para cumplir esto lo que básicamente se hace es:

- Análisis, y pre-procesamiento datos.
- Un análisis de componentes principales para mayor comprensión del problema.
- Clasificador por análisis discriminante lineal de Fisher (LDA).
- Clasificador de maquinas de vector soporte (Support Vector Machine - SVM).
- Clasificador por regresión logística.
- Clasificador por random forest.
- Ensamble con los mejores clasificadores.

Todo el procesamiento y análisis se hizo usando algoritmos propios en Python 3. Se utilizaron como soporte las siguientes librerías:

- Pandas[2]: Para usar la estructura de datos Data Frame.
- NumPy[3]: Para operaciones vectoriales.
- Scikit Learn[4]: Implementaciones del análisis de discriminante lineal, maquinas de vector soportem, análisis de componentes principales, random forest y regresión logística.
- Matplotlib: Herramienta de graficación en Python.

Todo los códigos fuentes como el trabajo base de AID, se encuentran en el siguiente repositorio público GitHub: *https://github.com/gmoncarz/nomao-challenge*

II-A. Análisis de datos y pre-procesamiento

El dataset no presenta datos fuera de rango, ya que todas las variables continuas estan dentro del dominio especificado: entre 0 y 1. Las variables categóricas también respetan el estándar: no hay ninguna que contenga un valor no especificado. Estas variable se convirtieron a variables dummies, con el objetivo de poder aplicar algoritmos que requieran variables numéricas. Las variables categóricas originales son eliminadas del dataset, dejando sólamente las dummies como entrada de los algorítmos.

Todas las variables continuas, excepto las que comienzan con el nombre *clean_name*, tienen datos faltantes. Como el rango de estas variables es de 0 a 1, todas aquellas que tienen datos faltantes se las reemplaza por el valor -1. No hay una justificación teórica de por que se escoge el valor -1, pero los clasificadores respondieron efectivamente a este valor.

II-B. Análisis de Componentes Principales

Se corre un análisis de componentes principales. Se recuerda que las variables categóricas son eliminadas del dataset como tales, y se las reemplaza por variables dummies.

[2]http://pandas.pydata.org/
[3]http://www.numpy.org/
[4]http://scikit-learn.org/

II-C. Validación Cruzada

Los métodos que se corren en las secciones siguientes pueden tener overfitting. Para tener resultados precisos evitando lo más posible el sobreentrenamiento, todos los métodos que se corren y se detallan en las secciones posteriores se realizan aplicando validación cruzada de 5 folders.

II-D. Análisis Discriminante Lineal

Se corre un análisis de discriminante lineal de Fisher. Este proceso no tiene parámetros especiales a configurar.

II-E. Maquinas de vector soporte

Las máquinas de vector soporte pueden discriminar instancias con distintos tipos de kernel, dependiendo de la naturaleza de los datos de entradas. Como la naturaleza de los datos es desconocido, y parte del desafío es identificarla, en este trabajo se realiza un análisis de 3 kernels : lineal, polinómico y sigmoide.

A su vez, la eficacia en la clasificación de cada kernel depende de los parámetros en que el clasificador es entrenado. Estos son valores empíricos que dependen exclusivamente de cada problema en particular. Es por eso que se corren varios entrenamientos con distintos kernels y distintos parámetros. La tabla II especifica todas las variaciones de maquinas de vector soporte corridas.

II-F. K-Vecinos mas cercanos

Se realiza 18 iteraciones de K-Vecinos mas cercanos, iterando entre k=3 a k=20. Se utiliza la distancia de Minkowski.

II-G. Regresion logística

Se realiza una regresión logística sobre todas las variables, sin penalización, sin balanceo de clases y sin límite de iteraciones.

II-H. Random Forest

Se realizan corridas de Random Forest de 100 arboles CART, con una profundidad máxima de 50 hojas, un mínimo de 2 instancias para abrir un nodo y sin límite máximo de nodos por hoja.

II-I. Ensamble de 3 clasificadores

Este método realiza un ensamble por votación de los siguientes 3 clasificadores corridos previamente:

- Discriminante lineal de Fisher
- Máquina de vector soporte polinómica de 3° orden y $\gamma = 10^{-1}$.
- K Vecinos mas cercanos con K=3

A diferencia de los métodos previos, este método no se aplica validación cruzada, sino que se separa un 80 % aleatorio del dataset para entrenamiento, se entrenan los 3 modelos con este conjunto, y luego se verifica la performance del ensamble con el 20 % restante.

Cuadro II
DISTINTAS CONFIGURACIONES DE SVM EJECUTADAS

Número	Kernel	γ	Grado
1	lineal		
2	polinómico	1	2
3	polinómico	0	2
4	polinómico	10^{-1}	2
5	polinómico	10^{-2}	2
6	polinómico	10^{-3}	2
7	polinómico	10^{-4}	2
8	polinómico	10^{-5}	2
9	polinómico	1	3
10	polinómico	0	3
11	polinómico	10^{-1}	3
12	polinómico	10^{-2}	3
13	polinómico	10^{-3}	3
14	polinómico	10^{-4}	3
15	polinómico	10^{-5}	3
16	polinómico	1	4
17	polinómico	0	4
18	polinómico	10^{-1}	4
19	polinómico	10^{-2}	4
20	polinómico	10^{-3}	4
21	polinómico	10^{-4}	4
22	polinómico	10^{-5}	4
23	polinómico	1	5
24	polinómico	0	5
25	polinómico	10^{-1}	5
26	polinómico	10^{-2}	5
27	polinómico	10^{-3}	5
28	polinómico	10^{-4}	5
29	polinómico	10^{-5}	5
30	sigmoide	1	
31	sigmoide	0	
32	sigmoide	10^{-1}	
33	sigmoide	10^{-2}	
34	sigmoide	10^{-3}	
35	sigmoide	10^{-4}	
36	sigmoide	10^{-5}	

II-J. Ensamble de 5 clasificadores

Este método es una extensión del ensemble anterior, agregándole 2 clasificadores: regresión logística y random forest.

III. RESULTADOS

La tabla III muestra la varianza acumulada explicada por las primeras 30 componentes principales. La figura III muestra el gráfico de dispersión de las primeras 3 componentes principales.

La tabla IV muestra la efectividad en la clasificación y los tiempos de ejecución de cada uno de los algoritmos descriptos en la sección de *Materiales*

El ensamble de 3 clasificadores tiene una performance del 97.78 %, y el de 5 clasificadores del 98.07 %.

IV. DISCUSIÓN

IV-A. Análisis de Componentes Principales

La tabla III muestra la varianza acumulada de las primeras 30 componentes principales. La tabla muestra que el 95 % de la varianza puede ser explicada con 17 componentes de las 118 variables totales del dataset. Con 30 componentes principales, se puede explicar el 99 % de la varianza.

Cuadro III
VARIANZA EXPLICADA ACUMULADA POR LAS PRIMERAS 30 COMPONENTES PRINCIPALES

Componente	Var. Acumulada.
1	0.315
2	0.457
3	0.578
4	0.653
5	0.708
6	0.757
7	0.797
8	0.832
9	0.859
10	0.884
11	0.900
12	0.913
13	0.925
14	0.935
15	0.942
16	0.947
17	**0.953**
18	0.957
19	0.962
20	0.966
21	0.969
22	0.973
23	0.976
24	0.979
25	0.981
26	0.983
27	0.985
28	0.987
29	0.989
30	**0.990**

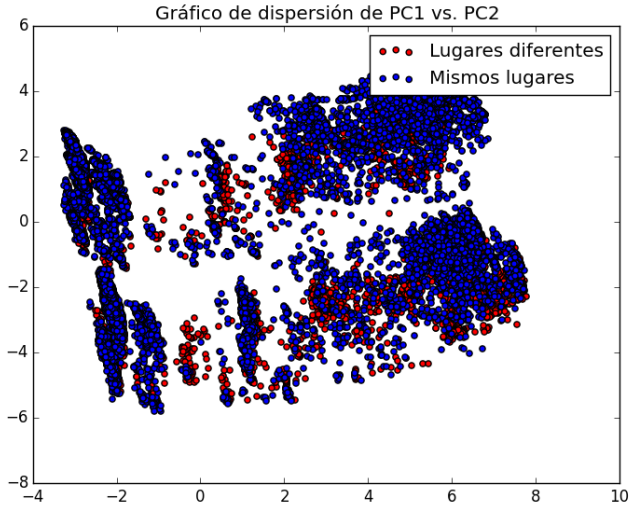
Los gráficos de la figura III muestran 3 gráficos de dispersión de las 3 primeras componentes principales. Los gráficos de la componente principal 1 contra la componente 2 y 3, muestran que no se pueden identificar las clases, ya que ambas estan distribuidas por todo el gráfico. En el gráfico de la componente principal 2 y la 3 ya se puede ver que las clases que apuntan a *"lugares diferentes"*, se encuentran más agrupadas que en las componentes previas. Sin embargo, pese al mayor poder de agrupamiento, estan muy juntas a las instancias de la clase contraria, haciendo imposible una clara identificación por componentes principales.

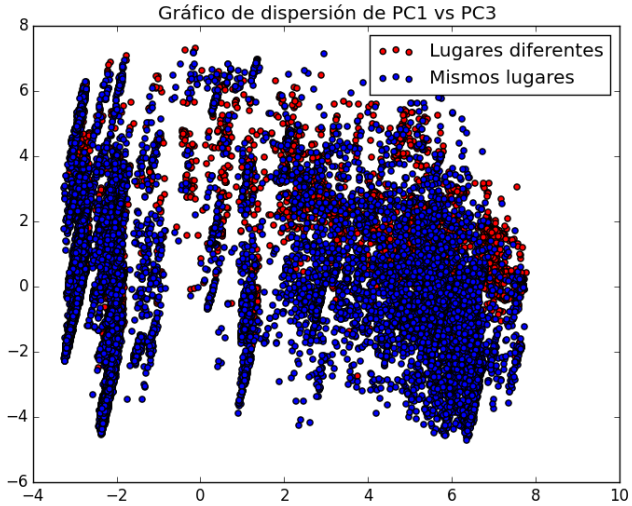
IV-B. Métodos de clasificación

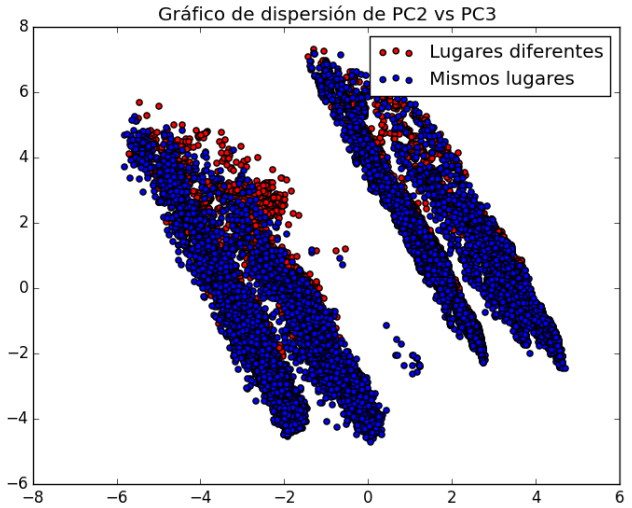
El objetivo que se persigue en este trabajo es encontrar varios clasificadores de diferentes tipos, que tengan un buen desempeño encontrando las clases: instancias que apuntan al mismo lugar y las que no. El objetivo final es apalancar el desempeño de estos clasificadores individuales haciendo un ensamble de los mejores de estos. Para esto se prueba clasificadores de diversas especies con distintos parámetros. Se prueba con discriminantes lineal de Fisher, máquinas de vector soporte, K vecinos mas cercanos, regresión logística y random forest. Se intenta en primera instancia encontrar el mejor clasificador de cada tipo. Luego, se verifica si haciendo un ensamble de los mejores, la performance general es aún mejor.

Se aclara que todas las mediciones de performance en esta sección son aplicando validación cruzada, a excepción de los ensembles.

Figura 1. Gráfico de dispersión de las componentes principales







IV-C. Análisis discriminante lineal de Fisher

El clasificador por discriminante lineal de Fisher no requieren una combinación de parámetros para escoger el mejor. Este clasificador pudo diferenciar correctamente el 94.14 % de las

Cuadro IV
PERFORMANCE CON VALIDACIÓN CRUZADA DE TODOS LOS CLASIFICADORES

Id	Clasificador	Grado	γ	Tiempo [s] Val. Cruzada	Performance c/ Val. Cruzada
1	LDA			12	**94.14**
2	SVM lineal			156	94.82
3	SVM polinómico	2	1	774	96.06
4	SVM polinómico	2	0	181	94.44
5	SVM polinómico	2	10^{-1}	138	96.13
6	SVM polinómico	2	10^{-2}	168	95.11
7	SVM polinómico	2	10^{-4}	404	92.44
8	SVM polinómico	2	10^{-4}	551	71.44
9	SVM polinómico	2	10^{-5}	487	71.44
10	SVM polinómico	3	1	1369	94.57
11	SVM polinómico	3	0	199	94.5
12	SVM polinómico	3	10^{-1}	196	**96.14**
13	SVM polinómico	3	10^{-2}	157	95.35
14	SVM polinómico	3	10^{-4}	524	71.44
15	SVM polinómico	3	10^{-4}	474	71.44
16	SVM polinómico	3	10^{-5}	477	71.44
17	SVM polinómico	4	1	2005	93.65
18	SVM polinómico	4	0	223	94.3
19	SVM polinómico	4	10^{-1}	333	95.49
20	SVM polinómico	4	10^{-2}	158	95.48
21	SVM polinómico	4	10^{-4}	549	71.44
22	SVM polinómico	4	10^{-4}	476	71.44
23	SVM polinómico	4	10^{-5}	475	71.44
24	SVM polinómico	5	1	2251	93.42
25	SVM polinómico	5	0	252	94.05
26	SVM polinómico	5	10^{-1}	450	94.99
27	SVM polinómico	5	10^{-2}	150	95.56
28	SVM polinómico	5	10^{-4}	530	71.44
29	SVM polinómico	5	10^{-4}	476	71.44
30	SVM polinómico	5	10^{-5}	473	71.44
31	SVM sigmoide		1	666	60.89
32	SVM sigmoide		0	451	71.44
33	SVM sigmoide		10^{-1}	787	60.76
34	SVM sigmoide		10^{-2}	446	72.39
35	SVM sigmoide		10^{-3}	270	92.92
36	SVM sigmoide		10^{-4}	436	90.94
37	SVM sigmoide		10^{-5}	550	71.44
38	K-nn	3		68	**94.99**
39	K-nn	4		68	94.61
40	K-nn	5		71	94.81
41	K-nn	6		70	94.58
42	K-nn	7		73	94.63
43	K-nn	8		74	94.5
44	K-nn	9		72	94.66
45	K-nn	10		74	94.53
46	K-nn	11		75	94.48
47	K-nn	12		76	94.28
48	K-nn	13		73	94.22
49	K-nn	14		76	94.1
50	K-nn	15		78	94.04
51	K-nn	16		74	93.9
52	K-nn	17		74	93.97
53	K-nn	18		78	93.83
54	K-nn	19		79	93.86
55	K-nn	20		75	93.78
56	Regresión Log.			11	**94.73**
57	Random forest			16	**96.89**
58	ensemble de 3 clasificadores				97.78
59	ensemble de 5 clasificadores				98.07

instancias. Como es el único clasificador de esta especie y su capacidad predictiva es superior al 90 %, se lo selecciona para formar parte de los ensembles.

IV-D. Máquinas de vector soporte

Las máquinas de vector soporte son buenas clasificadoras para algunos problemas complejos, permiten clasificar datos no lineales y en algunos casos tienen una perforance superior a la media. Como contrapartida, son costosas de entrenar: consumen mucha memoria y demandan muchas operaciones de CPU (respecto a otros clasificadores), y además hay que encontrar los parámetros empíricos que mejor se ajustan a los datos. Se intenta escoger el mejor modelo de maquinas de vector soporte. Se decide probar con 3 kernels distintos y combinar parámetros sobre cada uno de ellos, para escoger finalmente un modelo para formar parte del ensamble definitivo. Se prueban los kerners: lineal, polinómico y sigmoide. El modelo lineal, el más sencillo de todos y usado como referencia, puede clasificar correctamente el 94.82 % de los datos. Se prueban los kernels polinómicos de $2°$, $3°$, $4°$ y $5°$ orden. A cada uno de ellos se los corre con varios valores de γ (entre 1 y 10^{-5}) . Los modelos con buen desempeño tienen una performance que oscilan entre el 92 y 96 %. Para todos los grados el mejor γ es de 10^{-1}, excepto para el polinómico de $5°$ orden, cuyo mejor γ es de 10^{-2}.

Si se considera el mejor γ de cada kernel polinómico, todos los modelos tienen un desempeño de entre el 95 % y 96 %.

Respecto a los tiempos de ejecución, estos crecen sensiblemente a medida que el γ disminuye. Esto se debe a que mientras menor sea el γ, mayor es la precisión que se le exige al modelo y hay que realizar mas cálculos aritméticos para encontrar el polinomio discriminante.

Respecto al kernel sigmoide, solo se obtiene buenos resultados con $\gamma = 10^{-3}$ y $\gamma = 10^{-4}$. Para los demás γ, la performance es del orden del 70 %.

Tras evaluar los 36 modelos distintos de máquinas de vector soporte, se escoge el del kernel polinómico de grado 3 y $\gamma = 10^{-1}$ porque tiene una performance del 96.14 %, siendo este el mejor de todos.

IV-E. K vecinos mas cercanos

Se realizan corridas de K vecinos más cercanos, iterando con K desde 3 a 20 y distancia de Minkowski. En todos los casos se clasificaron correctamente entre el 93 % y 95 % de las instancias. Los tiempos de corrida aumentaron a medida que K aumentaba, pero siempre estos tiempos son sensiblemente mas bajos que los de máquinas de vector soporte.

Lo curioso de este clasificador es que para este problema puntual, a medida que aumenta el K, disminuye la capacidad de predicción. Posiblemente lo que sucede es que como el dataset contiene muchas variables, los datos estan en un espacio de muchas dimensiones, tendiendo a estar alejados entre si, de manera que el vecindario esta formado por puntos lejanos. De cualquier modo, la mayor diferencia por variación del K no es mayor al 1.20 %.

Se escoge el clasificador con K=3 para ser usado en el ensemble final, porque su performance del 94.99 % es la mayor de esta especie.

IV-F. Regresión logística

La regresión logística que no descarta ninguna variable tiene una capacidad predictiva del 94.73 %, equivalente a la del discriminante lineal de Fisher.

IV-G. Random forest

Random forest de por si ya es un ensemble de varios arboles CART. La performance de este algortimo es del 96.89 %, siendo el algoritmo de mayor poder de discriminación individual.

Hay que destacar que random forest no solo supero en capacidad preditiva al mejor SVM, sino que en tiempo de corrida fue ampliamente superior: Random Forest fue entrenado en 16 segundos, mientras que SVN requirió 196. De todas formas, se debe tener en cuenta que el nivel de concurrencia de random forest es superior al de SVM.

IV-H. Ensembles

La tabla V resume la performance de los 5 clasificadores bases y los ensambles. Estos últimos son construido por votos, lo que significa que la clase seleccionada es la escogida por la mayoría de los clasificadores.

Cuadro V
CLASIFICADORES INDIVIDUALES Y ENSEMBLES

Tipo	Performance
LDA	94.14
SVM polinómico	96.14
K-nn	94.99
Reg. log.	94.73
Random forest	96.89
Ensemble de 3 clasif.	97.78
Ensemble de 5 clasif.	98.07

Lo mas importante es que la eficiencia del peor de los ensemble es superior al mejor de los clasificadores individuales, inclusive que a random forest, que es un ensemble de arboles CART.

La performance del ensamble de 3 clasificadores es de un 97.78 %, superior a todos los clasificadores individuales. Usando 5 clasificadores en vez de 3, e incluyendo a random forest, el clasificador de mejor potencia predictiva, la ganancia en clasificación es marginal: 0.29 %.

Se recuerda que los ensembles son las únicas mediciones que no se realiza validación cruzada, sino que el procedimiento es separando una muestra de un 20 % para testeo y entrenando nuevamente los clasificadores del ensamble con el 80 % de entrenamiento.

Se ha intentado mejorar la performance usando mas clasificadores de máquinas vector soporte o K vecinos mas cercanos, obteniendo los mismos resultados.

V. CONCLUSIONES

Se ha logrado construir un ensemble de clasificadores capaz de identificar con un 98.07 % de precisión si los atributos de lugares de 2 registros referencian al mismo lugar o no, usando los datos transformados y provistos por el motor de búsqueda de Nomao.

AGRADECIMIENTOS

Se agradece a Marcelo Soria por el tiempo, dedicación y esfuerzo revisando, aconsejando y corrigiendo este trabajo, con el que me recibo de Especialista en Data Mining.

APÉNDICE A
ESPECIFICACIÓN DE LOS DATOS PROVISTOS POR NOMAO

Todas las variables categóricas pueden tener 3 valores posibles: *'n'*, *'s'* o *'m'*. Todas las variables continuas son reales con valores entre 0 y 1. Las variables continuas pueden tener valores faltantes.

Cuadro VI
DESCRIPCION DE LAS 118 VARIABLES

Numero	Nombre	Tipo
1	id	string
2	clean_name_intersect_min	real
3	clean_name_intersect_max	real
4	clean_name_levenshtein_sim	real
5	clean_name_trigram_sim	real
6	clean_name_levenshtein_term	real
7	clean_name_trigram_term	real
8	clean_name_including	categorica
9	clean_name_equality	categorica
10	city_intersect_min	real
11	city_intersect_max	real
12	city_levenshtein_sim	real
13	city_trigram_sim	real
14	city_levenshtein_term	real
15	city_trigram_term	real
16	city_including	categorica
17	city_equality	categorica
18	zip_intersect_min	real
19	zip_intersect_max	real
20	zip_levenshtein_sim	real
21	zip_trigram_sim	real
22	zip_levenshtein_term	real
23	zip_trigram_term	real
24	zip_including	categorica
25	zip_equality	categorica
26	street_intersect_min	real
27	street_intersect_max	real
28	street_levenshtein_sim	real
29	street_trigram_sim	real
30	street_levenshtein_term	real
31	street_trigram_term	real
32	street_including	categorica
33	street_equality	categorica
34	website_intersect_min	real
35	website_intersect_max	real
36	website_levenshtein_sim	real
37	website_trigram_sim	real
38	website_levenshtein_term	real
39	website_trigram_term	real
40	website_including	categorica
41	website_equality	categorica
42	countryname_intersect_min	real
43	countryname_intersect_max	real
44	countryname_levenshtein_sim	real
45	countryname_trigram_sim	real
46	countryname_levenshtein_term	real
47	countryname_trigram_term	real
48	countryname_including	categorica
49	countryname_equality	categorica
50	geocoderlocalityname_intersect_min	real
51	geocoderlocalityname_intersect_max	real
52	geocoderlocalityname_levenshtein_sim	real
53	geocoderlocalityname_trigram_sim	real
54	geocoderlocalityname_levenshtein_term	real
55	geocoderlocalityname_trigram_term	real
56	geocoderlocalityname_including	categorica
57	geocoderlocalityname_equality	categorica
58	geocoderinputaddress_intersect_min	real
59	geocoderinputaddress_intersect_max	real
60	geocoderinputaddress_levenshtein_sim	real
61	geocoderinputaddress_trigram_sim	real
62	geocoderinputaddress_levenshtein_term	real
63	geocoderinputaddress_trigram_term	real
64	geocoderinputaddress_including	categorica
65	geocoderinputaddress_equality	categorica
66	geocoderoutputaddress_intersect_min	real
67	geocoderoutputaddress_intersect_max	real
68	geocoderoutputaddress_levenshtein_sim	real
69	geocoderoutputaddress_trigram_sim	real
70	geocoderoutputaddress_levenshtein_term	real

Numero	Nombre	Tipo	Rango
71	geocoderoutputaddress_trigram_term	real	
72	geocoderoutputaddress_including	categorica	
73	geocoderoutputaddress_equality	categorica	
74	geocoderpostalcodenumber_intersect_min	real	
75	geocoderpostalcodenumber_intersect_max	real	
76	geocoderpostalcodenumber_levenshtein_sim	real	
77	geocoderpostalcodenumber_trigram_sim	real	
78	geocoderpostalcodenumber_levenshtein_term	real	
79	geocoderpostalcodenumber_trigram_term	real	
80	geocoderpostalcodenumber_including	categorica	
81	geocoderpostalcodenumber_equality	categorica	
82	geocodercountrynamecode_intersect_min	real	
83	geocodercountrynamecode_intersect_max	real	
84	geocodercountrynamecode_levenshtein_sim	real	
85	geocodercountrynamecode_trigram_sim	real	
86	geocodercountrynamecode_levenshtein_term	real	
87	geocodercountrynamecode_trigram_term	real	
88	geocodercountrynamecode_including	categorica	
89	geocodercountrynamecode_equality	categorica	
90	phone_diff	real	
91	phone_levenshtein	real	
92	phone_trigram	real	
93	phone_equality	categorica	
94	fax_diff	real	
95	fax_levenshtein	real	
96	fax_trigram	real	
97	fax_equality	categorica	
98	street_number_diff	real	
99	street_number_levenshtein	real	
100	street_number_trigram	real	
101	street_number_equality	categorica	
102	geocode_coordinates_long_diff	real	
103	geocode_coordinates_long_levenshtein	real	
104	geocode_coordinates_long_trigram	real	
105	geocode_coordinates_long_equality	categorica	
106	geocode_coordinates_lat_diff	real	
107	geocode_coordinates_lat_levenshtein	real	
108	geocode_coordinates_lat_trigram	real	
109	geocode_coordinates_lat_equality	categorica	
110	coordinates_long_diff	real	
111	coordinates_long_levenshtein	real	
112	coordinates_long_trigram	real	
113	coordinates_long_equality	categorica	
114	coordinates_lat_diff	real	
115	coordinates_lat_levenshtein	real	
116	coordinates_lat_trigram	real	
117	coordinates_lat_equality	categorica	
118	geocode_coordinates_diff	real	
119	coordinates_diff	real	
120	label (clase)	categorica (1 o 0).	