Nomao Challenge

Trabajo Final de AID

Maestria en Data Mining - año 2014

Moncarz, Gabriel

***Resumen*—Nomao es un motor de busqueda 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 peculiaridades y muchas veces el mismo lugar es almacenado con datos distintos, similares o equivalentes (por ejemplo, "av." o "avenida" o "Avenue"), como también con datos erroneos o faltantes. El Desafio 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 97 %.**

***Index Terms*—clasificación ensemble discriminante lineal Fisher maquina de vector soporte K-vecinos mas cercanos**

I. Introducción

I-A. 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 busquedas de lugares, que colecta información de lugares 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 busqueda, éste debe retornar una respuesta unificada. Una de las complejidades de devolver una respuesta unificada, radica en el proceso de deduplicacion de datos. Este proceso es el encargado de detectar si la información de 2 fuentes distintas son asociadas a un mismo lugar o no. Por ejemplo, la tabla I muestra los lugares 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 sitio 1 no.

Cuadro I
Posibles lugares a retornar por una consulta

ID	Nombre	Direccion	Telefono
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

Cada lugar provisto por un usuario es almacenada internamente. Se guarda información sobre el nombre, dirección, geolocalizacion, página web, telefono, fax, etc. Uno de los inconvenientes es que como los datos provienen de fuentes distintas o a veces son tipeados manualmente, lugares iguales pueden tener información distinta, como también distintos lugares pueden tener algunos campos iguales, como por ejemplo el nombre.

El objetivo del desafio Nomao Challenge 2012 es utilizar algoritmos de aprendizaje automático para identificar si distintos items con datos de lugares, se refieren al mismo lugar o no, teniendo en cuenta que estas instancias pueden provenir de fuentes diferentes.

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

I-B. 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 usario, sino que cada instancia representa una comparación de 2 lugares. Los datos originales son transformados y representados en 118 variables, de las cuales 89 son continuas y 29 son nominales. Ademas se entrega una variable adicional de identificación (id) y otra con la clase, que identifica si ambos lugares referencian a un mismo destino o no.

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.

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

II. Materiales y métodos

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

- Análisis, limpieza y pre-procesamiento datos.
- Un análisis de componentes principales para entender mayor el problema.
- Análisis de discriminante lineal de Fisher (LDA).
- Análisis cuadratico de Fisher (QDA).

[1] www.nomao.com

- Análisis de Maquinas de Vector Soporte (Support Vector Machine - SVM).
- 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 de Análisis discriminante lineal y cuadrático de Fisher, Maquinas de Vector Soporte y Análisis de componentes principales.
- Matplotlib: Herramienta de graficación en Python.

Todo los códigos fuentes se encuentran en el siguiente repositorio *GitHub: https://github.com/gmoncarz/nomao-challenge*

II-A. Análisis de datos, Limpieza y pre-procesamiento

El dataset no presenta outliers. Esto se debe a que todas las variables continuas estan en el rango de datos especificados: entre 0 y 1. Esto hace que no sea necesario eliminar ninguna instancia de datos.

Las variables categóricas también respetan el standard: no hay ninguna de estas variable que contenga un valor no especificado. Estas variable se convirtieron en variables dummies, con el objetivo por aplicar métodos que requieran variables númericas.

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 reemplazo 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 puede correr sobre todas las variables categóricas ya que estas se convirtieron en variables dummies.

II-C. Cross Validation

Los métodos que se corren en las secciones siguientes pueden tener overfitting. Para tener resultados precisos evitando lo más posible el overfitting, todos los métodos que se corren y se detallan en las secciones posteriores se realizan aplicando Cross Validation 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. Análisis Discriminante Cuadrático

Se corre un análisis de discriminante cuadrático de Fisher. Este proceso no tiene variables especiales a configurar.

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

II-F. 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 desafio es identificarla, en este trabajo se realiza un análisis de los 4 kernels más conocidos: lineal, polinómico, sigmoide y RBF (Radial Basis Function).

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 corren varios entrenamientos con distintos kernels y distintos parámetros. La tabla II especifica todas las variaciones de maquinas de vector soporte corridas.

Cuadro II
DISTINTAS CONFIGURACIONES DE SVM EJECUTADAS

Número	Kernel	γ	Degree
1	lineal		
2	rbf	1	
3	rbf	0	
4	rbf	0,1	
5	rbf	0,01	
6	rbf	0,001	
7	rbf	0,0001	
8	rbf	0,00001	
9	polinómico	1	2
10	polinómico	0	2
11	polinómico	0,1	2
12	polinómico	0,01	2
13	polinómico	0,001	2
14	polinómico	0,0001	2
15	polinómico	0,00001	2
16	polinómico	1	3
17	polinómico	0	3
18	polinómico	0,1	3
19	polinómico	0,01	3
20	polinómico	0,001	3
21	polinómico	0,0001	3
22	polinómico	0,00001	3
23	polinómico	1	4
24	polinómico	0	4
25	polinómico	0,1	4
26	polinómico	0,01	4
27	polinómico	0,001	4
28	polinómico	0,0001	4
29	polinómico	0,00001	4
30	polinómico	1	5
31	polinómico	0	5
32	polinómico	0,1	5
33	polinómico	0,01	5
34	polinómico	0,001	5
35	polinómico	0,0001	5
36	polinómico	0,00001	5
37	sigmoide	1	
38	sigmoide	0	
39	sigmoide	0,1	
40	sigmoide	0,01	
41	sigmoide	0,001	
42	sigmoide	0,0001	
43	sigmoide	0,00001	

II-G. 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-H. Ensamble

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

- Discriminante lineal de Fisher
- Máquina de vector soporte polinómica de grado 3 y $\gamma = 0{,}1$.
- K Vecinos mas cercanos con K=3

A diferencia de los metodos previos, este método no se aplica Cross Validation, 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.

III. Resultados

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

Cuadro III
Varianza explicada acumulada por las primeras 30 componentes principales

Componente	Var acumulada.
1	0.31546
2	0.457657
3	0.578424
4	0.653907
5	0.708183
6	0.757168
7	0.797937
8	0.832912
9	0.859699
10	0.884695
11	0.90015
12	0.913947
13	0.92542
14	0.935086
15	0.942277
16	0.947831
17	**0.953028**
18	0.95763
19	0.962066
20	0.966069
21	0.969952
22	0.973757
23	0.976644
24	0.979232
25	0.981546
26	0.983753
27	0.985802
28	0.987694
29	0.989397
30	**0.99089**

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

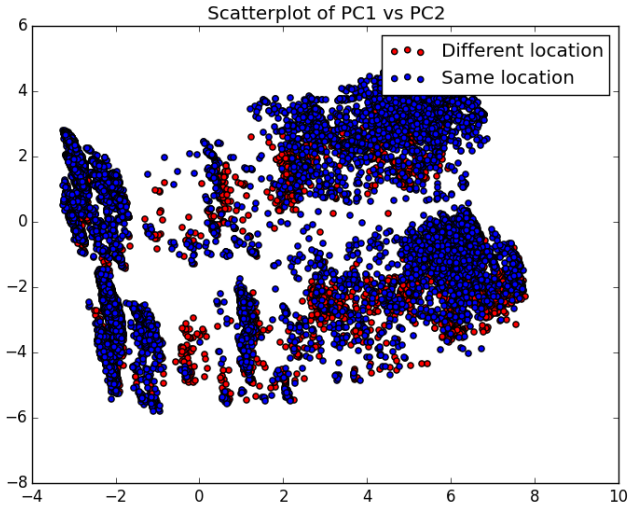
El ensamble tiene una performance del 97.78 %.

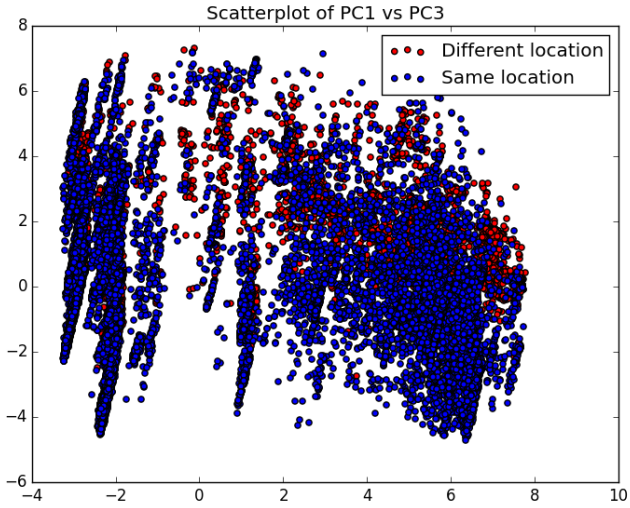
IV. Discusión

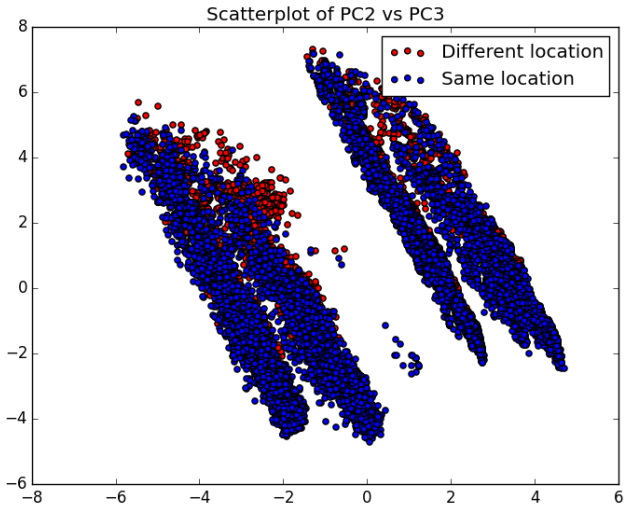
IV-A. Análisis de Componentes Principales

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

Figura 1. Scaterplot de las componentes principales







variables totales del dataset. Con 30 componentes principales, se puede explicar el 99 % de la varianza.

Los graficos de la figura III muestran 3 scratterplot de las 3 primeras componentes principales. Los graficos 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 *"distintos destinos"*, 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.

Cuadro IV
PERFORMANCE CROSSVALIDADA DE TODOS LOS CLASIFICADORES

Id	Clasificador	Degree	γ	Tiempo [s] CrossVal.	Performance CrossVal.
1	LDA			12	**94.14**
2	QDA			15	42.00
3	SVM lineal			156	94.82
4	SVM rbf		1	156	94.82
5	SVM rbf		0	154	94.82
6	SVM rbf		0,1	155	94.82
7	SVM rbf		0,01	155	94.82
8	SVM rbf		0,0001	155	94.82
9	SVM rbf		0,0001	156	94.82
10	SVM rbf		0,00001	156	94.82
11	SVM polinómico	2	1	774	96.06
12	SVM polinómico	2	0	181	94.44
13	SVM polinómico	2	0,1	138	96.13
14	SVM polinómico	2	0,01	168	95.11
15	SVM polinómico	2	0,0001	404	92.44
16	SVM polinómico	2	0,0001	551	71.44
17	SVM polinómico	2	0,00001	487	71.44
18	SVM polinómico	3	1	1369	94.57
19	SVM polinómico	3	0	199	94.5
20	SVM polinómico	3	0,1	196	**96.14**
21	SVM polinómico	3	0,01	157	95.35
22	SVM polinómico	3	0,0001	524	71.44
23	SVM polinómico	3	0,0001	474	71.44
24	SVM polinómico	3	0,00001	477	71.44
25	SVM polinómico	4	1	2005	93.65
26	SVM polinómico	4	0	223	94.3
27	SVM polinómico	4	0,1	333	95.49
28	SVM polinómico	4	0,01	158	95.48
29	SVM polinómico	4	0,0001	549	71.44
30	SVM polinómico	4	0,0001	476	71.44
31	SVM polinómico	4	0,00001	475	71.44
32	SVM polinómico	5	1	2251	93.42
33	SVM polinómico	5	0	252	94.05
34	SVM polinómico	5	0,1	450	94.99
35	SVM polinómico	5	0,01	150	95.56
36	SVM polinómico	5	0,0001	530	71.44
37	SVM polinómico	5	0,0001	476	71.44
38	SVM polinómico	5	0,00001	473	71.44
39	SVM sigmoide		1	666	60.89
40	SVM sigmoide		0	451	71.44
41	SVM sigmoide		0,1	787	60.76
42	SVM sigmoide		0,01	446	72.39
43	SVM sigmoide		0,001	270	92.92
44	SVM sigmoide		0,0001	436	90.94
45	SVM sigmoide		0,00001	550	71.44
46	K-nn	3		68	**94.99**
47	K-nn	4		68	94.61
48	K-nn	5		71	94.81
49	K-nn	6		70	94.58
50	K-nn	7		73	94.63
51	K-nn	8		74	94.5
52	K-nn	9		72	94.66
53	K-nn	10		74	94.53
54	K-nn	11		75	94.48
55	K-nn	12		76	94.28
56	K-nn	13		73	94.22
57	K-nn	14		76	94.1
58	K-nn	15		78	94.04
59	K-nn	16		74	93.9
60	K-nn	17		74	93.97
61	K-nn	18		78	93.83
62	K-nn	19		79	93.86
63	K-nn	20		75	93.78

IV-B. Métodos de clasificación

El objetivo que se persigue en este trabajo es encontrar varios clasificadores de diferentes tipos, que tengan una 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 distintos tipos de clasificadores con distintos parámetros. Se prueba con discriminantes lineal de Fisher, discriminante cuadrático, máquinas de vector soporte y K vecinos mas cercanos. 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 cross-validation, a excepción de que se mencione lo contrario.

IV-C. Análisis discriminante lineal y cuadrático de Fisher

El análisis de discriminantes lineal y cuadrático son los únicos de los probados que no requieren una combinación de parámetros para escoger el mejor. El clasificador por discriminantes de Fisher lineal pudo diferenciar correctamente el 94,14 % de las instancias. Es curioso que el clasificador cuadrático tiene una performace 42,00 %, por debajo del clasificador lineal y más aún, por debajo del azar.

Dado estos resultados, se opta por seleccional el clasificar lineal de Fisher para ser usado en el ensamble.

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, 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 4 kernels distintos y combinar parámetros sobre cáda uno de ellos, para escoger finalmente un modelo para formar parte del ensamble definitivo. Se prueban 4 kerners: lineal, RBF, polinómico y sigmoide. El modelo lineal, el más estandard de todos y usado como referencia, puede clasificar correctamente el 94.82 % de los datos. El modelo RBF, para sorpresa del autor, tiene exactamente el mismo desempeño que el lineal, en cualquier de sus variantes. Por este motivo se descarta este modelo, ya que es uno más complejo sin valor agregado.

Respecto al kernel polinómico se prueba con grado 2, 3, 4 y 5. 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 0.1, excepto para el polinómio de grado 5, cuyo mejor γ es de 0.01.

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 es el γ, mayor es la precisión que se le exige al modelo y mas calculos aritméticos hay que realizar para encontrar el polinomio discriminante.

Respecto al kernel sigmoide, solo se obtiene buenos resultados con γ iguales a 0.001 y 0.0001. Para los distintos γ, la performance era del orden del 70 %.

Tras evaluar los 43 modelos distintos de máquinas de vector soporte, se escoge el del kernel polinómico de grado 3 y $\gamma = 0{,}1$ 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 instacias. 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, dispinuye la capacidad de predicción. De cualquier modo, se debe tener en cuenta que esta diferencia no es mayor a 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. Ensemble

La tabla V resume los 3 clasificadores usados como base para el ensamble. El ensamble construido es por votos, lo que significa que la clase seleccionada es la escogida por 2 de los 3 clasificadores.

Cuadro V
CLASIFICADORES INDIVIDUALES Y ENSEMBLE

Tipo	Performance
LDA	94.14
SVM polinómico	96.14
K-nn	94.99
Ensemble	**97.78**

La performance del ensamble es de un 97.78 %, superior a todos los clasificadores individuales. Es importante aclarar que esta es la única medición que no es crossvalidada, sino que el procedimiento es separando una muestra de un 20 % para testing y entrenando nuevamente los 3 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. Es por esto que se escoge usar el ensemble mas simple como clasificador del desafio planteado.

V. CONCLUSIONES

Se ha logrado construir un ensemble de clasificadores capaz de identificar con un 97.78 % si 2 registros de ubicaciones pertenecen al mismo destino o no, usando los datos transformados y provistos por el motor de busqueda de Nomao.

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
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).