Universidad de Burgos

Escuela Politécnica Superior

Grado en Ingeniería Informática

Weka GUI

Minería en datos misteriosos

Alumnos Javier Martínez Riberas
Oscar Fernández Armengol

Tutor José Francisco Díez Pastor
Departamento de Ingeniería Civil
Área de Lenguajes y Sistemas Informáticos

Burgos, 27 de febrero de 2017

Introducción **3**

Proceso y pruebas **3**

Clasificadores 3

ZeroR 3

OneR 3

Decision table 3

M5P 3

Bagging decision table 4

J48 4

REPTree 4

Hoeffding Tree 4

Random Forest 4

Rotation Forest 5

Resultados básicos 5

Filtros 5

Fine tuning 5

Prueba 1 6

Prueba 4 6

Prueba 5 6

Prueba 7 6

Prueba 8 7

Resultados Fine tuning 7

Datos **8**

Conclusiones **9**

Introducción

Para empezar a usar la minería de datos en casos semi reales se nos ha proporcionado un dataset con 26000 instancias (2000 de ellas son para pruebas públicas).

Este dataset tiene 28 variables numéricas y una clase booleana. El set de datos está poco balanceado ya que tiene más instancias negativas que positivas. Para solucionar este problema se nos ha propuesto el uso de ciertos filtros para igualar las clases.

Todo esto se usará para reducir el error y aumentar el valor F (F-measure).

La minería de datos como tal se hará como se nos ha pedido (mediante la interfaz gráfica de Weka).

Los resultados se han colgado en github que estará publico cuando se termine la asignatura para evitar plagios, hasta entonces se ha invitado al profesor.

Proceso y pruebas

Para empezar y asentar un punto mínimo con el que poder comparar el resto de pruebas se ha probado tanto con el ZeroR como con OneR

Clasificadores

ZeroR

El ZeroR es un método simple que consiste en votar siempre la clase mayoritaria de manera que el porcentaje de los sets de datos desequilibrados siempre es mayor del 50%.
Nos sirve para decir que un clasificador peor no tiene futuro alguno

OneR

El OneR es un método que consiste en separar los datos con una sola regla. Por eso lo usaremos, al no ser tan simple como el ZeroR nos sirve para ver un punto a partir del cual podemos decir que un clasificador es malo o no.

Decision table

Dado al 'magnífico' resultado de un método tan simple como el OneR decidimos probar más métodos basados en reglas como decisión table. Este consiste en un conjunto de reglas en vez de una sola.

Como podemos ver los resultados son mejores pero no en extremo.

M5P

Se ha decidido usar M5P debido a la cantidad de los datos que eran numéricos de manera que hemos pensado que quizá la regresión era una buena opción
Este clasificador consiste en generar en cada hoja un nuevo modelo de regresión lineal ya que la clase es booleana y no podemos usar regresión sobre ella hemos usado Classification Via Regresion.

Este modelo ha sido algo decepcionante ya que no ha conseguido igualar al mejor modelo visto anteriormente (Decision table).

Bagging decision table

Dado que decision table era el mejor método hasta el momento hemos decidido probar un ensemble de decision tables. Esto supone que deberían de ser diferentes los clasificadores individuales, si no nos darán el mismo resultado que el original. Para esto solo les damos un 18% de los datos a cada uno.

El resultado es el mismo que la decisión table lo cual es malo ya que requiere más esfuerzo el construirlo y no se puede aprender nada del modelo debido al número de clasificadores.

J48

Aunque no parecía que los árboles fueran a dar mejores resultados que otros modelos decidimos probarlos por si acaso daban mejores resultados.
Este clasificador construye árboles de decisión basándose en la entropía de la información, dió unos resultados un tanto malos (peores que OneR) por lo que reafirmamos nuestras dudas sobre los árboles.

REPTree

Genera diferentes particiones, cada partición lo considera como una hoja, prediciendo la clase mayoritaria de esa hoja. También funciona en regresión, devuelve la media de cada hoja.
Usando su método de clasificación dio unos resultados mediocres. Aparte también probamos usandolo para regresión gracias a classification via regresión lo cual dio mejores resultados aunque no una maravilla.

Hoeffding Tree

Es un clasificador que construye un árbol en el cual está sujeto a cambios, si se insertan nuevas instancias el árbol irá cambiado. Dentro de los clasificadores basados en árboles es de los que mejor resultado nos ha dado.

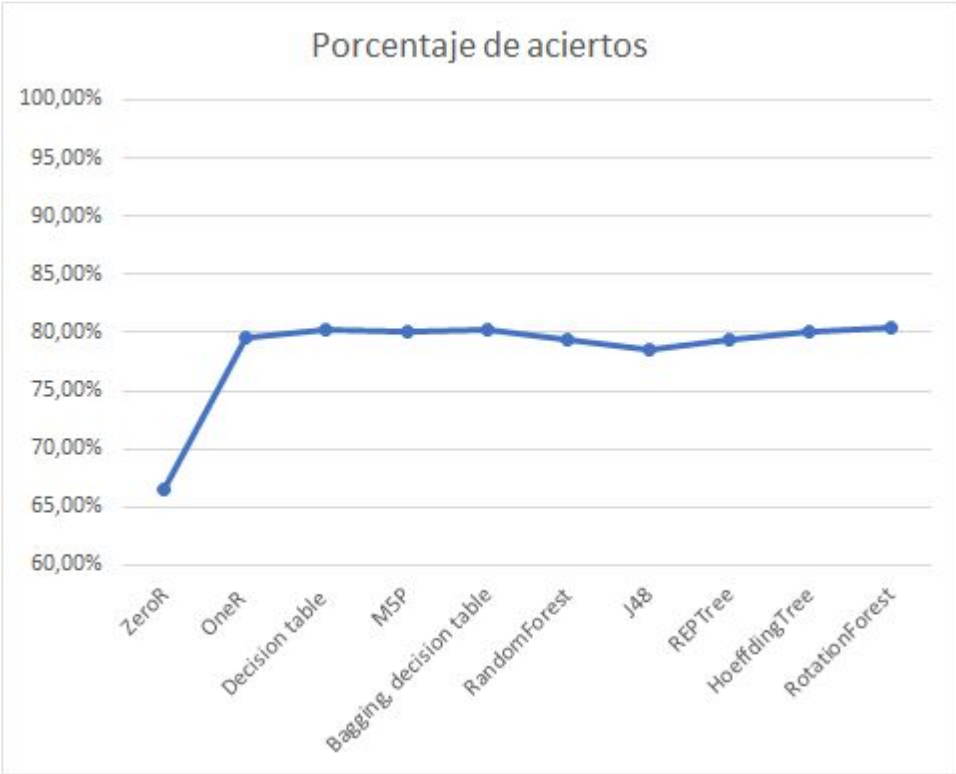
Random Forest

Este método no es un árbol simple si no un ensemble de árboles en los que no se elige el mejor atributo de cada nodo, sino, el mejor de un subconjunto aleatorio de atributos.
Los resultados son parecidos a los otros clasificadores basados en árboles

Rotation Forest

Por curiosidad se probó rotation forest (no es un método de los básicos de weka) y se vio que daba unos resultados muy buenos para lo conseguido hasta el momento.
El método consiste en dados unas features(para que no haya una varianza extrema) se rota sobre el resto de features con otros clasificadores para conseguir una mezcla de capacidad individual bastante buena con diversidad lo cual es fundamental en un ensemble.
Dió los mejores resultados por un estrecho margen.

Resultados básicos



Filtros

Tras probar varios filtros un tanto al azar se vio que para solucionar el problema del poco balance de los datos SMOTE fue lo que mejores resultados dio. Este método crea instancias artificiales de la clase minoritaria.

Fine tuning

Una vez tenemos el clasificador mejor de los que hemos encontrado con el mejor filtro hemos decidido intentar mejorar sus resultados con modificaciones de los atributos por defecto. Para bajar la cantidad de varianza del modelo se ha decidido envolverlo con bagging ya que simplemente cambiando la seed de números aleatorios (sin probar demasiadas) había más de un 1% de varianza en los resultados. Se espera reducir este margen.

Prueba 1

```
weka.classifiers.meta.FilteredClassifier -F "weka.filters.MultiFilter -F
\"weka.filters.supervised.instance.SMOTE -C 0 -K 5 -P 150.0 -S 1\"" -W
weka.classifiers.meta.Bagging -- -P 15 -print -S 1 -num-slots 3 -I 7 -W
weka.classifiers.meta.RotationForest -- -G 3 -H 3 -P 50 -F
"weka.filters.unsupervised.attribute.PrincipalComponents -R 1.0 -A 5 -M -1" -S 1 -num-slots 1 -I 10
-W weka.classifiers.meta.ClassificationViaRegression -- -W weka.classifiers.trees.M5P -- -M 4.0
```

Prueba 4

```
weka.classifiers.meta.FilteredClassifier -F "weka.filters.MultiFilter -F
\"weka.filters.supervised.instance.SMOTE -C 0 -K 5 -P 150.0 -S 1\"" -W
weka.classifiers.meta.Bagging -- -P 15 -S 1 -num-slots 3 -I 9 -W
weka.classifiers.meta.RotationForest -- -G 5 -H 7 -P 50 -F
"weka.filters.unsupervised.attribute.PrincipalComponents -R 1.0 -A 5 -M -1" -S 1 -num-slots 1 -I 10
-W weka.classifiers.meta.ClassificationViaRegression -- -W weka.classifiers.trees.M5P -- -M 4.0
```

Prueba 5

```
weka.classifiers.meta.FilteredClassifier -F "weka.filters.MultiFilter -F
\"weka.filters.supervised.instance.SMOTE -C 0 -K 5 -P 150.0 -S 1\"" -W
weka.classifiers.meta.Bagging -- -P 15 -S 1 -num-slots 3 -I 9 -W
weka.classifiers.meta.RotationForest -- -G 3 -H 7 -P 50 -F
"weka.filters.unsupervised.attribute.PrincipalComponents -R 1.0 -A 5 -M -1" -S 1 -num-slots 1 -I 10
-W weka.classifiers.meta.ClassificationViaRegression -- -W weka.classifiers.trees.M5P -- -M 5.0
```

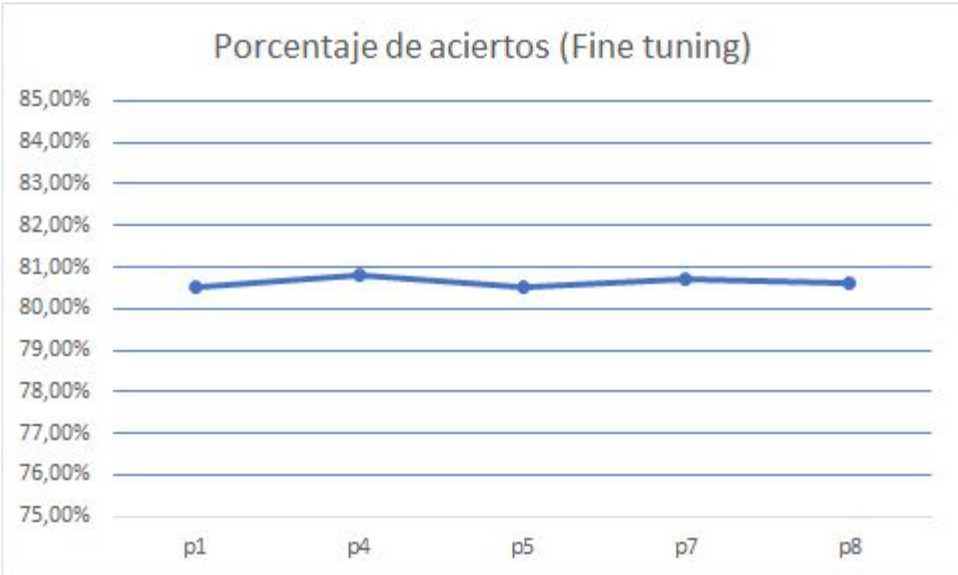
Prueba 7

```
weka.classifiers.meta.FilteredClassifier -F "weka.filters.MultiFilter -F
\"weka.filters.supervised.instance.SMOTE -C 0 -K 5 -P 150.0 -S 1\"" -W
weka.classifiers.meta.Bagging -num-decimal-places 6 -- -P 12 -S 1 -num-slots 3 -I 9 -W
weka.classifiers.meta.RotationForest -num-decimal-places 6 -- -G 4 -H 7 -P 50 -F
"weka.filters.unsupervised.attribute.PrincipalComponents -R 1.0 -A 5 -M -1" -S 1 -num-slots 1 -I 15
-W weka.classifiers.meta.ClassificationViaRegression -num-decimal-places 6 -- -W
weka.classifiers.trees.M5P -num-decimal-places 6 -- -M 4.0 -num-decimal-places 6
```

Prueba 8

```
weka.classifiers.meta.FilteredClassifier -F "weka.filters.MultiFilter -F
\"weka.filters.supervised.instance.SMOTE -C 0 -K 5 -P 150.0 -S 1\"" -W
weka.classifiers.meta.Bagging -num-decimal-places 6 -- -P 18 -S 1 -num-slots 3 -I 9 -W
weka.classifiers.meta.RotationForest -num-decimal-places 6 -- -G 4 -H 7 -P 50 -F
"weka.filters.unsupervised.attribute.PrincipalComponents -R 1.0 -A 5 -M -1" -S 1 -num-slots 1 -I 15
-W weka.classifiers.meta.ClassificationViaRegression -num-decimal-places 6 -- -W
weka.classifiers.trees.M5P -num-decimal-places 6 -- -M 4.0 -num-decimal-places 6
```

Resultados Fine tuning



Datos

Pruebas	**Porcentaje de aciertos**
ZeroR	66,50%
OneR	79,60%
Decision table	80,20%
M5P	80%
Bagging, decision table	80,20%
RandomForest	79,30%
J48	78,60%
REPTree	79,30%
HoeffdingTree	80,00%
RotationForest	80,50%

Pruebas	**Porcentaje de aciertos**
p1	80,50%
p4	80,80%
p5	80,50%
p7	80,70%
p8	80,60%

Conclusiones

Tras bastantes intentos y trabajo no se ha conseguido aumentar el porcentaje de aciertos de forma significativa sobre un clasificador tan simple como OneR lo cual es algo decepcionante por lo que nos planteamos que los datos estaban algo ofuscados como para no encontrar una solución sin poder automatizar las pruebas.

La mejor prueba que hemos conseguido es la P4 con un valor F de 0.808:

```
weka.classifiers.meta.FilteredClassifier -F "weka.filters.MultiFilter -F
\"weka.filters.supervised.instance.SMOTE -C 0 -K 5 -P 150.0 -S 1\"" -W
weka.classifiers.meta.Bagging -- -P 15 -S 1 -num-slots 3 -I 9 -W
weka.classifiers.meta.RotationForest -- -G 5 -H 7 -P 50 -F
"weka.filters.unsupervised.attribute.PrincipalComponents -R 1.0 -A 5 -M -1" -S 1 -num-slots 1 -I 10
-W weka.classifiers.meta.ClassificationViaRegression -- -W weka.classifiers.trees.M5P -- -M 4.0
```