Minería de datos usando Weka.

Rojas López José Alejandro.
alex.lnn@hotmail.com
Universidad Politécnica de Amozoc

Resumen—Este es un estudio sobre la minería de datos aplicada a los repositorios de datos incluidos al instalar el software. En éste estudio se pondrán en práctica los siguientes algoritmos incluidos en Weka: native bayes, multilayer perceptron, kstar, simpleMI, zeroR, *trees decition stump, random forest, random tree*. Es, por tanto, que se expondrán las comparaciones entre los resultados de los distintos métodos para los siguientes 3 casos: Votos (republicanos vs demócratas), lentes de contacto y crédito.

Índice de Términos— Minería de datos, repositorios de información, Weka.

I. INTRODUCCIÓN

Weka (Waikato Environment for Knowledge Analysis - *Entorno para Análisis del Conocimiento de la Universidad de Waikato*) es una plataforma de software para aprendizaje automático y minería de datos escrito en Java y desarrollado en la Universidad de Waikato. Weka es un software libre distribuido bajo licencia GNU-GPL. Gracias a la facilidad que nos brinda trabajar con una interfaz amigable, la capacidad que tiene de realizar multitareas y ser un software libre lo hace la elegida para nuestro estudio sobre otras suites gráficas como Orange o Rapid Miner.

Como sabemos la minería de datos consiste reconocer patrones en conjuntos grandes de datos, el éxito de ésta tarea determina la importancia del estudio, ya que al hallar tendencias, patrones y conjeturas parcial o totalmente concluyentes se es capaz de tomar decisiones y/o acciones en un campo o rama determinada.

Durante el estudio realizado de los casos se pudieron identificar diversos aspectos a partir de los cuales se puede ser capaz de valorar la información obtenida, cumpliendo la meta de convertir datos en información útil y fiable para de aquí tomar conclusiones y en un momento dado formular nuevas hipótesis para futuros estudios A partir de la información arrojada al final del estudio es también importante determinar y proponer alternativas para contribuir a la mejora en la calidad de la información, con el objetivo de cubrir los aspectos ya mencionados concernientes a la minería de datos, se propusieron varios parámetros en cada una de las mediciones con los distintos algoritmos para abarcar un contexto científico más amplio. Tales variaciones se explicaran a lo largo del presente artículo. Cada caso de estudio será atacado por separado exponiendo sus respectivos resultados en cada algoritmo mencionado en el resumen al inicio del documento.

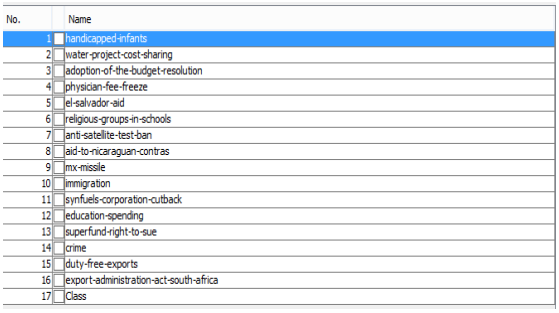
.

II. CASOS DE ESTUDIO.

A. Votaciones

En unas elecciones típicas de Estados Unidos entre demócratas y republicanos se estiman las implicaciones e impacto que los resultados de las votaciones tendrán sobre el concepto que se tiene sobre la opinión de las personas. Asimismo se observará que propuestas tuvieron mayor impacto para definir el voto de la gente por uno u otro partido.

Para este primer caso se tienen 435 instancias, es decir 435 personas a las cuales se recuperaron sus respectivas preferencias en cuanto a temas de relevancia política, los 17 atributos que se expresan en el estudio siendo el último quien indica si es demócrata o republicano. A continuación se muestran cada uno de ellos.



- Class
- adoption-of-the-budget-resolution
- aid-to-nicaraguan-contras
- anti-satellite-test-ban
- crime
- duty-free-exports
- education-spending
- el-salvador-aid
- export-administration-act-south-africa
- handicapped-infants
- immigration
- mx-missile
- physician-fee-freeze
- religious-groups-in-schools
- superfund-right-to-sue
- synfuels-corporation-cutback
- water-project-cost-sharing

Cada uno de estos atributos se puede decir que determinan que tan bien o mal percibe cada votante a su partido y así deducir que tipo de personas tienden a votar por demócratas y cuales por republicanos.

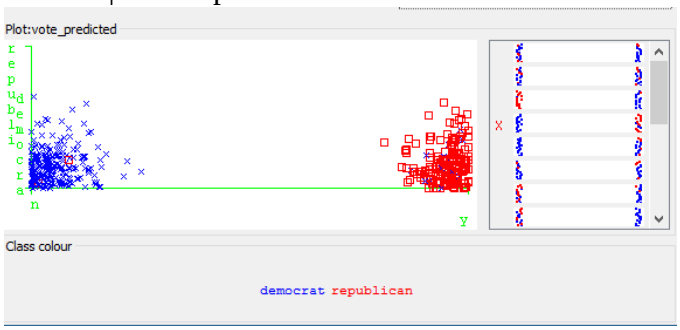
Para hacernos una idea de lo que los datos recolectados representan, veamos la siguiente gráfica usando el algoritmo zeroR:

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances	267	61.3793 %
Incorrectly Classified Instances	168	38.6207 %
Kappa statistic	0	
Mean absolute error	0.4742	
Root mean squared error	0.4869	
Relative absolute error	100 %	
Root relative squared error	100 %	
Total Number of Instances	435	

=== Confusion Matrix ===

a	b	<-- classified as
267	0	a = democrat
168	0	b = republican



Podemos observar que hay una gran cantidad de datos calculados incorrectamente, casi el 40%, por tanto recurriremos al siguiente algoritmo, el naive-bayes: obteniendo el siguiente resultado:

=== Evaluation on training set ===
=== Summary ===

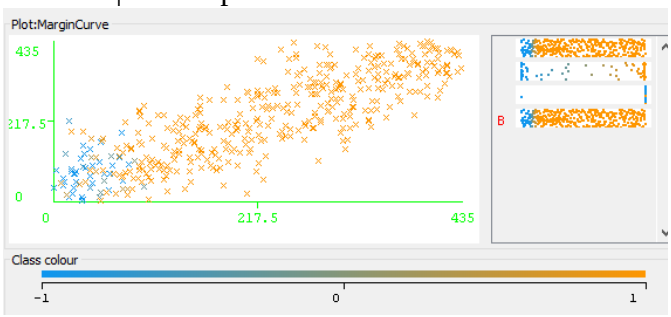
Correctly Classified Instances	393	90.3448 %
Incorrectly Classified Instances	42	9.6552 %
Kappa statistic	0.7999	
Mean absolute error	0.0975	
Root mean squared error	0.2944	
Relative absolute error	20.555 %	
Root relative squared error	60.469 %	
Total Number of Instances	435	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.891	0.077	0.948	0.891	0.919	0.974	democrat
0.923	0.109	0.842	0.923	0.881	0.974	republican
Weighted Avg. 0.903	0.089	0.907	0.903	0.904	0.974	

=== Confusion Matrix ===

```
  a   b   <-- classified as
238  29 |   a = democrat
 13 155 |   b = republican
```



Se puede ver que los valores cambian dramáticamente de manera positiva, este algoritmo nos da el 90 % de clasificaciones correctas, lo que nos indicaría unos resultados más fiables, pero el algoritmo del perceptrón multicapa nos da aún mayos exactitud como se muestra a continuación:

Time taken to build model: 2.61 seconds

=== Stratified cross-validation ===
=== Summary ===

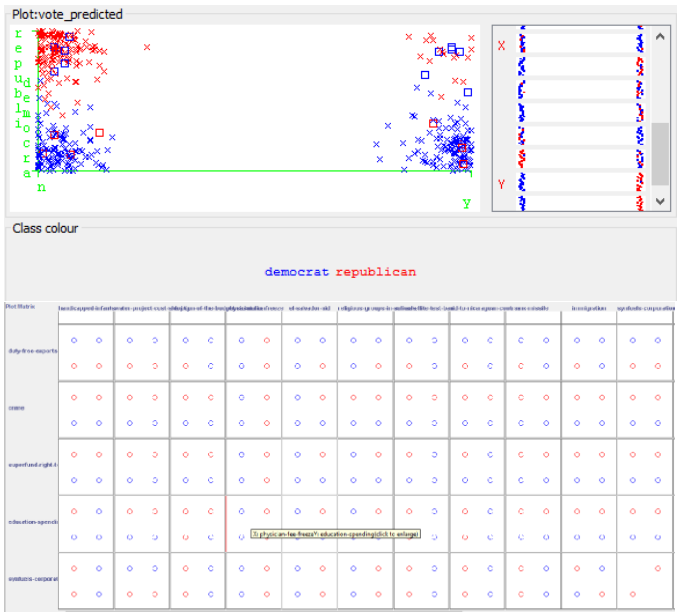
Correctly Classified Instances	412	94.7126 %
Incorrectly Classified Instances	23	5.2874 %
Kappa statistic	0.8888	
Mean absolute error	0.0528	
Root mean squared error	0.2078	
Relative absolute error	11.135 %	
Root relative squared error	42.6788 %	
Total Number of Instances	435	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.951	0.06	0.962	0.951	0.957	0.991	democrat
0.94	0.049	0.924	0.94	0.932	0.991	republican
Weighted Avg. 0.947	0.055	0.947	0.947	0.947	0.991	

=== Confusion Matrix ===

```
  a   b   <-- classified as
254  13 |   a = democrat
 10 158 |   b = republican
```



Como se aprecia la mejora es mayor, sin embargo el tiempo de construcción del modelo se cuadruplica lo cual hace más tardada la clasificación de los datos, sin embargo separa los puntos de manera muy precisa, un punto a su favor.

Procederemos a probar el algoritmo NBtree entraremos de lleno a los algoritmos basados en árboles:

Size of the tree : 17

Time taken to build model: 1.25 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	416	95.6322 %
Incorrectly Classified Instances	19	4.3678 %
Kappa statistic	0.9076	
Mean absolute error	0.0608	
Root mean squared error	0.1907	
Relative absolute error	12.8108 %	

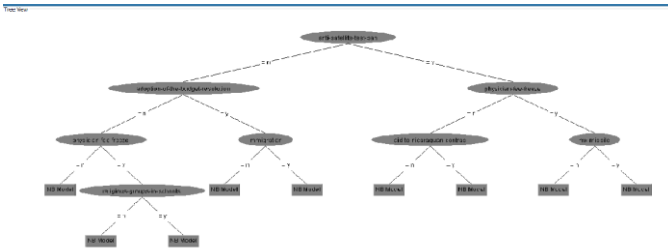
Root relative squared error 39.1664 %
Total Number of Instances 435

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.97	0.065	0.959	0.97	0.965	0.985	democrat
0.935	0.03	0.952	0.935	0.943	0.985	republican
Weighted Avg. 0.956	0.052	0.956	0.956	0.956	0.985	

=== Confusion Matrix ===

a	b	<-- classified as
259	8	a = democrat
11	157	b = republican



Ahora probemos con otro algoritmo basado en árboles:

=== Evaluation on training set ===
=== Summary ===

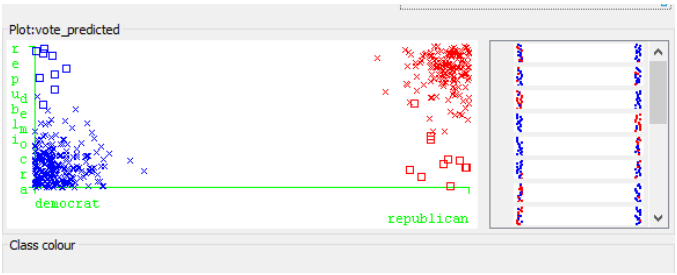
Correctly Classified Instances 423 97.2414 %
Incorrectly Classified Instances 12 2.7586 %
Kappa statistic 0.9416
Mean absolute error 0.0334
Root mean squared error 0.1328
Relative absolute error 7.0457 %
Root relative squared error 27.2726 %
Total Number of Instances 435

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.985	0.048	0.97	0.985	0.978	0.998	democrat
0.952	0.015	0.976	0.952	0.964	0.998	republican
Weighted Avg. 0.972	0.035	0.972	0.972	0.972	0.998	

=== Confusion Matrix ===

a	b	<-- classified as
263	4	a = democrat
8	160	b = republican



Nótese que la cantidad de instancias clasificadas correctamente es aún mayor, obteniendo un total de poco más del 98%, una precisión más que aceptable, prácticamente las muestras quedan totalmente separadas; además de que el tiempo es mucho menor. Por tanto tomaremos este algoritmo como el elegido para este caso.

En conclusión vemos el porcentaje de personas que votaron por los demócratas representados en azul es mayor, tal y como se muestra en la última gráfica de la figura anterior.

B. Lentes de Contacto.

En este caso se determinan las variantes que indican al médico optometrista que tipo de lente es el adecuado para el paciente. Es un estudio más sencillo pero muy interesante ya q se pueden establecer relaciones entre las diferentes variantes para elegir el tipo de lente óptimo para del paciente. Pueden elegirse 2 tipos de lentes, las variables a tomarse en cuenta son la edad, el rango entre miopía o hipermetropía, el astigmatismo y el rango de lagrimeo

Instances: 24
Attributes: 5
age
spectacle-prescrip
astigmatism
tear-prod-rate
contact-lenses

a partir del perfil de cada usuario se determina si el paciente usara un lente suave, duro o en dado caso no lo necesita.

Procedemos a analizar el primer algoritmo zeroR:

Time taken to build model: 0 seconds

=== Evaluation on training set ===
=== Summary ===

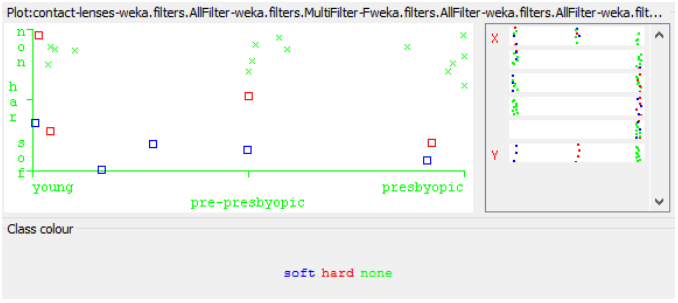
Correctly Classified Instances 15 62.5 %
Incorrectly Classified Instances 9 37.5 %
Kappa statistic 0
Mean absolute error 0.3683
Root mean squared error 0.4242
Relative absolute error 100 %
Root relative squared error 100 %
Total Number of Instances 24

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0	0	0	0	0	0.5	soft
0	0	0	0	0	0.5	hard
1	1	0.625	1	0.769	0.5	none
Weighted Avg.	0.625	0.625	0.391	0.625	0.481	0.5

=== Confusion Matrix ===

a b c <-- classified as
0 0 5 | a = soft
0 0 4 | b = hard
0 0 15 | c = none



Tal y como se observó en el caso anterior, los resultados parecen ser poco precisos al ofrecer una clasificación correcta de tan solo el 62%, lo cual hace que los resultados arrojados no sean del todo fiables. Procederemos entonces a evaluar con un par de métodos bayesianos; el naive-bayes simple y naive-bayes actualizable:

Naive-bayes:

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances 23 95.8333 %
Incorrectly Classified Instances 1 4.1667 %
Kappa statistic 0.925
Mean absolute error 0.1809
Root mean squared error 0.2357
Relative absolute error 49.1098 %

Root relative squared error 55.5663 %
Total Number of Instances 24

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure ROC Area Class
1 0.053 0.833 1 0.909 1 soft
1 0 1 1 1 1 hard
0.933 0 1 0.933 0.966 1 none
Weighted Avg. 0.958 0.011 0.965 0.958 0.96 1

=== Confusion Matrix ===

a b c <-- classified as
5 0 0 | a = soft
0 4 0 | b = hard
1 0 14 | c = none

Ahora observemos el naive-bayes actualizable:

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances 23 95.8333 %
Incorrectly Classified Instances 1 4.1667 %
Kappa statistic 0.925
Mean absolute error 0.1809
Root mean squared error 0.2357
Relative absolute error 49.1098 %
Root relative squared error 55.5663 %
Total Number of Instances 24

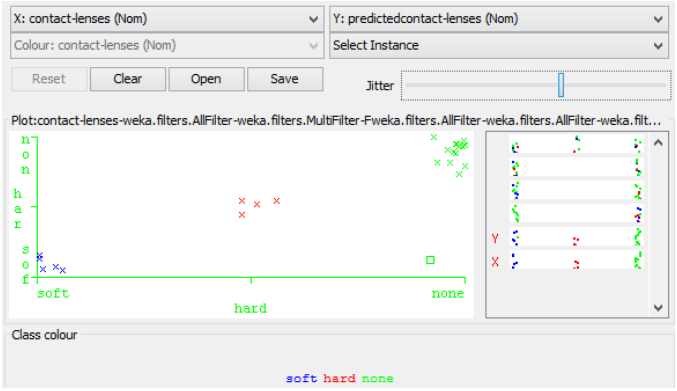
=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure ROC Area Class
1 0.053 0.833 1 0.909 1 soft
1 0 1 1 1 1 hard
0.933 0 1 0.933 0.966 1 none
Weighted Avg. 0.958 0.011 0.965 0.958 0.96 1

=== Confusion Matrix ===

a b c <-- classified as
5 0 0 | a = soft
0 4 0 | b = hard
1 0 14 | c = none

Como vemos, ambos algoritmos nos arrojan resultados prácticamente iguales, además de un total de 95.83% de clasificaciones correctas, lo que aumenta la fiabilidad de la información recuperada; a continuación una gráfica del error en los métodos bayesianos:



Podemos decir dados los parámetros observados que estos algoritmos clasifican bastante bien a comparación de otros métodos de clasificación, como por ejemplo el de tabla de decisión, el cual arroja resultados interesantes.

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances 21 87.5 %

Incorrectly Classified Instances 3 12.5 %
Kappa statistic 0.7895
Mean absolute error 0.2465
Root mean squared error 0.2917
Relative absolute error 66.9344 %
Root relative squared error 68.7605 %
Total Number of Instances 24

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
1	0.053	0.833	1	0.909	0.974	soft
1	0.1	0.667	1	0.8	0.95	hard
0.8	0	1	0.8	0.889	0.922	none
Weighted Avg. 0.875	0.028	0.91	0.875	0.878	0.938	

=== Confusion Matrix ===

a	b	c	<-- classified as
5	0	0	a = soft
0	4	0	b = hard
1	2	12	c = none

Los resultados arrojados nos dan un margen más amplio sobre que tipo de lente debe utilizar cada una de las instancias, sin embargo la cantidad de clasificaciones correctas deja mucho que desear por tanto hace difícil confiar plenamente en los datos obtenidos; cosa que no es así con los algoritmos basados en árboles como se verá mas adelante, pero antes veamos un algoritmo que no dio grandes resultados de clasificación para el caso anterior de los votos, sin embargo en esta ocasión supero con creces lo antes alcanzado; estamos hablando del perceptron multicapa:

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances 24 100 %
Incorrectly Classified Instances 0 0 %
Kappa statistic 1
Mean absolute error 0.0318
Root mean squared error 0.0433
Relative absolute error 8.64 %
Root relative squared error 10.1999 %
Total Number of Instances 24

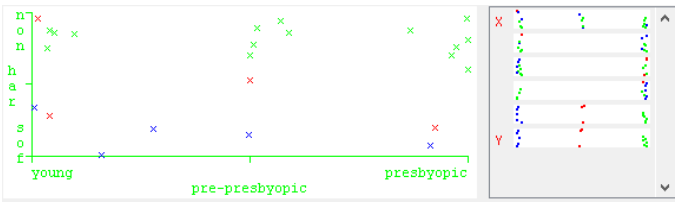
=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
1	0	1	1	1	1	soft
1	0	1	1	1	1	hard
1	0	1	1	1	1	none
Weighted Avg. 1	0	1	1	1	1	

=== Confusion Matrix ===

a	b	c	<-- classified as
5	0	0	a = soft
0	4	0	b = hard
0	0	15	c = none

El mejoramiento en el desempeño con respecto al caso anterior es sin duda notable, captando la totalidad de la información, logrando una precisión correcta con datos contundentes.



Al parecer el perceptron multicapa e excelente para casos con pocas variantes.

Finalmente no encontramos con los algoritmos

basados en árboles; ocuparemos 3 de ellos: de decisión, árbol aleatorio y bosque aleatorio; los cuales se muestran a continuación:

Decision Stump

Classifications

tear-prod-rate = reduced : none
tear-prod-rate != reduced : soft
tear-prod-rate is missing : none

Class distributions

tear-prod-rate = reduced
soft hard none
0.0 0.0 1.0
tear-prod-rate != reduced
soft hard none
0.4166666666666667 0.3333333333333333 0.25
tear-prod-rate is missing
soft hard none
0.20833333333333334 0.16666666666666666 0.625

Time taken to build model: 0 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances 17 70.8333 %
Incorrectly Classified Instances 7 29.1667 %
Kappa statistic 0.5
Mean absolute error 0.2176
Root mean squared error 0.3298
Relative absolute error 59.0782 %
Root relative squared error 77.7605 %
Total Number of Instances 24

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure ROC Area Class
1 0.368 0.417 1 0.588 0.816 soft
0 0 0 0 0 0.8 hard
0.8 0 1 0.8 0.889 0.9 none
Weighted Avg. 0.708 0.077 0.712 0.708 0.678 0.866

=== Confusion Matrix ===

a b c <-- classified as
5 0 0 | a = soft
4 0 0 | b = hard
2 0 12 | c = none

Este es el primero de los 3 algoritmos de árbol, se observa claramente que no clasificó correctamente todas las instancias; de hecho solo fue capaz de calificar un 70% de manera correcta, lo que lo descarta como nuestro algoritmo óptimo. Probemos con el árbol aleatorio:

Size of the tree : 19

Time taken to build model: 0 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances 24 100 %
Incorrectly Classified Instances 0 0 %
Kappa statistic 1
Mean absolute error 0
Root mean squared error 0
Relative absolute error 0 %
Root relative squared error 0 %
Total Number of Instances 24

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure ROC Area Class

	1	0	1	1	1	1 soft
	1	0	1	1	1	1 hard
	1	0	1	1	1	1 none
Weighted Avg.	1	0	1	1	1	1

=== Confusion Matrix ===

a	b	c	<-- classified as
5	0	0	a = soft
0	4	0	b = hard
0	0	15	c = none

La diferencia es clara, con el método de árbol aleatorio se han clasificado correcta el 100% de las instancias, lo cual nos da la certeza de datos perfectamente confiables, podemos observar claramente que los resultados son iguales a lo del perceptron multicapa y el último algoritmo a utilizar:

Random forest

=== Classifier model (full training set) ===

Random forest of 10 trees, each constructed while considering 3 random features.
Out of bag error: 0.3333

Time taken to build model: 0 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances	24	100 %
Incorrectly Classified Instances	0	0 %
Kappa statistic	1	
Mean absolute error	0.0694	
Root mean squared error	0.1258	
Relative absolute error	18.8547 %	
Root relative squared error	29.6652 %	
Total Number of Instances	24	

=== Detailed Accuracy By Class ===

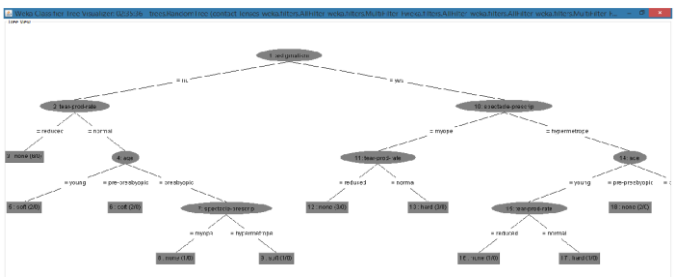
TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
1	0	1	1	1	1	soft
1	0	1	1	1	1	hard
1	0	1	1	1	1	none
Weighted Avg. 1	0	1	1	1	1	

=== Confusion Matrix ===

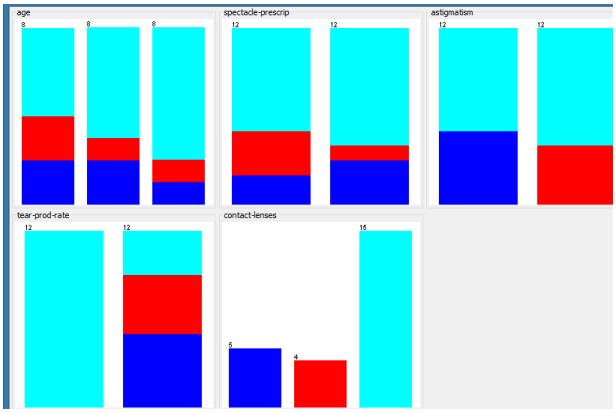
a	b	c	<-- classified as
5	0	0	a = soft
0	4	0	b = hard
0	0	15	c = none

Sin mucho que decir, este algoritmo al igual que el caso anterior obtuvo el 100% de datos calificados correctamente; la diferencia que podemos encontrar es que se tomó menos tiempo para realizar el algoritmo, por tanto se puede tomar como el algoritmo óptimo.

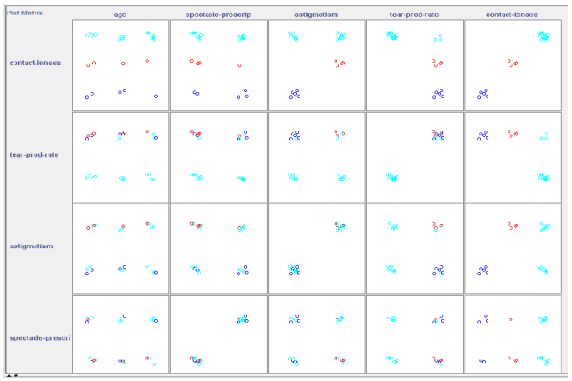
A continuación el árbol generado por weka:



Una gráfica con las distintas enfermedades, y cuántos de ellos necesita que tipo de lente:



Dadas las gráficas obtenidas podemos concluir que la mayoría de los sujetos del estudio (expresada en color agua) no necesitan aun unos lentes de contacto, asi mismo de los que si los necesitan, un porcentaje ligeramente mayor necesita lentes suaves para mejorar su visión, y por último se puede concluir que los jóvenes tienden a necesitar lentes mas suaves.



Muestras separadas.

Crédito

Este es el último caso a tratarse en este estudio, este caso trata sobre la evaluación de 1000 sujetos sobre su historial crediticio e información socio económica para determinar la instancia es sujeto de crédito o no lo es.

El caso consta de 21 atributos los cuales se definen a continuación:

Attributes:

- checking status
- duration
- credit_history
- purpose
- credit_amount
- savings_status
- employment
- installment_commitment
- personal_status
- other_parties
- residence_since
- property_magnitude
- age
- other_payment_plans
- housing
- existing_credits
- job
- num_dependents
- own_telephone
- foreign_worker
- class

Después de probar métodos bayesianos, lazy y de reglas y basados en árboles; se optó por solo incluir aquellos cuyas clasificaciones correctas fueron más amplias. Estos algoritmos son árbol aleatorio, bosque aleatorio y perceptrón multicapa, ya que la exactitud y precisión de estos 3 métodos fueron superiores al 99%. He aquí pues los resultados:

Bosque aleatorio:

Este algoritmo fue el más bajo de los 3 logrando el 99% exacto de muestras evaluadas correctamente:

Out of bag error: 0.287

Time taken to build model: 0.06 seconds

=== Evaluation on training set ===
=== Summary ===

```
Correctly Classified Instances         990               99      %
Incorrectly Classified Instances        10                1      %
Kappa statistic                          0.976
Mean absolute error                      0.1244
Root mean squared error                  0.1761
Relative absolute error                 29.601  %
Root relative squared error             38.431  %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.999    0.03      0.987     0.999     0.993      1        good
                 0.97     0.001     0.997     0.97      0.983      1        bad
Weighted Avg.    0.99     0.021     0.99      0.99      0.99       1

=== Confusion Matrix ===

   a   b   <-- classified as
 699   1 |   a = good
   9 291 |   b = bad
```

Vemos que tan solo 10 instancias no fueron evaluadas concluyentemente, un gran resultado sin duda aunque a pesar de que en los 2 primeros casos fue el algoritmo óptimo en este último caso se vio superado por los siguientes algoritmos.

Perceptrón multicapa.

```
Time taken to build model: 68.54 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances         993               99.3    %
Incorrectly Classified Instances         7                0.7    %
Kappa statistic                          0.9832
Mean absolute error                      0.012
Root mean squared error                  0.0841
Relative absolute error                  2.8505 %
Root relative squared error             18.3471 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1        0.023      0.99      1         0.995      0.982    good
                 0.977    0          1         0.977     0.988      0.982    bad
Weighted Avg.    0.993    0.016      0.993     0.993     0.993      0.982

=== Confusion Matrix ===

   a   b   <-- classified as
 700   0 |   a = good
   7 293 |   b = bad
```

Un método sumamente preciso aunque debido a la complejidad en la construcción del modelo toma un tiempo exponencialmente superior a los demás algoritmos, con un 90.3% de clasificaciones correctas, apenas 3 instancias más que el algoritmo de bosque aleatorio.

Árbol aleatorio.

El ultimo de nuestros algoritmos un árbol bastante amplio pero preciso al 100%, echemos un vistazo a la información arrojada:

Size of the tree: 1073

Time taken to build model: 0.01 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances	1000	100 %
Incorrectly Classified Instances	0	0 %
Kappa statistic	1	
Mean absolute error	0	
Root mean squared error	0	
Relative absolute error	0	%
Root relative squared error	0	%
Total Number of Instances	1000	

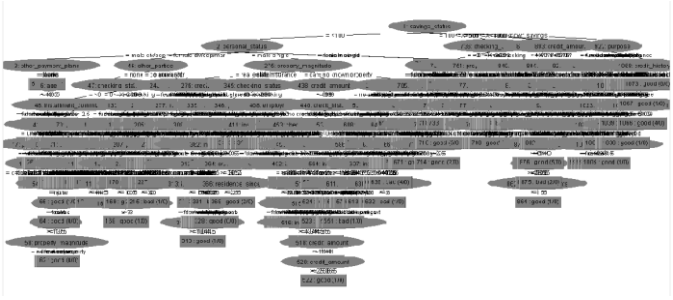
=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	0	1	1	1	1	good
	1	0	1	1	1	1	bad
Weighted Avg.	1	0	1	1	1	1	

=== Confusion Matrix ===

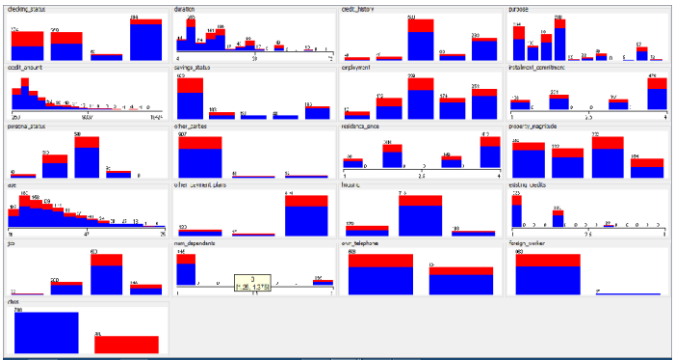
a	b	<-- classified as
700	0	a = good
0	300	b = bad

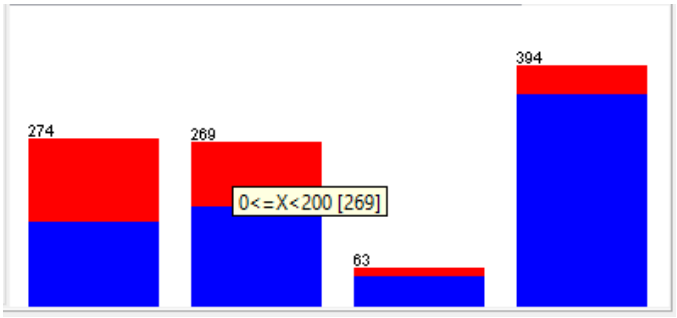
A continuación mostramos el árbol generado por weka:



Como podemos observar el árbol es bastante amplio, esto debido a la altura que el algoritmo exige, sin embargo cada nodo hace que valga el consumo de recursos ya q clasifica de manera correcta todas y cada una de las instancias, lo cual nos da la certeza de una información certera y confiable, además el tiempo tomado para construir el modelo es considerablemente menor al usado por el perceptrón multicapa, haciendo del algoritmo de árbol aleatorio el óptimo para este caso.

A continuación las gráficas de los resultados generados:





Como vemos se logró la separación de todas las muestras, producto de que nuestro algoritmo es el óptimo.

III. Conclusiones

Realizar minería de datos nos permite dar sentido a los datos que podamos tener a nuestro alcance convirtiéndolos así en información. En el mundo moderno esto nos da la ventaja sobre las problemáticas que surjan en nuestro entorno y las implicaciones e importancia que estas puedan generar son abrumadoras, es por esto que con herramientas como weka a nuestro se hace imprescindible el uso de la búsqueda de una información de calidad, certera y confiable, para ello hay que elegir y saber identificar nuestro algoritmo óptimo para cada caso específico; ya que en cada uno alguno nos servirá más que otros, haciendo la labor del analista crucial en la era de la información.

REFERENCIAS

1. Ian H. Witten; Eibe Frank, Len Trigg, Mark Hall, Geoffrey Holmes, and Sally Jo Cunningham (1999). «Weka: Practical Machine Learning Tools and Techniques with Java Implementations». *Proceedings of the ICONIP/ANZIIS/ANNES'99 Workshop on Emerging Knowledge Engineering and Connectionist-Based Information Systems* págs. 192-196. Consultado el 26-06-2007.
2. Gregory Piatetsky-Shapiro (28-06-2005). «KDnuggets news on SIGKDD Service Award 2005». Consultado el 25-06-2007.
3. «Overview of SIGKDD Service Award winners» (2005). Consultado el 25-06-2007.
4. Ian H. Witten; Eibe Frank (2005). «Data Mining: Practical machine learning tools and techniques, 2nd Edition». Morgan Kaufmann, San Francisco. Consultado el 25-06-2007.
5. G. Holmes; A. Donkin and I.H. Witten (1994). «Weka: A machine learning workbench». *Proc Second Australia and New Zealand Conference on Intelligent Information Systems, Brisbane, Australia*. Consultado el 25-06-2007.
6. S.R. Garner; S.J. Cunningham, G. Holmes, C.G. Nevill-Manning, and I.H. Witten (1995). «Applying a machine learning workbench: Experience with agricultural databases». *Proc Machine Learning in Practice Workshop, Machine Learning Conference, Tahoe City, CA, USA* págs. 14-21. Consultado el 25-06-2007.
7. P. Reutemann; B. Pfahringer and E. Frank (2004). «Proper: A Toolbox for Learning from Relational Data with Propositional and Multi-Instance Learners». *17th Australian Joint Conference on Artificial Intelligence (AI2004)*. Springer-Verlag. Consultado el 25-06-2007.

Autor
José Alejandro Rojas López. Universidad Politécnica de Amozoc, estudiante del noveno cuatrimestre de la ingeniería de Software.

2014