

Práctica 1: Análisis Predictivo Mediante Clasificación

Pablo Núñez García

13 de noviembre de 2024

Índice

1. Introducción	3
1.1. Predicción de aprobación de créditos	3
1.2. Predicción de una segunda cita	4
1.3. Predicción del tipo de enfermedad eritemato-escamosa	5
2. Procesado de datos	6
2.1. Predicción de créditos	6
2.2. Predicción de citas	7
2.3. Predicción de enfermedades	7
3. Resultados obtenidos	8
3.1. Árbol de decisión	8
3.2. ZeroR	10
3.3. Naïve Bayes	11
3.4. k-NN	12
3.5. Random Forest	13
4. Configuración de algoritmos	14
5. Análisis de resultados	18
5.1. Predicción de créditos	18
5.2. Predicción de citas	20
5.3. Predicción de enfermedades	21
5.3.1. Árbol de decisión	21
5.3.2. ZeroR	23
5.3.3. Naïve Bayes	24
5.3.4. k-NN	26
5.3.5. Random Forest	27
5.3.6. Medidas conjuntas	28
6. Interpretación de los resultados	29
7. Bibliografía	29

1. Introducción

En esta sección vemos cuáles son los problemas que vamos a tratar y cómo abordarlos.

1.1. Predicción de aprobación de créditos

Este problema está diseñado para predecir la aprobación de créditos. Según la página web que se nos facilita en la documentación de la práctica, este dataset contiene *32.581 entradas y 12 atributos*.

De estos 12 atributos destacamos **loan-status** ya que es el que nos interesa en este caso porque es el que nos dice si el crédito está aprobado o no aprobado.

Como podemos apreciar en el gráfico, las clases están **muy desbalanceadas** por lo que para evitar que se vea beneficiada la clase mayoritaria en los resultados del modelo de clasificación aplicaremos un *sobre muestreo*

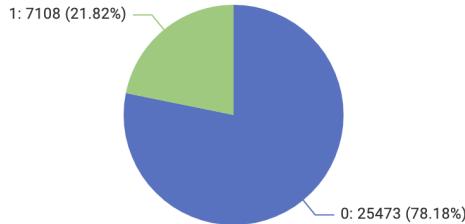


Figura 1: Pie chart de la aprobación de créditos

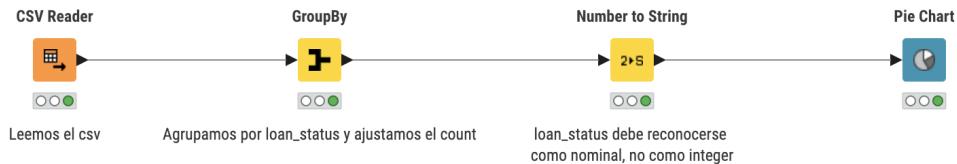


Figura 2: Flujo de la aprobación de créditos

1.2. Predicción de una segunda cita

Los datos utilizados para este problema fueron extraídos de personas que participaron en un experimento de citas rápidas desde 2002 a 2004

En este *dataset* tenemos desde atributos como Atractivo, Sinceridad, Inteligencia hasta datos de cuestionarios sacados en diferentes puntos del proceso como Hábitos diarios, Demográficos, Pensamientos.

Este dataset cuenta con *119 atributos y 8.378 entradas*

De estos 119 atributos destacamos **match** ya que es el que nos interesa en este caso porque es el que nos dice si volverían a tener otra cita o no tener otra cita

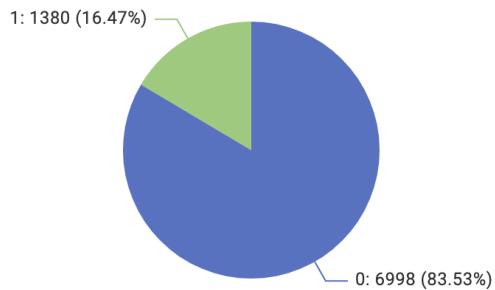


Figura 3: Pie chart de una segunda cita

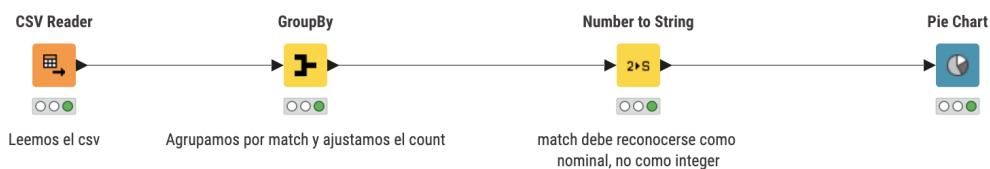


Figura 4: Flujo de la segunda cita

1.3. Predicción del tipo de enfermedad eritemato-escamosa

Este problema se centra en el diagnóstico diferencial de enfermedades eritematoescamosas. Todas comparten características clínicas pero con muy poca diferencia. Este *dataset* cuenta con *34 atributos* y *366 entradas*

De estos 34 atributos destacamos **class** ya que es el que nos interesa en este caso porque es el que nos dice que clase de enfermedad es. **1 (psoriasis)**, **2 (dermatitis seborreica)**, **3 (líquen plano)**, **4 (pitiriasis rosada)**, **5 (dermatitis crónica)**, **6 (pitiriasis rubra pilaris)**

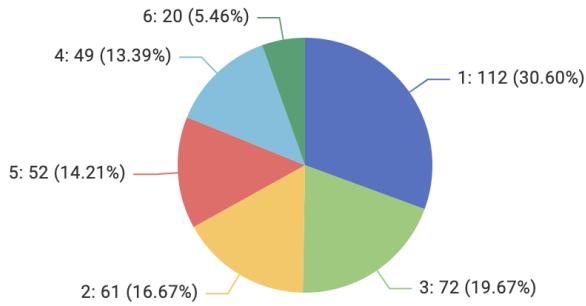


Figura 5: Pie chart de las enfermedades

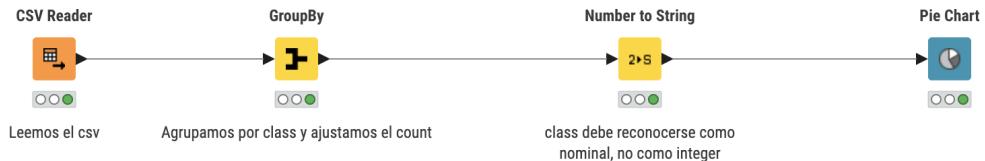


Figura 6: Flujo de las enfermedades

2. Procesado de datos

2.1. Predicción de créditos

Este dataset contiene una cantidad bastante decente de atributos, ni muchos ni pocos, además, considero que todos son necesarios por lo que he optado por no quitar ninguno.

Sin embargo, se ve a la lejana que hay alguna columna con missing values y con algun que otro valor que no es realista (p.ej una persona con 123 años)

Como vemos en la *figura 8* controlo que no pueda haber valores mayores que 40 en el atributo que representa el tiempo que lleva empleada una persona debido a que en alguna instancia había valores como 70, 123, los cuales no son realistas.

Asimismo, en la *figura 9* controlo que no pueda haber una persona con más de 70 años ya que es la edad en la que ya no se suelen conceder créditos.

Para el algoritmo k-NN en particular debo de pasar a variables numéricas y normalizar como muestra la *figura 10*

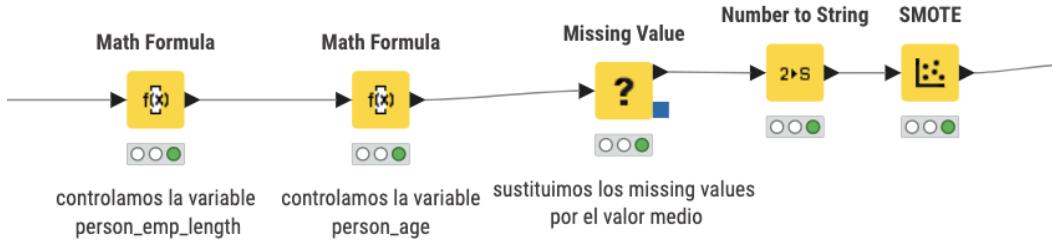


Figura 7: Flujo del procesado de predicción de créditos

```
Expression
1 if($person_emp_length$ > 40, 4.79, $person_emp_length$)
2
```

Figura 8: Control de la variable person-emp-length

```

1 if($person_age$ > 70, 28, $person_age$)
2

```

Figura 9: Control de la variable person-age

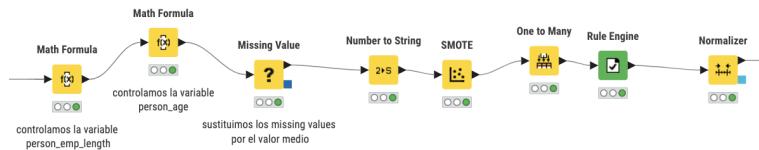


Figura 10: Flujo del procesado para K-NN de predicción de créditos

2.2. Predicción de citas

Este dataset a diferencia de los otros dos, está totalmente repleto de atributos redundantes, missing values, etc. Por esto, he decidido eliminar una gran cantidad de atributos para quedarnos finalmente con 47 de los 119 que había en un principio

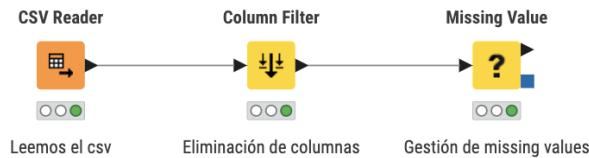


Figura 11: Flujo del procesado de predicción de citas

2.3. Predicción de enfermedades

Aquí he tenido que aplicar un tratamiento de missing values que no es muy recomendable (manual). El motivo de esta decisión ha sido que no me quedaba otra, estuve probando de todas las maneras pero no se por qué no me dejaba imputarlos. En este caso sería 'asequible' ya que solo había 8 missing values en todo el dataset. Recalco que esta decisión no es nada recomendable y ha sido porque no quedaba otra.

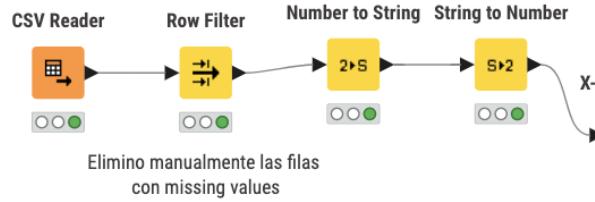


Figura 12: Flujo del procesado de predicción de enfermedad

3. Resultados obtenidos

3.1. Árbol de decisión

En la siguiente *figura(13)* vemos el flujo del algoritmo Árbol de decisión donde basicamente hacemos el particionado (5 particiones) con el **X-Partitioner**, configuramos por defecto el **Decisión Tree Learner**, en el Decisión Tree Predictor nos aseguramos de que se generen probabilidades para cada clase para posteriormente poder usarlas en la Curva ROC por ejemplo y para ver las medidas de evaluación en función de la clase.

Finalmente en el **X-Aggregator** configuramos la columna target y la de predicción

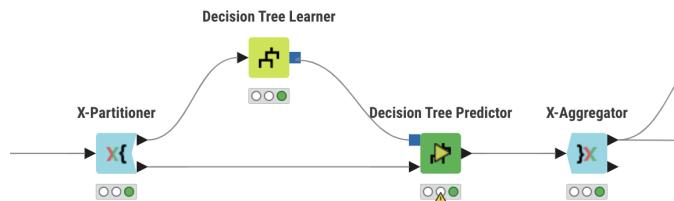


Figura 13: Flujo del algoritmo Árbol de decisión

RowID	1 Number (integer)	0 Number (integer)	RowID	1 Number (integer)	0 Number (integer)
1	21551	1587	1	5833	1249
0	3922	23886	0	1165	5749

(a) Predicción de créditos

RowID	1 Number (integer)	0 Number (integer)
1	5833	1249
0	1165	5749

(b) Predicción de citas

RowID	1 Number (integer)	2 Number (integer)	3 Number (integer)	4 Number (integer)	5 Number (integer)	6 Number (integer)
1	108	0	0	0	0	2
2	2	35	5	28	19	11
3	1	25	66	20	20	7
4	0	0	0	0	0	0
5	0	0	0	0	9	0
6	0	0	0	0	0	0

(c) Predicción de enfermedades

Figura 14: Matrices de confusión generadas por el Árbol de decisión

3.2. ZeroR

En la siguiente figura(15) vemos el flujo del algoritmo ZeroR donde hacemos el particionado con **X-Partitioner** (5 particiones), configuramos **ZeroR** por defecto y en el **Weka Predictor** añadimos las predicciones como hemos mencionado anteriormente y finalmente el **X-Aggregator**

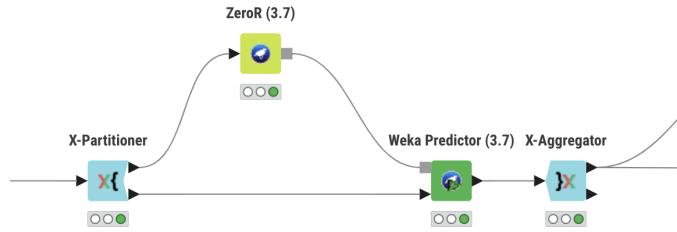


Figura 15: Flujo del algoritmo ZeroR

RowID	0 Number (integer)	1 Number (integer)	RowID	0 Number (integer)	1 Number (integer)
0	15283	15285	0	4198	4200
1	10190	10188	1	2800	2798

(a) Predicción de créditos
(b) Predicción de citas

RowID	1 Number (integer)	2 Number (integer)	3 Number (integer)	4 Number (integer)	5 Number (integer)	6 Number (integer)
1	111	60	71	48	48	20
2	0	0	0	0	0	0
3	0	0	0	0	0	0
5	0	0	0	0	0	0
4	0	0	0	0	0	0
6	0	0	0	0	0	0

(c) Predicción de enfermedades

Figura 16: Matrices de confusión generadas por el algoritmo ZeroR

3.3. Naïve Bayes

En la siguiente figura(17) vemos el flujo del algoritmo Naïve Bayes donde todos los nodos están configurados por defecto así que no explicaré más esto para no ser repetitivo.

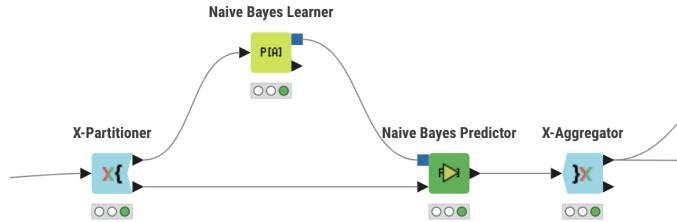


Figura 17: Flujo del algoritmo Naïve Bayes

RowID	1 Number (integer)	0 Number (integer)
1	20221	5092
0	5252	20381

(a) Predicción de créditos

RowID	1 Number (integer)	0 Number (integer)
1	5062	1637
0	1936	5361

(b) Predicción de citas

RowID	1 Number (integer)	2 Number (integer)	3 Number (integer)	4 Number (integer)	5 Number (integer)	6 Number (integer)
1	107	0	0	0	0	0
2	0	1	0	0	0	0
3	0	0	70	0	0	0
4	1	57	1	48	0	1
5	3	2	0	0	48	0
6	0	0	0	0	0	19

(c) Predicción de enfermedades

Figura 18: Matrices de confusión generadas por el algoritmo Naïve Bayes

3.4. k-NN

En la siguiente figura(19) vemos el flujo del algoritmo k-NN

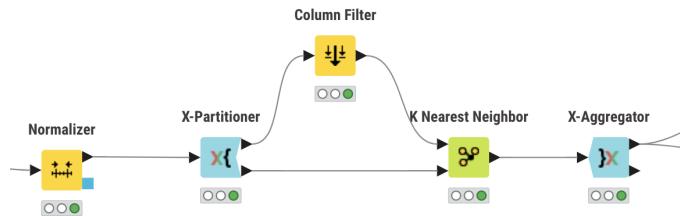


Figura 19: Flujo del algoritmo k-NN

RowID	1 Number (integer)	0 Number (integer)	RowID	0 Number (integer)	1 Number (integer)
1	21635	3806	0	4716	736
0	3838	21667	1	2282	6262

(a) Predicción de créditos

RowID	1 Number (integer)	2 Number (integer)	3 Number (integer)	4 Number (integer)	5 Number (integer)	6 Number (integer)
2	1	52	0	3	0	0
1	110	0	0	0	0	0
3	0	0	71	0	0	0
5	0	0	0	0	48	0
4	0	8	0	45	0	0
6	0	0	0	0	0	20

(b) Predicción de citas

RowID	0 Number (integer)	1 Number (integer)
0	4716	736
1	2282	6262

(c) Predicción de enfermedades

Figura 20: Matrices de confusión generadas por el algoritmo k-NN

3.5. Random Forest

En la siguiente figura(21) vemos el flujo del algoritmo Random Forest

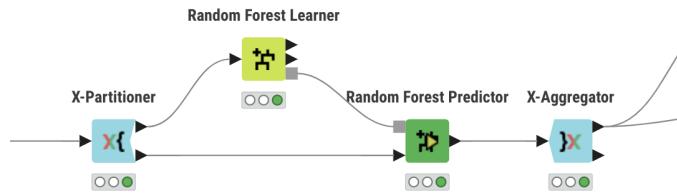


Figura 21: Flujo del algoritmo Random Forest

RowID	1	0
	Number (integer)	Number (integer)
1	23898	586
0	1575	24887

(a) Predicción de créditos
(b) Predicción de citas

RowID	1	0
	Number (integer)	Number (integer)
1	6727	469
0	271	6529

RowID	1	2	3	4	5	6
	Number (integer)					
1	111	0	0	0	0	1
2	0	56	0	4	0	0
3	0	0	71	0	0	0
4	0	4	0	44	0	0
5	0	0	0	0	48	0
6	0	0	0	0	0	19

(c) Predicción de enfermedades

Figura 22: Matrices de confusión generadas por el algoritmo Random Forest

4. Configuración de algoritmos

Para este apartado he decidido que voy a analizar el comportamiento del algoritmo Random Forest con diferentes configuraciones. En particular voy a testearlo con los diferentes **criterios de selección de características** que hemos visto en teoría

- **Information Gain**
- **Gain Ratio**
- **GINI**

Para cada uno de ellos lo ejecuto tambien variando la **profundidad del árbol** para analizar como varía la capacidad predictiva. Para ello he elegido las siguientes profundidades **(5,10,15,20)**

Además voy a analizar como responde el algoritmo (con la configuración por defecto) habiendo hecho un **oversampling** previo y sin él.

En las tablas el orden va de arriba abajo (5,10,15,20) y podemos observar como todas las medidas van mejorando y esto se debe a que el algoritmo Random Forest **cuanta más profundidad** se le proporcione **mayor capacidad de realizar divisiones específicas** en cuanto a las características se refiere lo que se traduce en una **notoria mejoría de la capacidad predictiva** del algoritmo.

Si nos paramos a ver entre los diferentes criterios de selección de variables vemos como el que mejor valores de Precisión y Specificity proporciona con una **profundidad menor** sería **GINI** ya que este prioriza particiones que reducen rápido la impureza, lo que es efectivo en configuraciones donde el modelo no tiene la capacidad de profundizar demasiado mientras que conforme aumenta la profundidad tendremos que decantarnos más por las otras dos opciones. Hay que decir también que hay que tener cuidado con la profundidad ya que cuanto **mayor profundidad mayor riesgo de sobreajuste**.

RowIndex	Algoritmo	Recall	Precision	Sensitivity	Specificity	F-measure
		Number (double)				
1	RandomForest	0.918	0.785	0.912	0.812	0.846
0	RandomForest	0.812	0.93	0.812	0.912	0.867
1_dsp	RandomForest	0.914	0.786	0.914	0.817	0.847
0_dsp	RandomForest	0.911	0.779	0.911	0.914	0.903
1_du..	Random Forest	0.979	0.828	0.979	0.851	0.897
0_du..	Random Forest	0.851	0.982	0.851	0.979	0.912
1_du..	Random Forest	0.979	0.889	0.979	0.899	0.932
0_du..	Random Forest	0.899	0.981	0.899	0.979	0.938

(a) Medidas con Information Gain

RowIndex	Algoritmo	Recall	Precision	Sensitivity	Specificity	F-measure
		Number (double)				
1	RandomForest	0.916	0.787	0.916	0.814	0.847
0	RandomForest	0.814	0.928	0.813	0.916	0.867
1_dsp	RandomForest	0.909	0.784	0.909	0.819	0.847
0_dsp	RandomForest	0.919	0.775	0.919	0.869	0.849
1_du..	Random Forest	0.979	0.805	0.979	0.835	0.884
0_du..	Random Forest	0.835	0.983	0.835	0.979	0.903
1_du..	Random Forest	0.978	0.845	0.978	0.864	0.907
0_du..	RandomForest	0.864	0.981	0.864	0.978	0.919

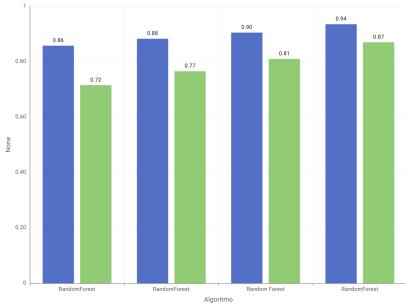
(b) Medidas con GainRatio

RowIndex	Algoritmo	Recall	Precision	Sensitivity	Specificity	F-measure
		Number (double)				
1	RandomForest	0.916	0.787	0.916	0.813	0.847
0	RandomForest	0.813	0.928	0.813	0.916	0.867
1_dsp	RandomForest	0.972	0.793	0.972	0.825	0.873
0_dsp	RandomForest	0.825	0.977	0.825	0.972	0.895
1_du..	Random Forest	0.977	0.84	0.977	0.859	0.903
0_du..	Random Forest	0.859	0.98	0.859	0.977	0.916
1_du..	RandomForest	0.976	0.901	0.976	0.908	0.937
0_du..	RandomForest	0.908	0.978	0.908	0.976	0.942

(c) Medidas con GINI

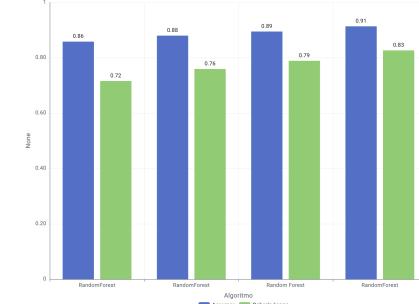
En cuanto al **Accuracy** y **Kappa** vemos como también aumenta a la par que la profundidad con lo que podríamos decir que están '*correlacionadas positivamente*'

Bar Chart Information Gain



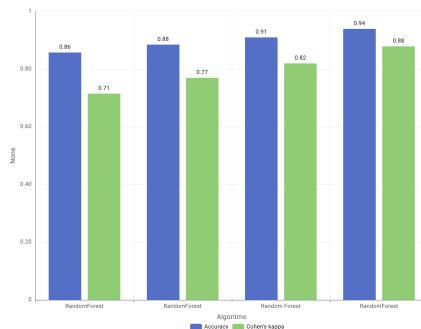
(a) Accuracy y Kappa con Information Gain

Bar Chart GainRatio



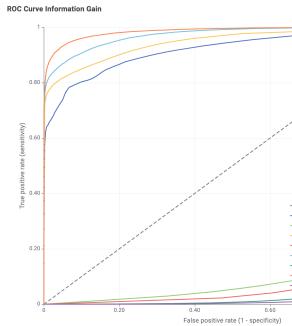
(b) Accuracy y Kappa con Gain Ratio

Bar Chart GINI

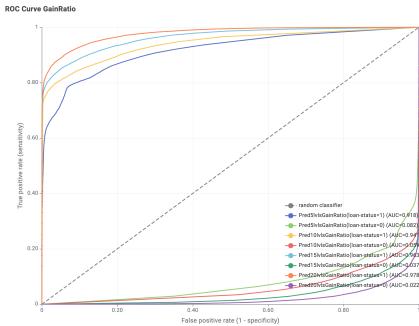


(c) Accuracy y Kappa con GINI

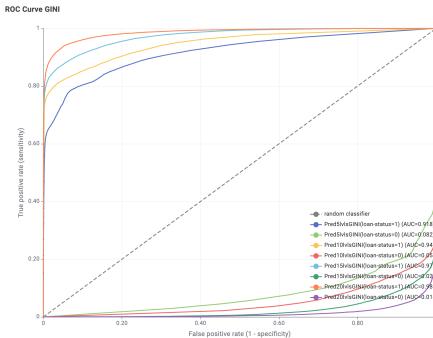
En cuanto a las curvas se refiere, vemos que no hay una variación significativa entre las diferentes configuraciones.



(a) Curvas ROC



(b) Curvas ROC con Gain Ratio

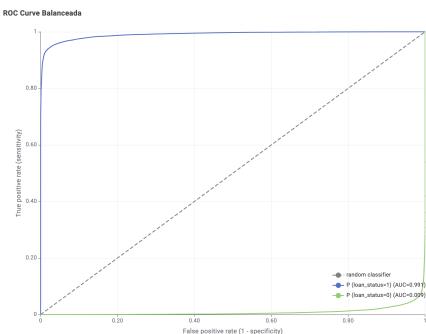


(c) Curvas ROC con GINI

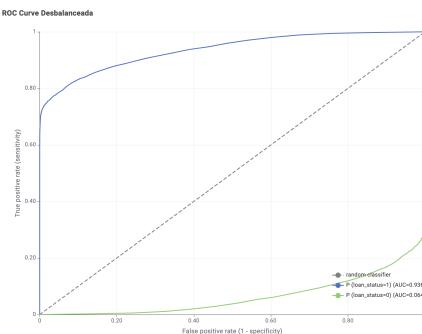
Analizando las diferencias entre los resultados con el oversampling previo y sin él nos damos cuenta como en términos generales mejora siempre con el oversampling.

Salta a la vista la diferencia en la Precisión que pasa de 0.721 a 0.939 realizándolo.

También la medida Kappa se ve bastante afectada al hacer el balanceo pasando de 0.79 a 0.916 lo que indica que se aumenta el equilibrio y consistencia en ambas clases.



(a) Curva ROC Balanceada



(b) Curva ROC Sin Balanceo

RowID	Algoritmo	Recall Número (double)	Precision Número (double)	Sensibility Número (double)	Specificity Número (double)	F-measure Número (double)
1	RandomForest	0.977	0.939	0.977	0.941	0.957
0	RandomForest	0.941	0.978	0.941	0.977	0.959

(a) Medidas Balanceadas

RowID	Algoritmo	Recall Número (double)	Precision Número (double)	Sensibility Número (double)	Specificity Número (double)	F-measure Número (double)
1	RandomForest	0.974	0.721	0.974	0.928	0.829
0	RandomForest	0.928	0.995	0.928	0.974	0.96

(b) Medidas Sin Balanceo

RowID	Algoritmo	Accuracy Número (double)	Cohen's kappa Número (double)
Over...	RandomForest	0.958	0.916

(a) Accuracy y Kappa Balanceadas

RowID	Algoritmo	Accuracy Número (double)	Cohen's kappa Número (double)
Over...	RandomForest	0.935	0.79

(b) Accuracy y Kappa Sin Balanceo

Las conclusiones que puedo sacar tras haber probado las diferentes configuraciones son las siguientes:

Para maximizar la Precisión sin sacrificar el Recall tendría que usar una configuración con una mayor profundidad (15 puede ser suficiente) y el criterio Information Gain si estoy trabajando con un dataset con una complejidad notable.

Sin embargo, si lo que necesito es un modelo más general y eficiente computacionalmente hablando me decantaría por un profundidad menor (5) y el criterio GINI aunque si elijo esta configuración soy consciente de que estaré perdiendo nivel de detalle en los datos

Ambas opciones optaría por realizarles el balanceo correspondiente para mejorar la clasificación de ambas clases y un rendimiento global

5. Análisis de resultados

5.1. Predicción de créditos

En base a los resultados de las figuras 29, 30 y 31 podemos ver como los que mejor desempeño tienen son los algoritmos basados en árboles (Árbol de decisión y Random Forest) con una superioridad algo notable de Random Forest debido a que es un algoritmo de tipo ensemble que combina varios árboles y le permite lograr una mayor precisión y capacidad de generalización.

Seguidamente tendríamos el algoritmo k-NN que también tiene un buen rendimiento ya que trabaja bien con datasets con pocas variables como es el caso de este

Naïve Bayes tiene un rendimiento algo menor que k-NN pero también aceptable, lo que le limita es que asume la independencia de las variables y que es un algoritmo bastante simple

Por último tenemos el algoritmo ZeroR que simplemente clasifica la clase mayoritaria y básicamente lo usamos como referencia

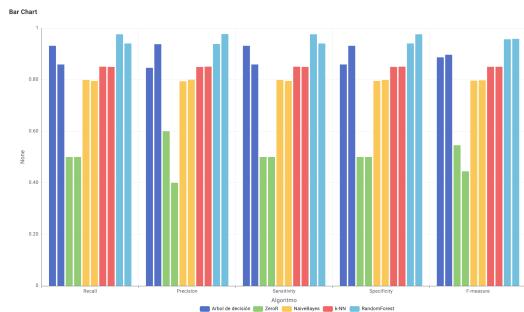


Figura 29: Medidas de los diferentes algoritmos para la predicción de creditos

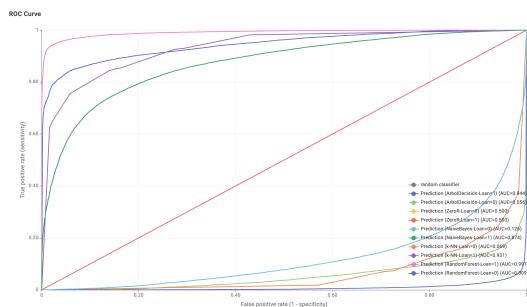


Figura 30: Curvas ROC de los diferentes algoritmos para la predicción de creditos

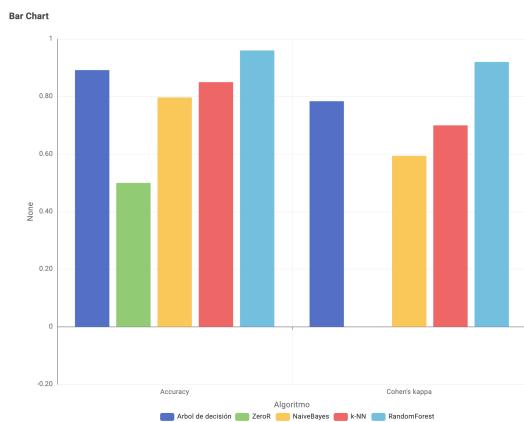


Figura 31: Accuracy y Kappa de los diferentes algoritmos para la predicción de creditos

5.2. Predicción de citas

En este problema tenemos el mismo orden de desempeño en cuanto a los algoritmos se refiere pero con un claro descenso en el rendimiento general del arbol de decisión debido a que el número de variables aumenta considerablemente y por este mismo motivo también se ve afectada la precisión del algoritmo k-NN aunque en menor medida Los demás algoritmos funcionan más o menos igual en ambos datasets

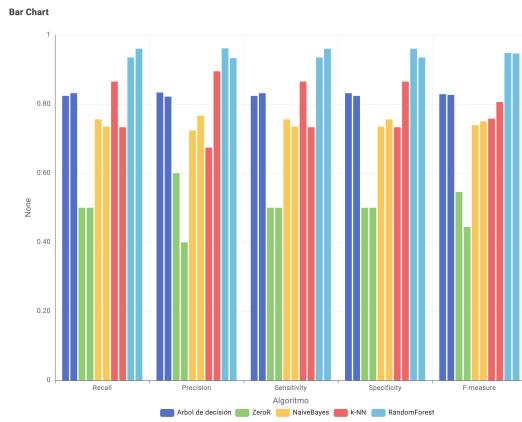


Figura 32: Medidas de los diferentes algoritmos para la predicción de citas

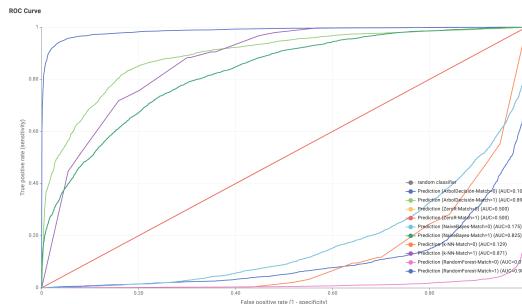


Figura 33: Curvas ROC de los diferentes algoritmos para la predicción de citas

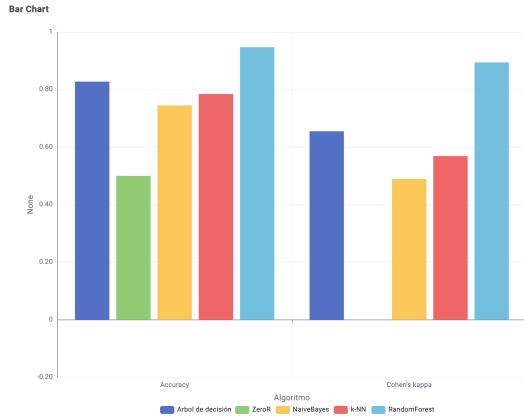


Figura 34: Accuracy y Kappa de los diferentes algoritmos para la predicción de citas

5.3. Predicción de enfermedades

Nos enfrentamos ahora a un problema multiclase por lo que vamos a añadir también las matrices de confusión para hacernos una idea más específica para cada clase e iremos algoritmo a algoritmo porque no podemos juntar todo en una tabla o gráfico

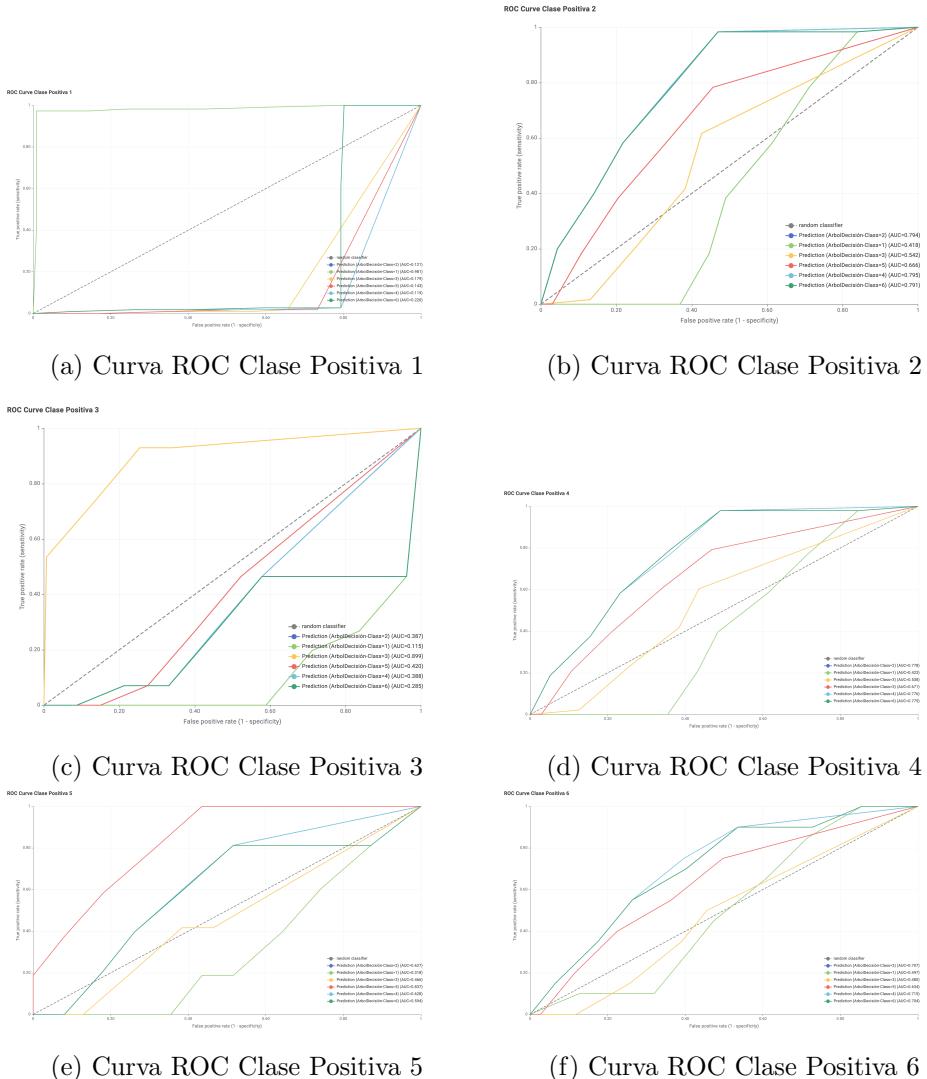
5.3.1. Árbol de decisión

	1 Number (Integer)	2 Number (Integer)	3 Number (Integer)	4 Number (Integer)	5 Number (Integer)	6 Number (Integer)
1	108	0	0	0	0	2
2	2	35	5	28	19	11
3	1	25	66	20	20	7
4	0	0	0	0	0	0
5	0	0	0	0	9	0
6	0	0	0	0	0	0

Figura 35: Matriz de confusión del Árbol de decisión para predicción de enfermedades

- Predice bien la clase 1 con simplemente 3 errores (dos con la clase 2 y uno con la clase 3)
- Predice bien 35 de los 60 ejemplos de la clase 2 osea la mitad
- Predice bien 66 de los 71 ejemplos de la clase 3

- Las clases 4, 5, y 6 no las predice



5.3.2. ZeroR

RowIndex	1 Number (integer)	2 Number (integer)	3 Number (integer)	4 Number (integer)	5 Number (integer)	6 Number (integer)
1	111	60	71	48	48	20
2	0	0	0	0	0	0
3	0	0	0	0	0	0
5	0	0	0	0	0	0
4	0	0	0	0	0	0
6	0	0	0	0	0	0

Figura 37: Matriz de confusión del ZeroR para predicción de enfermedades

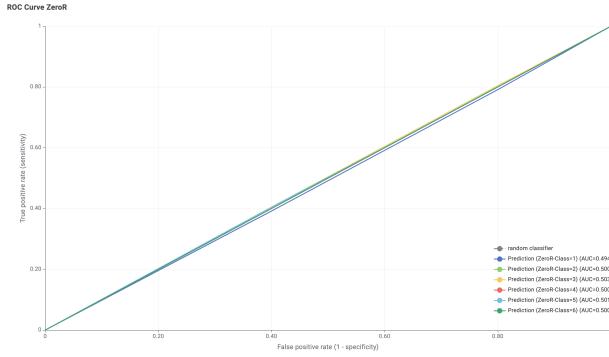


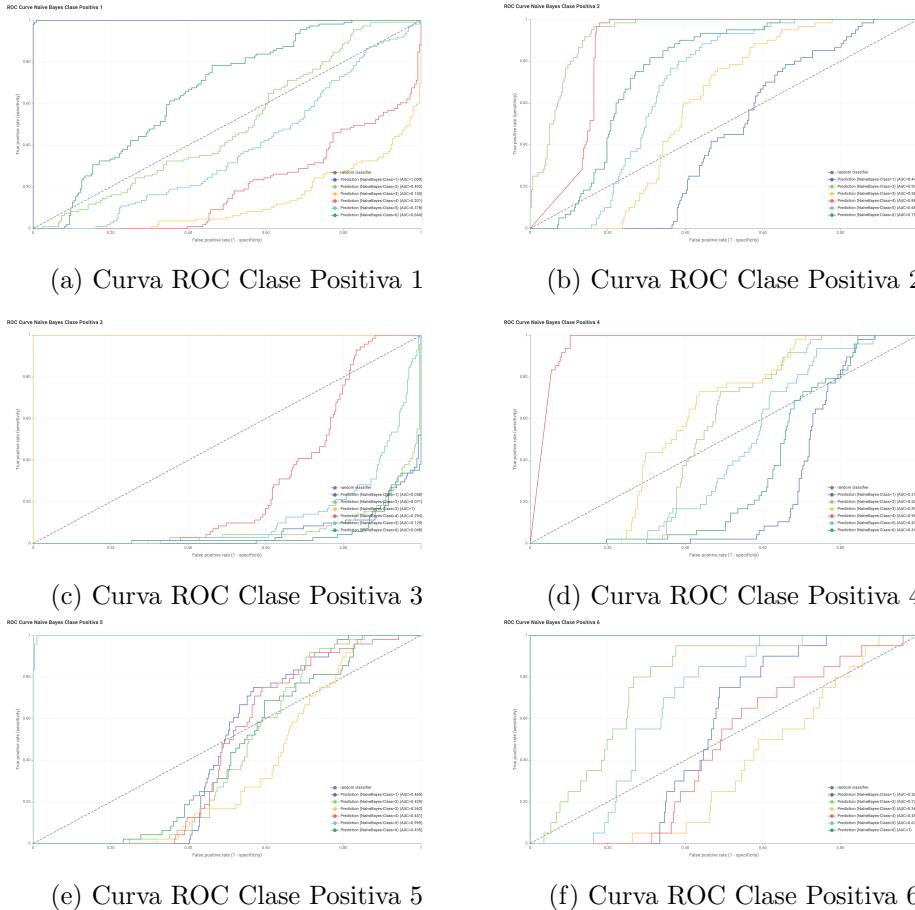
Figura 38: Curva ROC del ZeroR para predicción de enfermedades

- Predice solo la clase mayoritaria como ya sabemos que en este problema es la Clase 1
- No voy a poner todas las curvas ROC de este algoritmo ya que son prácticamente iguales todas

5.3.3. Naïve Bayes

R... ↑	1 Number (integer)	2 Number (integer)	3 Number (integer)	4 Number (integer)	5 Number (integer)	6 Number (integer)
1	107	0	0	0	0	0
2	0	1	0	0	0	0
3	0	0	70	0	0	0
4	1	57	1	48	0	1
5	3	2	0	0	48	0
6	0	0	0	0	0	19

Figura 39: Matriz de confusión del Naïve Bayes para predicción de enfermedades



Como vemos en la figura 39

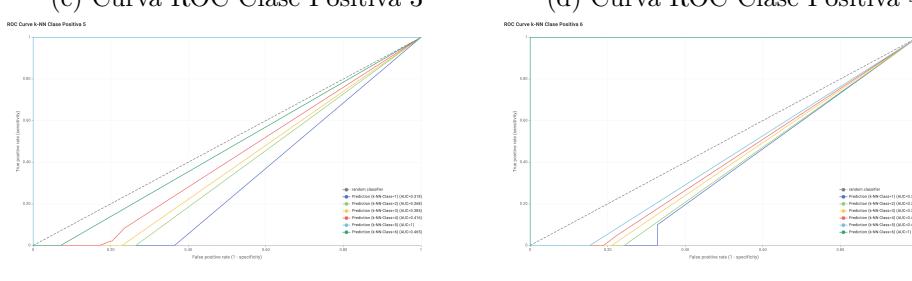
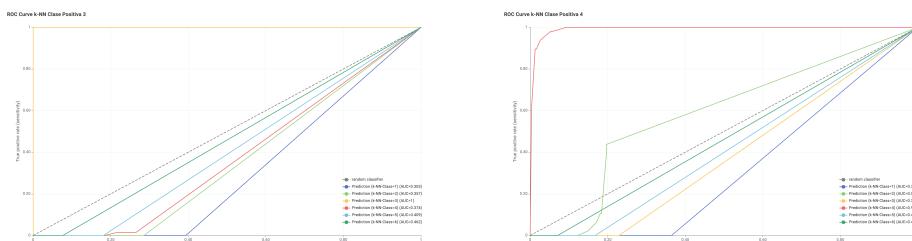
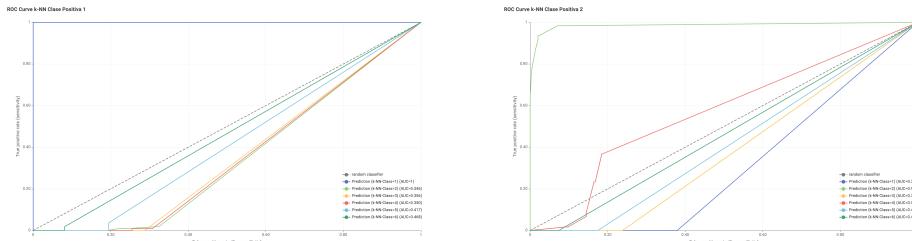
- Predice bien la clase 1 con simplemente 4 errores (1 con la clase 4 y tres con la clase 5)
- Predice mal la clase 2 (confunde con la clase 4 mayoritariamente y dos ejemplos con la 5)
- Predice casi perfectamente la clase 3, simplemente falla con un ejemplo que predice la clase 4
- Para la clase 4 y 5 la predicción es perfecta
- Para la clase 6 solo se equivoca en uno que confunde con la clase 4

En cuanto a las Curvas ROC podemos ver como unos pliegues característicos que no hemos visto hasta ahora. Esto puede deberse a que cuando el modelo produce probabilidades muy similares o agrupadas para muchas ejemplos, hay menos puntos de corte distintos al generar la curva ROC. Esto hace que la curva se vea 'escalonada' en lugar de ser suave.

5.3.4. k-NN

R...	1	2	3	4	5	6
	Number (integer)					
1	110	0	0	0	0	0
2	1	52	0	3	0	0
3	0	0	71	0	0	0
4	0	8	0	45	0	0
5	0	0	0	0	48	0
6	0	0	0	0	0	20

Figura 41: Matriz de confusión del k-NN para predicción de enfermedades

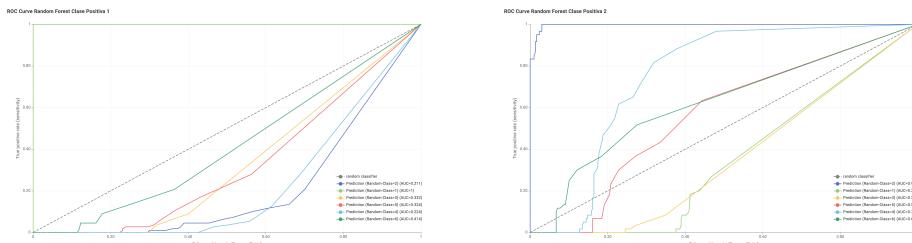


■ Predice casi perfectamente

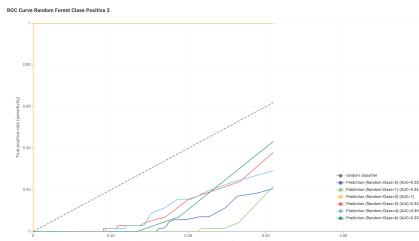
5.3.5. Random Forest

R...	↑	1 Number (integer)	↙	2 Number (integer)	↙	3 Number (integer)	↙	4 Number (integer)	↙	5 Number (integer)	↙	6 Number (integer)
1		111		0		0		0		0		1
2	0			56		0		4		0		0
3	0			0		71		0		0		0
4	0			4		0		44		0		0
5	0			0		0		0		48		0
6	0			0		0		0		0		19

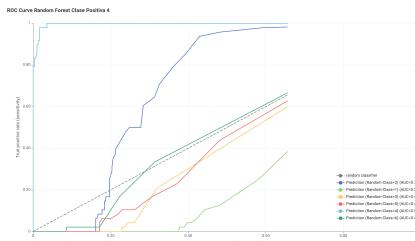
Figura 43: Matriz de confusión del Random para predicción de enfermedades



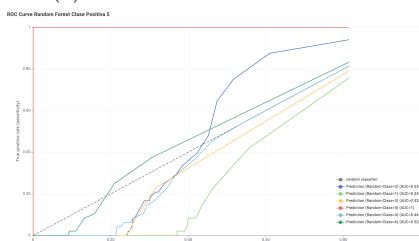
(a) Curva ROC Clase Positiva 1



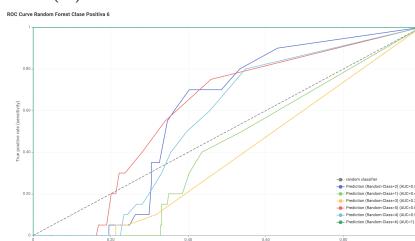
(c) Curva ROC Clase Positiva 3



(d) Curva ROC Clase Positiva 4



(e) Curva ROC Clase Positiva 5



(f) Curva ROC Clase Positiva 6

- Predice perfectamente la clase 1, 3 y 5
 - Falla en la predicción de la 2 4 y 6 pero con un par de ejemplos nada más

5.3.6. Medidas conjuntas

Ahora pasamos a ver las métricas conjuntas para los diferentes algoritmos

R...	Algoritmo	Recall Número (double)	Precision Número (double)	Sensitivity Número (double)	Specificity Número (double)	F-measure Número (double)			
1	Árbol de decisión	0.982	0.973	0.982	0.988	0.977			
1,du...	ZeroR	0.31	1	0.31	0.99	0.473			
1,du...	NaïveBayes	1	0.964	1	0.984	0.982			
1,du...	k-NN	1	0.991	1	0.996	0.995			
1,du...	RandomForest	0.991	1	0.991	1	0.996			
2,du...	Árbol de decisión	0.35	0.583	0.35	0.433	0.436			
2,du...	ZeroR	0	0	0	0.932	0			
2,du...	NaïveBayes	1	0.017	1	0.835	0.033			
2,du...	k-NN	0.929	0.867	0.929	0.974	0.897			
2,du...	RandomForest	0.933	0.933	0.933	0.987	0.933			
3	Árbol de decisión	0.475	0.93	0.475	0.977	0.629			
3,du...	ZeroR	0	0	0	0.802	0			
3,du...	NaïveBayes	1	0.986	1	0.997	0.993			
3,du...	k-NN	1	1	1	1	1			
3,du...	RandomForest	1	1	1	1	1			
4	Árbol de decisión	0	0	0	0	0	0.866	0	
4,du...	ZeroR	0	0	0	0	0	0.866	0	
4,du...	NaïveBayes	0.444	0	0	0.444	0	0.615	0	
4,du...	k-NN	0.849	0.938	0.849	0.849	0.93	0.891	0	
4,du...	RandomForest	0.917	0.917	0.917	0.917	0.927	0.917	0	
5	Árbol de decisión	0	0	0	0	0	0.889	0.916	
5,du...	ZeroR	0	0	0	0	0	0.866	0	
5,du...	NaïveBayes	0.906	1	1	0.906	1	0.945	0	
5,du...	k-NN	1	1	1	1	1	1	0	
5,du...	RandomForest	1	1	1	1	1	1	0	
6	Árbol de decisión	0	0	0	0	0	0.944	0	
6,du...	ZeroR	0	0	0	0	0	0.944	0	
6,du...	NaïveBayes	1	0.95	1	0.997	0.997	0.974	0	
6,du...	k-NN	1	1	1	1	1	1	0	
6,du...	RandomForest	1	0.95	1	0.99	0.997	0.974	0	

(a) Medidas de los diferentes algoritmos para predicción de enfermedades

(b) Medidas 2 de los diferentes algoritmos para predicción de enfermedades

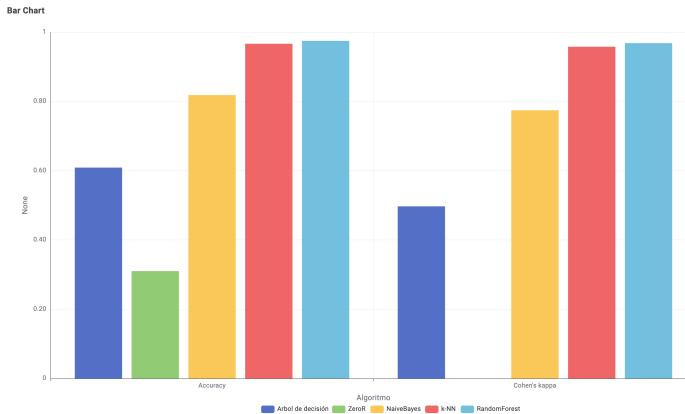


Figura 46: Accuracy y Kappa de los diferentes algoritmos para predicción de enfermedades

Como vemos en la figura 46, en este problema ya encontramos diferencias de eficiencia en los algoritmos donde ahora tenemos a **Random Forest** y a un **k-NN** (con mucha mejoría respecto a los otros problemas) como mejores opciones, un Naïve Bayes mediocre como siempre. El árbol de decisión obtiene resultados muy malos y ZeroR ya sabemos que es simplemente como referencia.

A continuación analizo el por qué pueden haber sido estos cambios en el rendimiento de los algoritmos.

El caso del árbol de decisión puede deberse a que como hay clases que tienen menos ejemplos que las demás, el árbol puede no llegar a dividirse lo suficiente para captarlas o por otro lado, como estamos usando GINI en este

caso que hace que las divisiones busquen maximizar la pureza puede que ciertas clases (4,5,6) no den una ganancia significativa por lo que el árbol no genera los nodos específicos para estas clases y por consiguiente la precisión del algoritmo se vea muy mermada como es el caso.

Por otro lado tenemos un k-NN subiendose al top y teniendo un rendimiento casi perfecto lo cual puede deberse a que cuando se tiene una cantidad adecuada de ejemplos para cada clase, el algoritmo puede ser muy robusto y preciso y también que una de sus características que es NO asumir ninguna distribución de los datos (como Naïve Bayes supone que son independientes las variables) hace que se pueda adaptar mejor en problemas en los que las clases tienen distribuciones complejas, lo cual es común en muchos problemas multiclas.

6. Interpretación de los resultados

En términos generales, Random Forest ha sido el algoritmo con los mejores resultados en todos los problemas.

k-NN ha resultado ser una opción sólida para problemas multiclas debido a su flexibilidad y capacidad de adaptación a diferentes distribuciones de clase.

El Árbol de Decisión es ideal cuando se busca interpretabilidad y rápidez en el análisis, aunque su capacidad de generalización es limitada en problemas complejos o multiclas como hemos podido comprobar

Naive Bayes ha tenido un rendimiento normal en todos los problemas.

ZeroR simplemente ha sido utilizado para tomar una referencia

7. Bibliografía

Referencias

[1] *Videos KNIME*

<https://ccia.ugr.es/~casillas/knime.html>

[2] *Tema 4. Parte 1. Clasificación*

https://pradogrado2425.ugr.es/pluginfile.php/311120/mod_resource/content/12/T4-prediccion_parte1_clasificacion.pdf

[3] *Tema 5. Preprocesado de datos*

https://pradogrado2425.ugr.es/pluginfile.php/311122/mod_resource/content/5/T5-preprocesado_de_datos.pdf