Práctica 1: Análisis Predictivo Mediante Clasificación

José Antonio Álvarez Ocete

Índice

1. Introducción

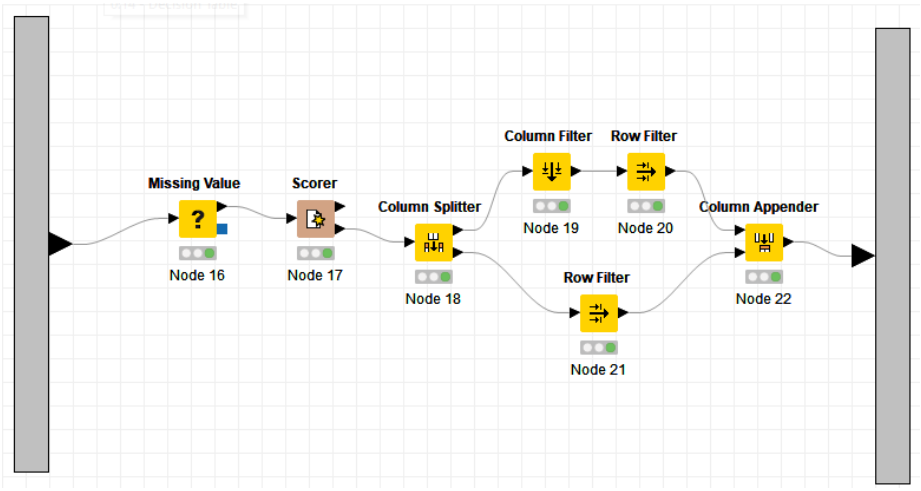
En esta práctica abordamos un problema de predicción del estado de bombas de agua en Tanzania de cara a la competición **Pump it Up: Data Mining the Water Table**. Para esta tarea utilizaremos datos obtenidos de Taarifa y el Ministerio de Agua de Tanzania. En particular, nuestro objetivo será predecir que bombas de agua requeriran reparación.

2. Resultados obtenidos

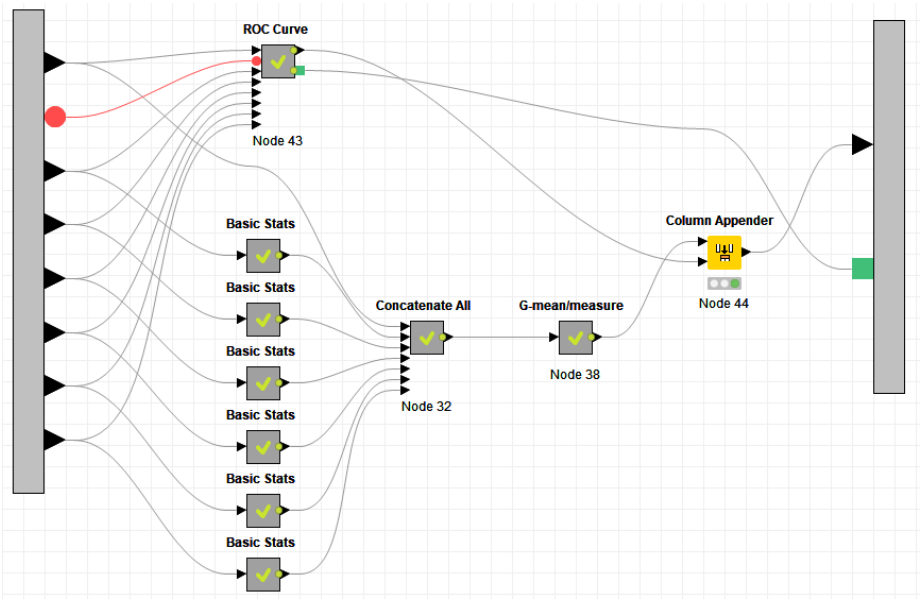
Se han utilizado los siguientes algoritmos, principalmente buscando diversidad entre los tipos de algoritmos y dos algoritmos relativamente similares (los dos primeros) para poder realizar una posterior comparación en detalle:

- Random Forest
- Gradient Boosted Trees
- Naive Bayes
- Multilayer Perceptron (MLP)
- K-NN
- Decision Table.

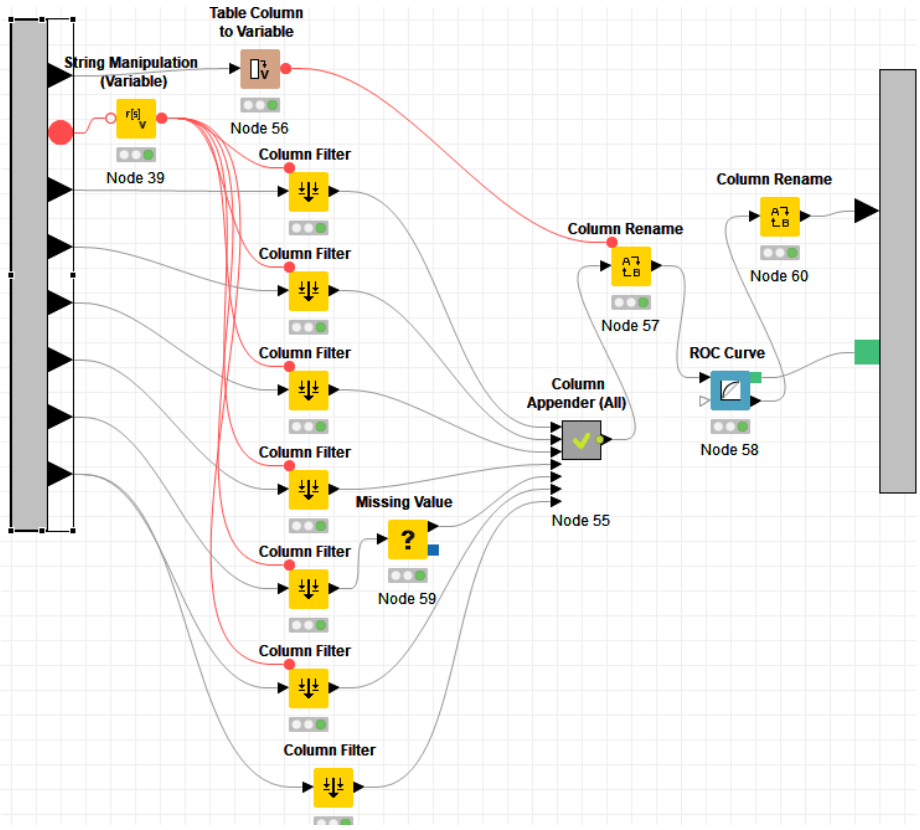
Tomaremos como clase positiva `non functional`, ya que buscamos predecir aquellas bombas que están completamente rotas. Se ha realizado un estudio de validación cruzada sobre el conjunto de test y el de entrenamiento, como podrá apreciarse en las próximas capturas de KNIME. Esto se debe a que en la sección de preprocesado discutimos el sobreajuste de los distintos algoritmos. Todos los datos presentados en esta sección hacen referencia a la validación cruzada sobre el test, ejecutada con igual semilla para todos los algoritmos, `123456`. El flujo de trabajo utilizado para calcular las estadísticas de cada algoritmo ha sido el siguiente.



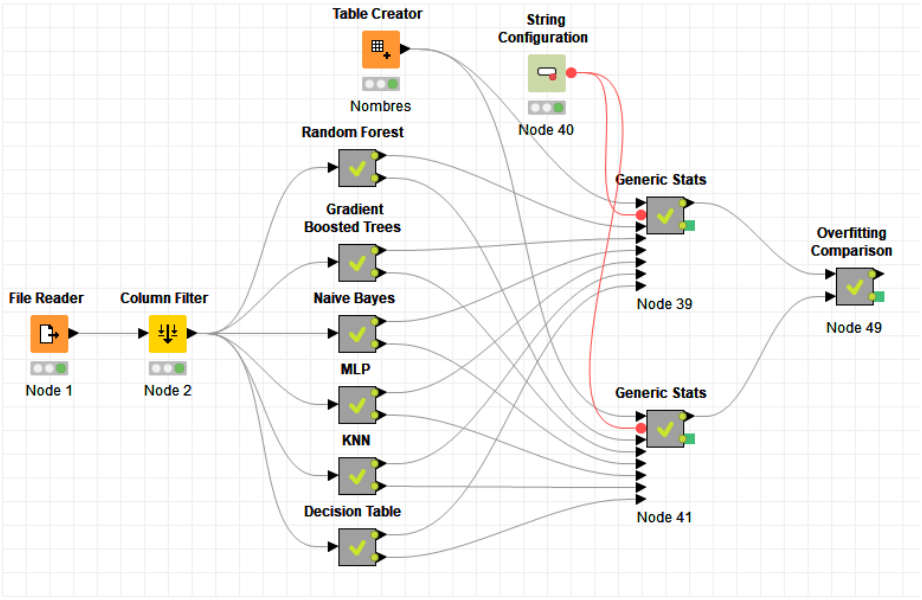
Este es el contenido del metanodo **BasicStats**, donde el **Column Splitter** nos permite recoger el `accuracy` de la fila `Overall`. Ejecutamos este nodo para los distintos algoritmos y agrupamos los datos:



Tras reunir las estadísticas de todos los algoritmos en una única tabla calculamos tres de medidas de precisión adicionales: `G-mean`, `G-measure` y `AUC`. Para esta última calculamos la `ROC Curve` utilizando el metanodo con el mismo nombre:

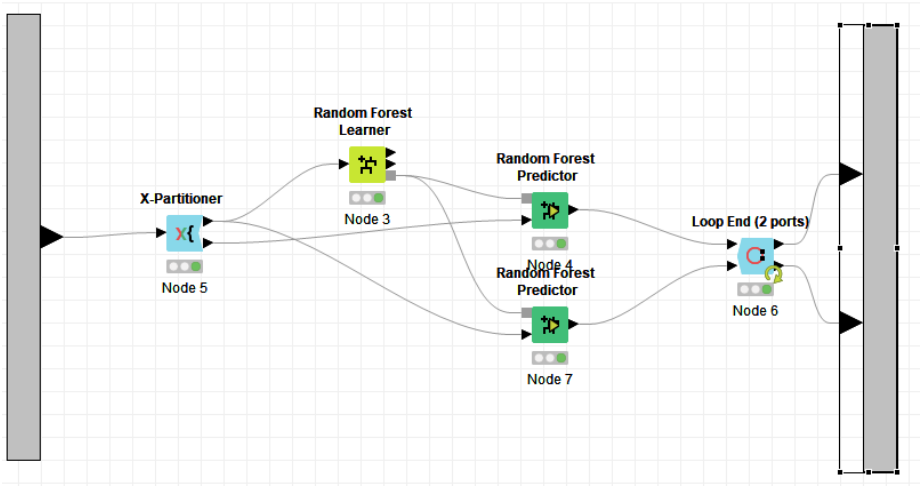


Finalmente mostramos el flujo completo, posteriormente utilizado en el estudio del sobreajuste.



2.1 Random Forest

El flujo del algoritmo utilizado en KNIME es el siguiente:



Inicialmente utilicé **Domain Calculator** para calcular los dominios de atributos no calculados por defecto. Esto se debe, de hecho, a que el número de valores que pueden tomar algunas variables categoricas es desorbitadamente alto. Se realizará un estudio en profundidad sobre este detalle en la sección de preprocesado. Presentamos a continuación la matriz de confusión obtenida, asi como otras

medidas de precisión.

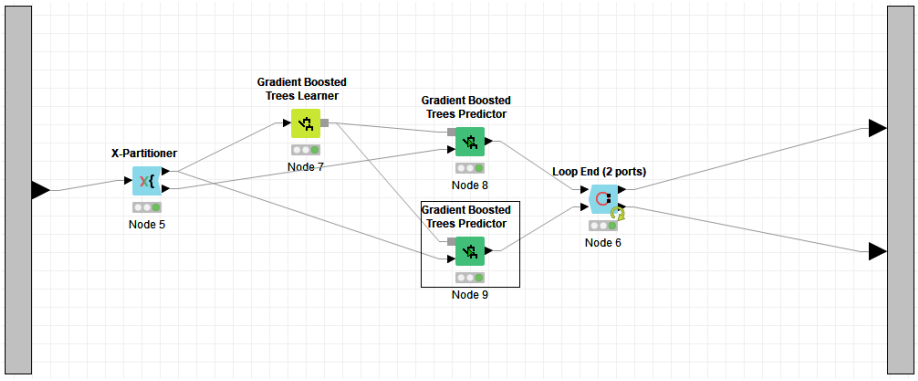
Row ID	functional	non functional	functional needs repair
functional	29444	2292	523
non functional	5403	17142	279
functional nees repair	2518	581	1218

Row ID	TP	FP	TN	FN	Recall	PPV	TPR	TNR	F1-score
functional	29444	7921	19220	2815	0.9127	0.7880	0.9127	0.7081	0.8458
non functional	17142	2873	33703	5682	0.7511	0.8565	0.7511	0.9215	0.8003
functional needs repair	1218	802	54281	3099	0.2821	0.6030	0.2821	0.9854	0.3844

	Accuracy	Cohen's kappa
Overall	0.8048	0.6292

2.2 Gradient Boosted Trees

El flujo del algoritmo utilizado en KNIME es el siguiente:



Presentamos a continuación la matriz de confusión obtenida, asi como otras medidas de precisión.

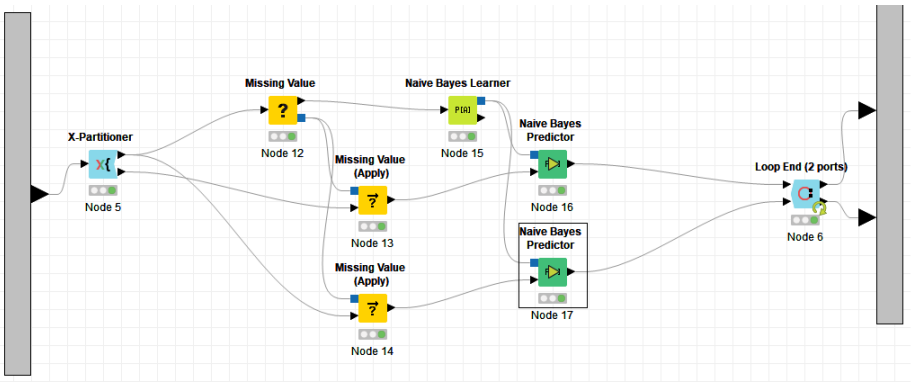
Row ID	functional	non functional	functional needs repair
functional	29428	2569	262
non functional	6954	15704	166
functional nees repair	2856	715	746

Row ID	TP	FP	TN	FN	Recall	PPV	TPR	TNR	F1-score
functional	29428	9810	17331	2831	0.9122	0.7500	0.9122	0.6385	0.8232
non functional	15704	3284	33292	7120	0.6880	0.8270	0.6880	0.9102	0.7512
functional needs repair	746	428	54655	3571	0.1728	0.6354	0.1728	0.9922	0.2717

	Accuracy	Cohen's kappa
Overall	0.7724	0.5597

2.3 Naive Bayes

El flujo del algoritmo utilizado en KNIME es el siguiente:



Del mismo modo que ocurría con los dos algoritmos anteriores, algunas variables categóricas no son utilizadas debido a que toman demasiados valores distintos. Cabe destacar que para que el algoritmo ejecute correctamente utilizando el atributo `region` hemos de aumentar el número máximo de valores por categoría a 21. Presentamos a continuación la matriz de confusión obtenida, asi como otras medidas de precisión.

Row ID	functional	non functional	functional needs repair
functional	19291	6385	6385
non functional	5141	14703	2980
functional nees repair	1240	934	2143

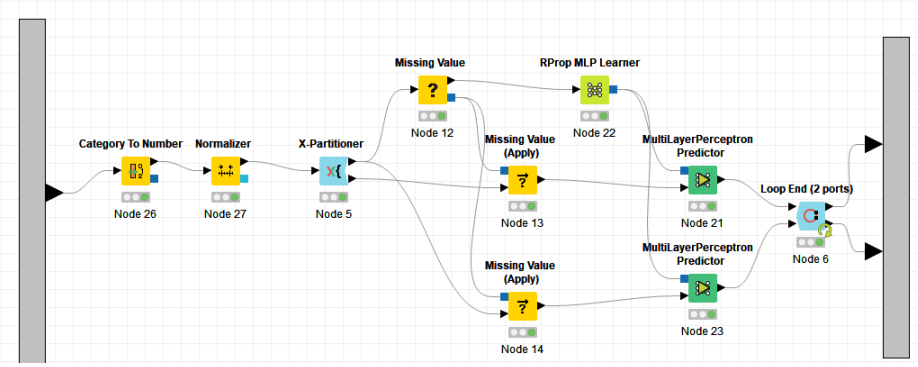
Row ID	TP	FP	TN	FN	Recall	PPV	TPR	TNR	F1-score
functional	19291	6381	20760	12968	0.5980	0.7514	0.5980	0.7649	0.6660
non functional	14703	7319	29257	8121	0.6442	0.6677	0.6442	0.7999	0.6557

Row ID	TP	FP	TN	FN	Recall	PPV	TPR	TNR	F1-score
functional needs repair	2143	9563	45520	2174	0.4964	0.1831	0.4964	0.8264	0.2675

	Accuracy	Cohen's kappa
Overall	0.6084	0.3564

2.4 Multilayer Perceptron

El flujo del algoritmo utilizado en KNIME es el siguiente:



Puesto que este algoritmo no acepta valores perdidos ni categoricos, hemos de transformar todos los valores a numéricos antes y normalizar en el intervalo `[0,1]` antes de ejecutarlo. Obtendríamos más información si pudiesemos ejecutar **Missing Values** sobre los datos antes de aplicar el **Category to Number**, ya que trataríamos los strings de forma distinta, obteniendo más información.

Si hiciesemos eso, tendría que ser unicamente sobre el conjunto de entrenamiento, y al normalizar y aplicar dicha normalización sobre los valores de test algunos escapan del rango `[0,1]`. Es por ello que numerizamos y normalizamos antes de realizar el particionamiento.

Presentamos a continuación la matriz de confusión obtenida, asi como otras medidas de precisión.

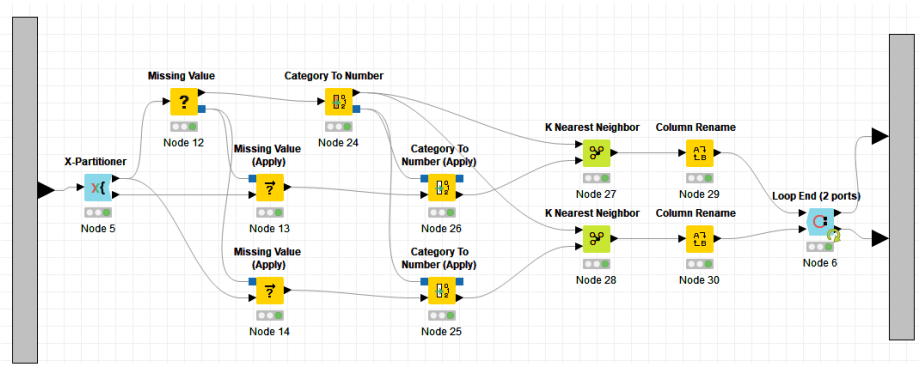
Row ID	functional	non functional	functional needs repair
functional	27624	4614	21
non functional	9693	13127	4
functional nees repair	3425	861	31

Row ID	TP	FP	TN	FN	Recall	PPV	TPR	TNR	F1-score
functional	27624	13118	14023	4635	0.8563	0.6780	0.8563	0.5167	0.7568
non functional	13127	5475	31101	9697	0.5751	0.7057	0.5751	0.8503	0.6338
functional needs repair	31	25	55058	4286	0.0072	0.5536	0.0072	0.9995	0.0142

	Accuracy	Cohen's kappa
Overall	0.6866	0.3819

2.5 K-NN

El flujo del algoritmo utilizado en KNIME es el siguiente:



Al igual que MLP, KNN no acepta valores perdidos ni categóricos. Tratamos debidamente estos casos. Presentamos a continuación la matriz de confusión obtenida, asi como otras medidas de precisión.

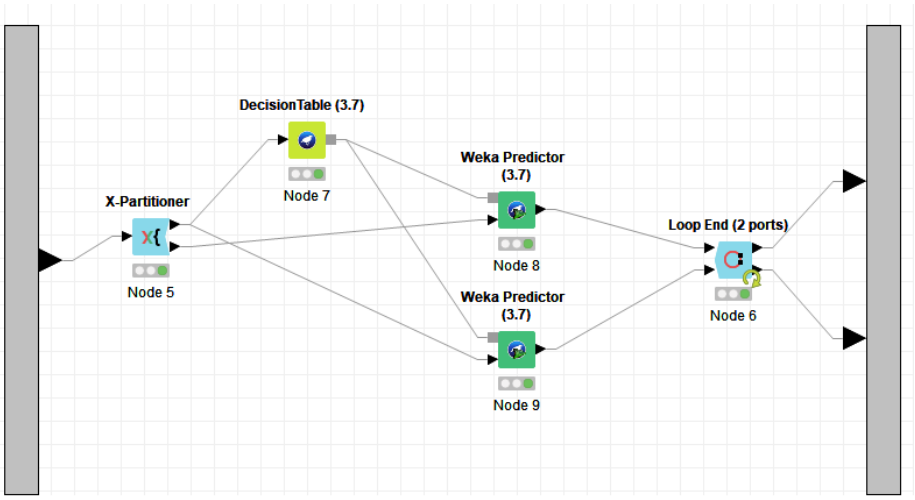
Row ID	functional	non functional	functional needs repair
functional	26001	5316	941
non functional	7517	14931	376
functional nees repair	2450	826	1041

Row ID	TP	FP	TN	FN	Recall	PPV	TPR	TNR	F1-score
functional	26001	9967	17174	6258	0.8060	0.7229	0.8060	0.6327	0.7622
non functional	14931	6142	30434	7893	0.6542	0.7085	0.6542	0.8321	0.68023
functional needs repair	1041	1317	53766	3276	0.2411	0.4415	0.2411	0.9760	0.3119

	Accuracy	Cohen's kappa
Overall	0.7066	0.4485

Decision Table

El flujo del algoritmo utilizado en KNIME es el siguiente:



Presentamos a continuación la matriz de confusión obtenida, asi como otras medidas de precisión.

Row ID	functional	non functional	functional needs repair
functional	29113	2710	436
non functional	8322	14208	294
functional nees repair	2794	716	807

Row ID	TP	FP	TN	FN	Recall	PPV	TPR	TNR	F1-score
functional	29113	11116	16025	3146	0.9025	0.7237	0.9025	0.5904	0.8033
non functional	14208	3426	33150	8616	0.6225	0.8057	0.6225	0.9063	0.7023
functional needs repair	807	730	54353	3510	0.1869	0.5250	0.1869	0.9867	0.2757

	Accuracy	Cohen's kappa
Overall	0.7429	0.5020

3. Análisis de resultados

Añadimos a continuación una tabla conjunta mostrando unicamente la fila `non functional` de la tabla de resultados de cada algoritmo.

Algoritmo	TP	FP	TN	FN	PPV	TPR	TNR
Random Forest	17142	2873	33703	5682	0.8565	0.7510	0.9215
Gradient Boosted Trees	15704	3284	33292	7120	0.8270	0.6880	0.9102
Naive Bayens	14703	7319	29257	8121	0.6677	0.6442	0.7999
MLP	13127	5475	31101	9697	0.7057	0.5751	0.8503
KNN	14931	6142	30434	7893	0.7085	0.6542	0.8321
Decision Table	14208	3426	33150	8616	0.8057	0.6225	0.9063

Algoritmo	F1-score	G-mean	G-measure	Accuracy	AUC
Random Forest	0.8003	0.8319	0.8020	0.8048	0.9195
Gradient Boosted Trees	0.7512	0.7914	0.7543	0.7723	0.8886
Naive Bayens	0.6557	0.7178	0.6558	0.6084	0.7920
MLP	0.6338	0.6993	0.6371	0.6866	0.7942
KNN	0.6803	0.7378	0.6808	0.7066	0.8038
Decision Table	0.7024	0.7511	0.7082	0.7429	0.8695

En términos generales, los algoritmos de ordenados de mejor a peor fijándonos unicamente en estas estadísticas es el siguiente:

1. Random Forest
2. Gradient Boosted Trees
3. Decision Table
4. MLP
5. K-NN
6. Naive Bayes

Estos resultados son, en líneas generales, los esperados. Ante un dataset con tantas variables categóricas se esperaría que los algoritmos de árboles y reglas sean mejores que aquellos centrados en atributos de tipo númerico como son K-NN y MLP. Resulta curioso como Random Forest es consistente mejor en todas y cada unas de las medidas calculadas. En el caso del algoritmo Naive Bayes, podríamos suponer que estos resultados se deben a dependencias entre las variables. Realizaremos un estudio al respecto en la siguiente sección.

Sin embargo, no todos los datos se alinean por completo. Por ejemplo, MLP tiene mayor `F1-score` y `accuracy` que Naive Bayes y K-NN, y menor `G-measue` y `G-mean`. Si observamos la matriz de confusión de MLP vemos como practicamente nunca precide que el valor de la clase es `functional needs repair`. De la misma

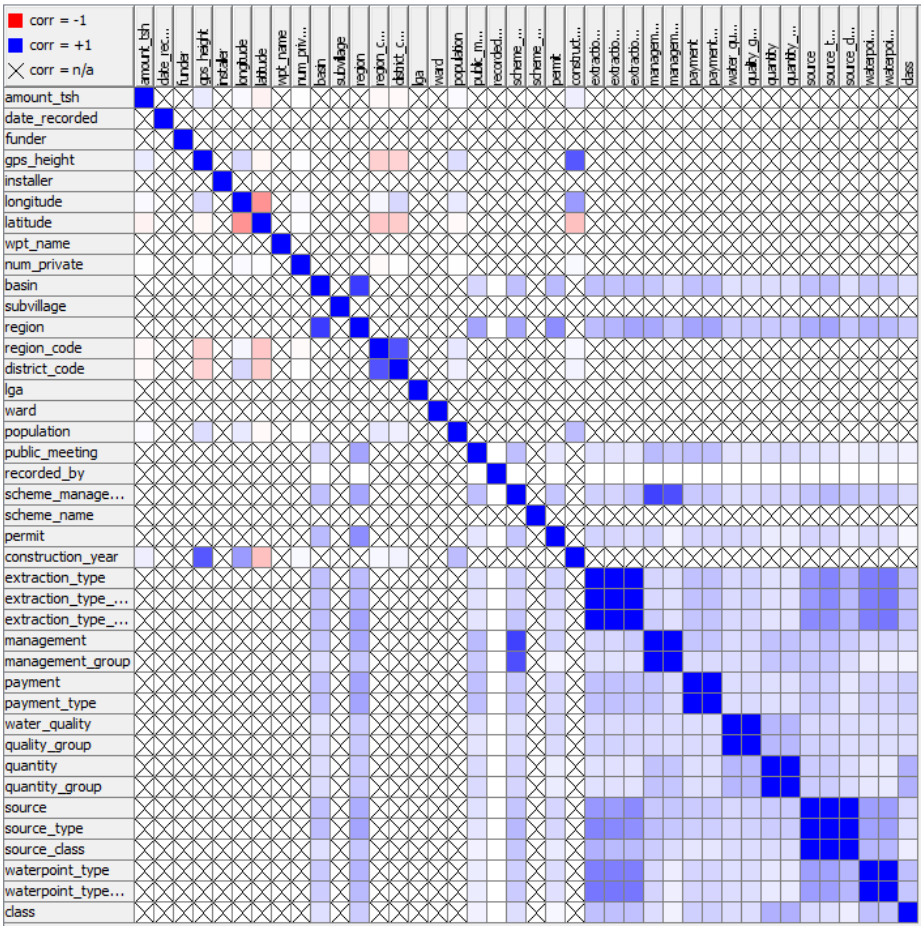
forma, en la tabla de comparativa global se observa como MLP predice más negativos que el resto de algoritmos. Debido a que predecimos mejor la clase negativa que la positiva obtenemos estas estadísticas en este algoritmo.

En relación a lo comentado sobre MLP, podemos observar un claro desbalanceo poblacional comparando los valores TP+FN frente a TN+FP. Observaremos este hecho claramente en la próxima sección.

En cuanto al algorimto KNN, es posible que las transformaciones de las variables categóricas no ayuden precisamente a la diferenciación en base a distintas en la que se basa se algoritmo KNN. En la última sección trabajaremos este concepto.

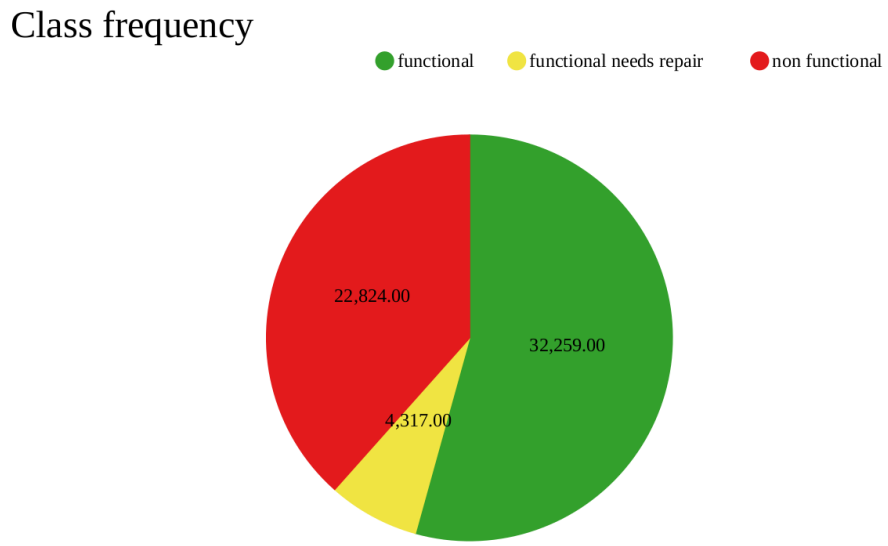
4. Procesado de datos

En primer lugar, tras el estudio del dominio realizado a partir del **Domain Calculator** de la sección 2.1 he decidido eliminar las siguientes variables: `date_recorded`, `funder`, `installer`, `wpt_name`, `subvillage`, `lga`, `ward` y `scheme_name`. Además, realizamos un estudio de la dependencia de las variables y descubrir que de hecho muchas de ellas dependen totalmente de otras. Para ello utilizaremos el nodo de KNIME **Linear Correlation**.



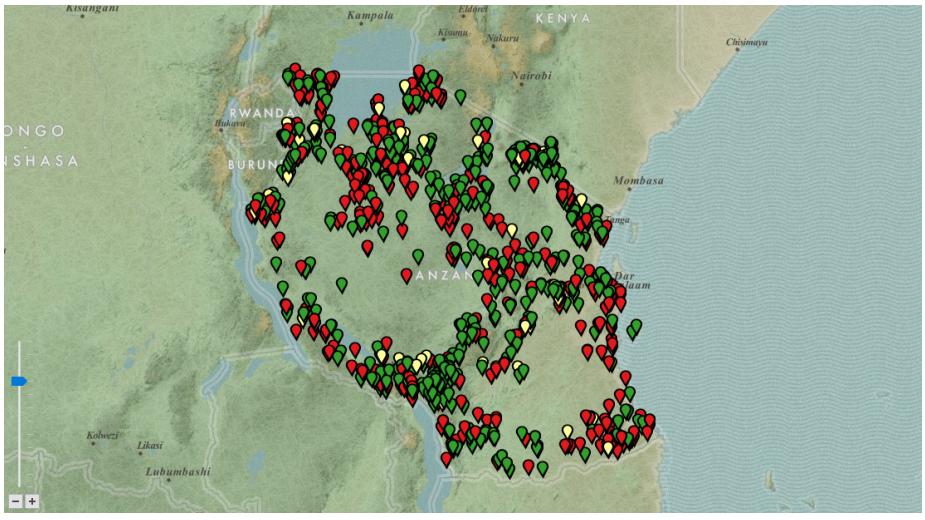
En base a estos resultados eliminamos además las variables `extraction_type_group`, `extraction_type_class`, `management_group`, `payment_type`, `quality_group`, `quantity_group`, `source_type`, `source_class` y `water_type_group` por tener un coeficiente de correlación de 1 o practicamente 1. Adicionalmente, esto verifica parcialmente nuestra hipótesis sobre el algoritmo Naive Bayes y su pobre rendimiento ante la presencia de estas relaciones.

Por otro lado, realizamos un estudio exploratorio mostrando la frecuencia con la que la variable `class` toma cada valor posible para ver si nuestra población está desbalanceada.



Debido a este hecho, utilizaremos la opción `Stratified sampling` en el nodo **X-partitioner** de la validación cruzada de cara a la ejecución final.

Buscamos ahora analizar la distribución geográfica de las bombas de agua con el objetivo de validar los datos verificando que las bombas de agua están, efectivamente, en Tanzania. Utilizamos para ello el paquete **Palladian** y su nodo **Map Viewer**:



Aunque en la imagen unicamente representamos 1000 instancias para poder verlas con claridad, si representamos todas ellas observamos que efectivamente todas se hayan en Tanzania.

Realizando estos preprocesados (esencialmente eliminando variables) ejecutamos nuestros algoritmos de nuevo, obteniendo los siguientes resultados:

Algoritmo	TP	FP	TN	FN	PPV	TPR	TNR
Random Forest	17131	2695	33881	5693	0.8641	0.7506	0.9263
Gradient Boosted Trees	15703	3285	33291	7121	0.8270	0.6880	0.9102
Naive Bayens	16127	10820	25756	6697	0.5985	0.7066	0.7042
MLP	13023	6540	30036	9801	0.6657	0.5706	0.8212
KNN	14877	6244	30332	7947	0.7044	0.6518	0.8293
Decision Table	14139	3432	33144	8685	0.8047	0.6195	0.9062

Algoritmo	F1-score	G-mean	G-measure	Accuracy	AUC
Random Forest	0.8033	0.8068	0.8053	0.8338	0.9208
Gradient Boosted Trees	0.7511	0.7723	0.7543	0.7913	0.8886
Naive Bayens	0.6480	0.5893	0.6503	0.7054	0.7849
MLP	0.6145	0.6680	0.6163	0.6845	0.7756
KNN	0.6771	0.7040	0.6776	0.7352	0.8014
Decision Table	0.7000	0.7416	0.7060	0.7492	0.8665

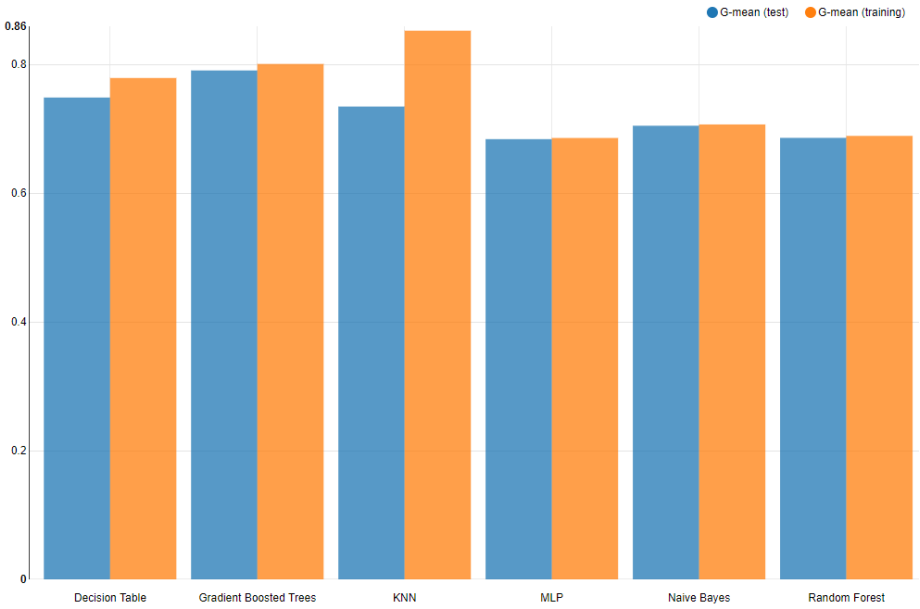
Observamos comparando esta tabla con la anterior como nuestros resultados han empeorado o se han mantenido iguales. A pesar de que la AUC ha mejorado ligeramente en algunos, el resto de medidas descienden en para muchos de ellos.

Fijándonos particularmente en Naive Bayes y nuestra hipótesis sobre su incorrecto funcionamiento debido a las dependencias entre variables, vemos como sus apenas mejoran: a pesar que el número de aciertos en la clase positiva (`TP`) asciende, el de la clase negativa (`TN`) desciende notablemente. Es por ella que la media `G-mean` tambien desciende. Observamos un comportamiento similar en los algoritmos MLP y KNN.

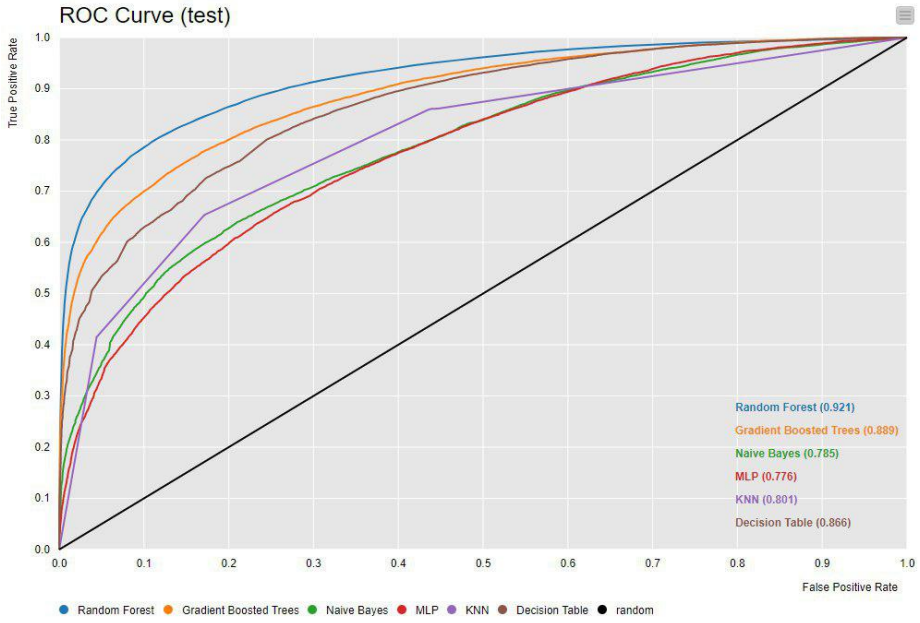
Podemos observar también como los valores `TP`, `FP`, `TN` y `FN` apenas varían para los algoritmos Random Forest, Gradient Boosted Trees y Decision Table. Esto nos indica que estos algoritmos habian detectado e ignorado dichas dependencias. Además, a pesar de que los resultados no mejoran, el tiempo de ejecución lo hace notablemente reduciéndose hasta en un `80%`. Es por ello que mantendremos esta configuración de paramétros para los estudios futuros.

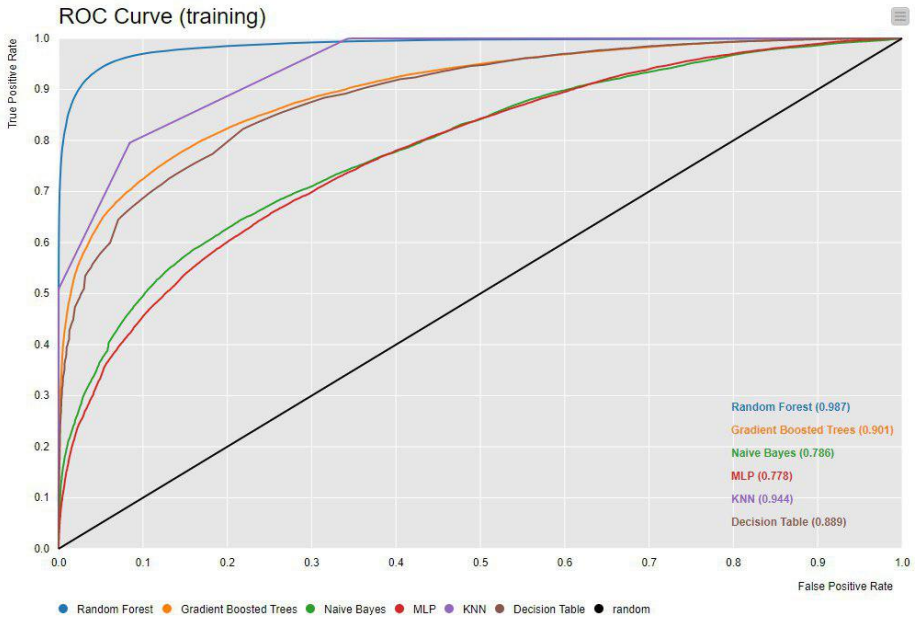
Estudiamos a continuación el sobreajuste de nuestro algoritmos. Para ello utilizamos el workflow completo presentado en la sección 2, junto con el metanodo **Overfitting Comparisson**:

Obtenemos la siguiente comparativa de `G-mean` cuando predecimos test frente a cuando predecimos los datos de entrenamiento:



Presentamos a continuación las diferentes curvas ROC de los algoritmos para ambas predicciones.





Vemos como las curvas ROC apenas mejoran sobre los datos de entrenamiento y presenciamos el sobreajuste en la gráfica anterior unicamente en los algoritmos KNN y Random Forest. En KNN, de nuevo esto debe de ocurrir debido a la pequeña distancia tomada al transformar variables categóricas a numéricas y lo sobreajustados que se crean los clusteres (los puntos tomados, en el caso de KNN) posteriormente. En cuanto a Random Forest este sobreajuste puede deberse a que el número de atributos escogidos no está acotado y obtenemos demasiado información de nuestro conjunto de entrenamiento.

5. Consideraciones finales

Lamentablemente y como es obvio no me ha dado tiempo a mejorar las práctica y trabajar los datos tanto como me gustaría. Debido a un error de KNIME perdí todo el workflow y tuve que rehacerlo. Soy plenamente consciente que la culpa de este hecho es unicamente mía.

Presento a continuación una serie de acciones que me habría gustado tomar respecto a la práctica.

- En primer lugar, completar la sección de configuración de algoritmos. Para ello utilizaría Random Forest ya que es el algoritmo con el que mejor resultados he obtenido. Probaría distinto número de árboles asi como atributos máximos a tener en cuenta. Estudiría también que variables son mas relevantes de cara a filtrar las especialmente poco relevantes y re-ejecutar todos los algoritmos. Finalmente realizaría una comparativa entre

la complejidad de Random Forest y Gradient Boosted Trees pasándolo los árboles generados a reglas.

- En la sección anterior estudiaría también el algoritmo KNN ya que creo que los cambios en la transformación de las variables categoricas a números puede afectar realmente a las distancias que se estudian y, por lo tanto, al rendimiento final del algoritmo. Tomaría también distintos valores de K mayores que 3, aunque no se hasta que punto esto ayudaría.
- En cuanto la variable `date_recorded` me parece intuitivamente el más relevante de todos, a pesar de que debido a su formato no he podido trabajarlo. Intenté utilizar **String to Date** y trabajar en base a este nuevo dato, así como pintar las bombas según se clase de forma temporal, pero no conseguí obtener ninguna gráfica bonita de la que pudiese leer nada relevante.
- Me habría gustado realizar también un estudio exploratorio utilizando diagramas de violín, asi como estudiar la frecuencia de valores perdidos (uno de los grandes problemas de este dataset), e incluso eliminar filas con valores perdidos en determinadas (o todas) las características.