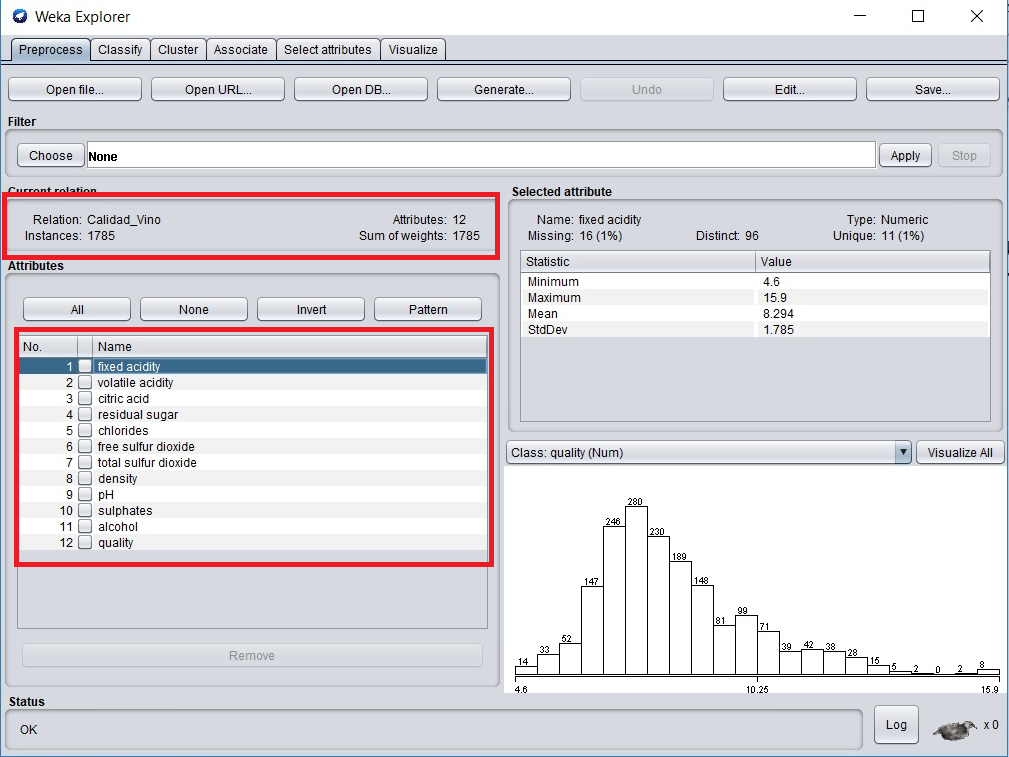
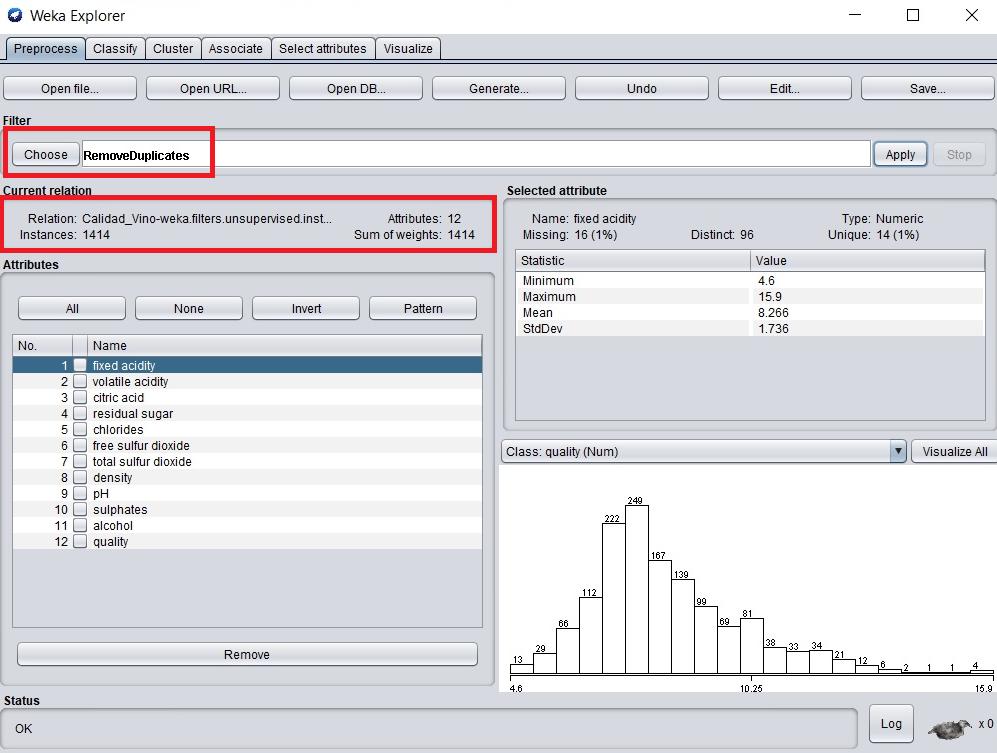
**TÉCNICAS SUPERVISADAS EN WEKA**

Inicialmente se carga el dataset y se miran sus características principales:



1. **Preprocesamiento**
   1. **Manejo de datos duplicados**

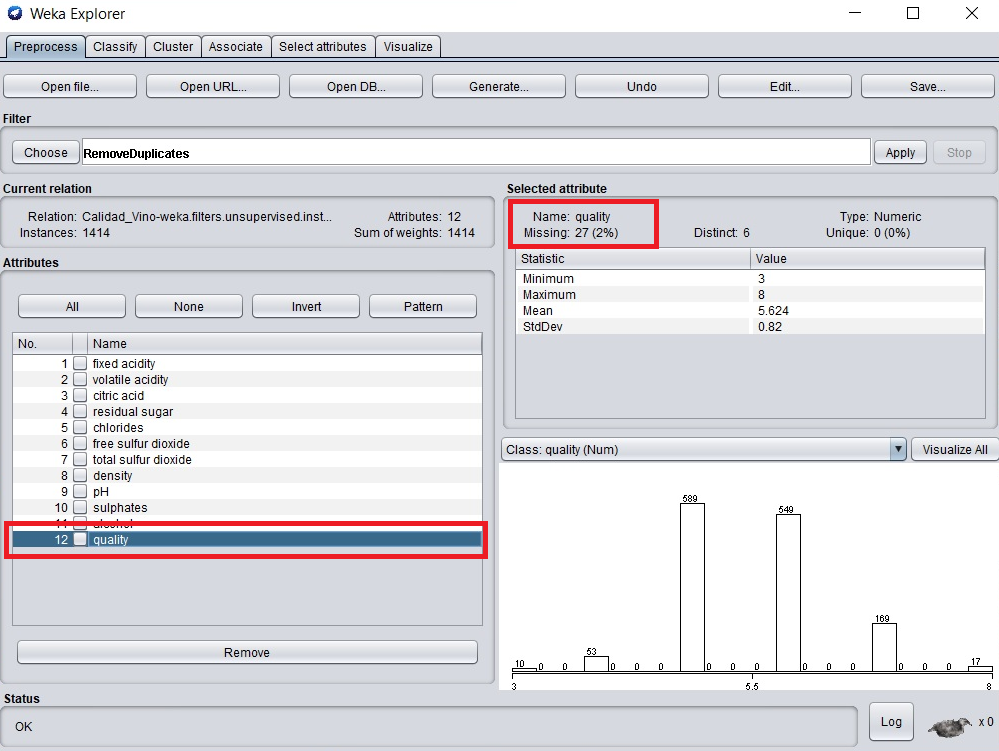
Se aplica el filtro RemoveDuplicates para remover las instancias que están duplicadas:

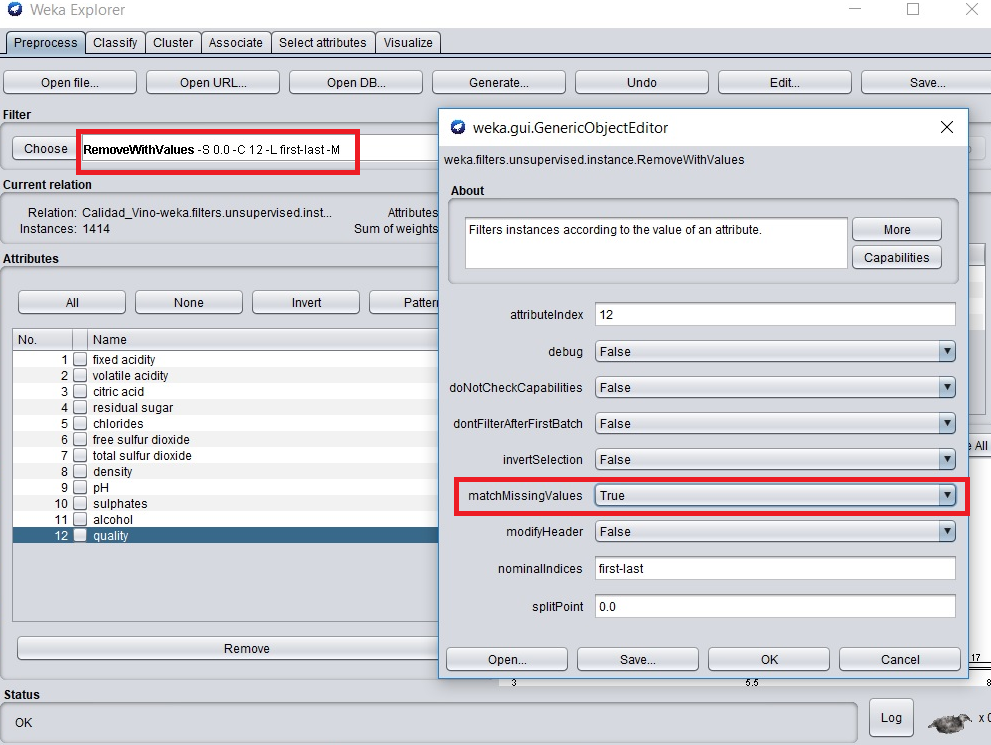


Se observa que el número de instancias se redujo a 1414

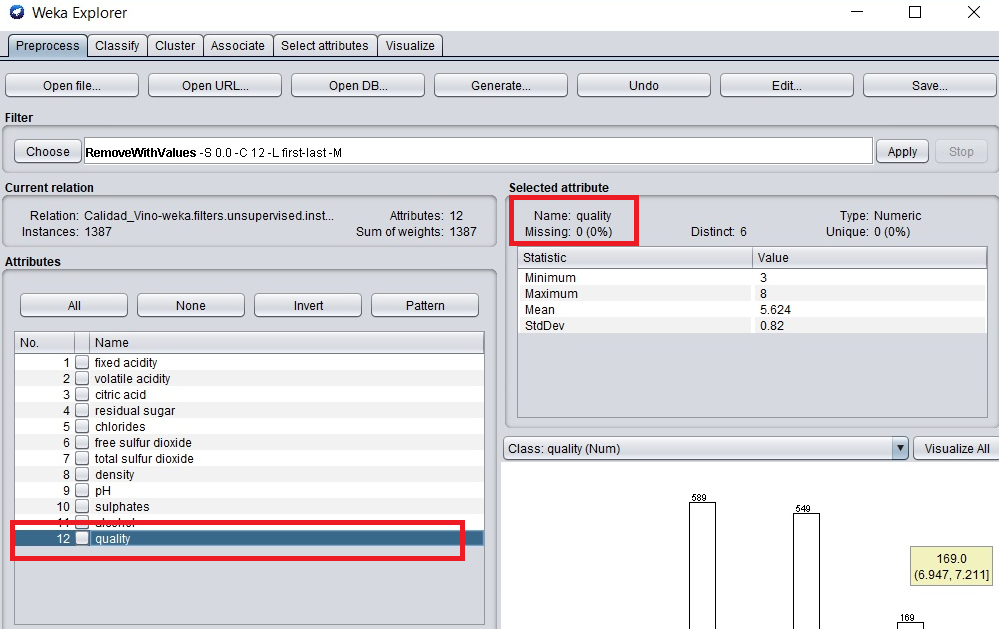
* 1. **Manejo de datos faltantes**

Inicialmente se eliminan todas las instancias que tienen datos faltantes en la variable quality aplicando el filtro RemoveWithValues:

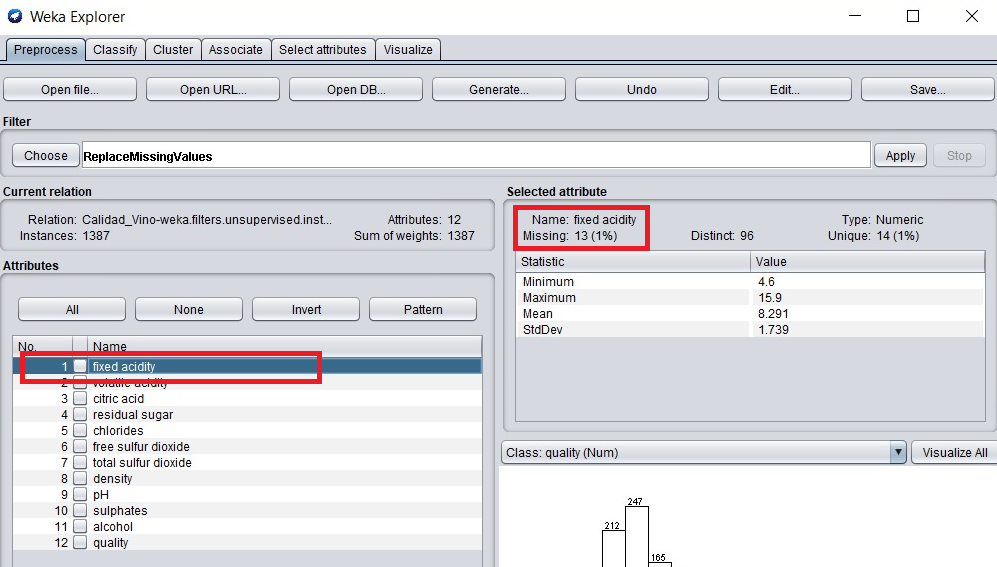




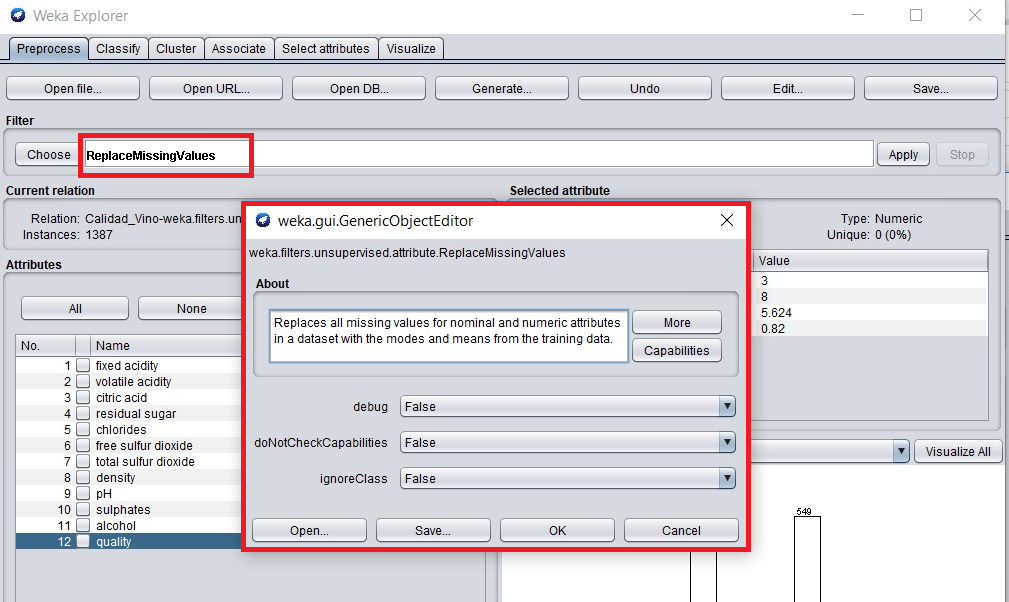
Se observa que esta variable ya no tiene variables faltantes:



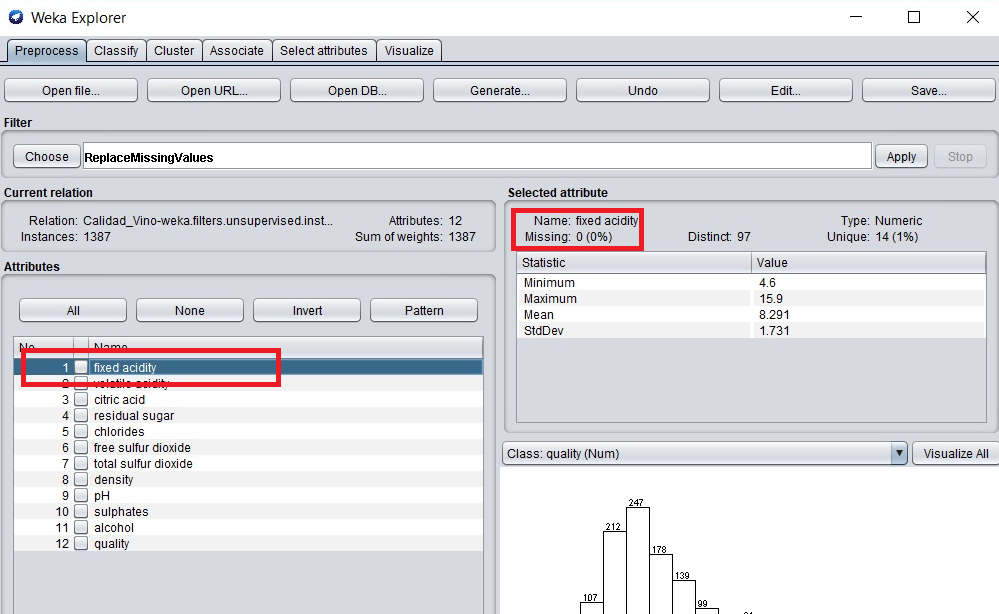
Después, se evidenció que algunas de las otras variables contenían datos faltantes en algunas de sus instancias:



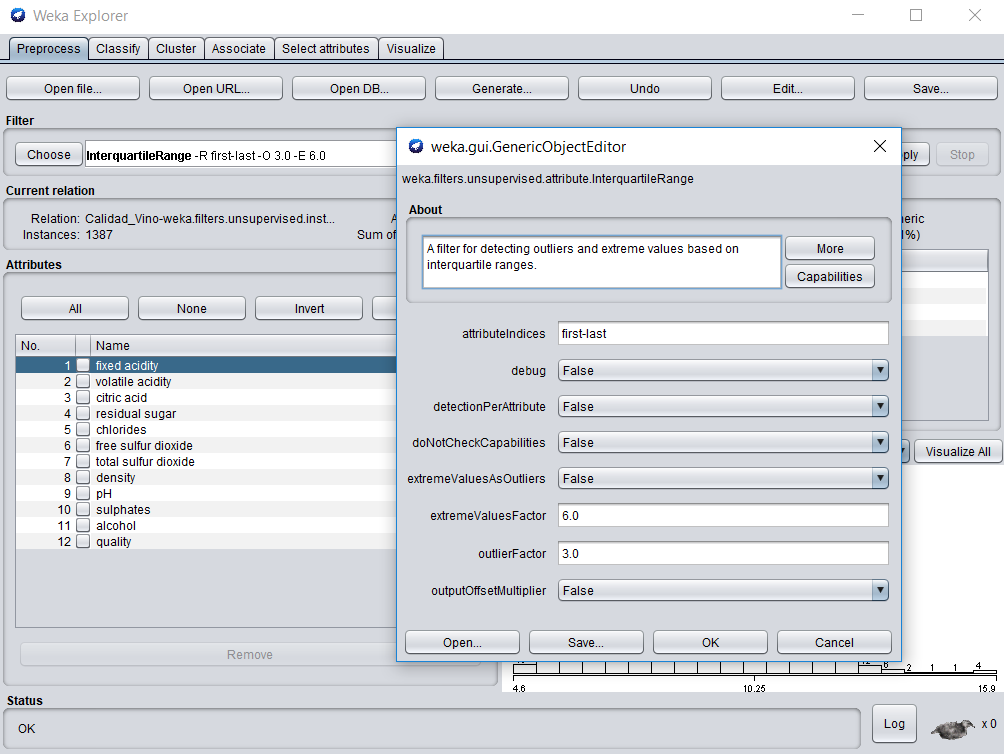
Se decide reemplazar todos los datos faltantes con la media de las variables a la que corresponden. Esto se hizo utilizando el filtro ReplaceMissingValues:



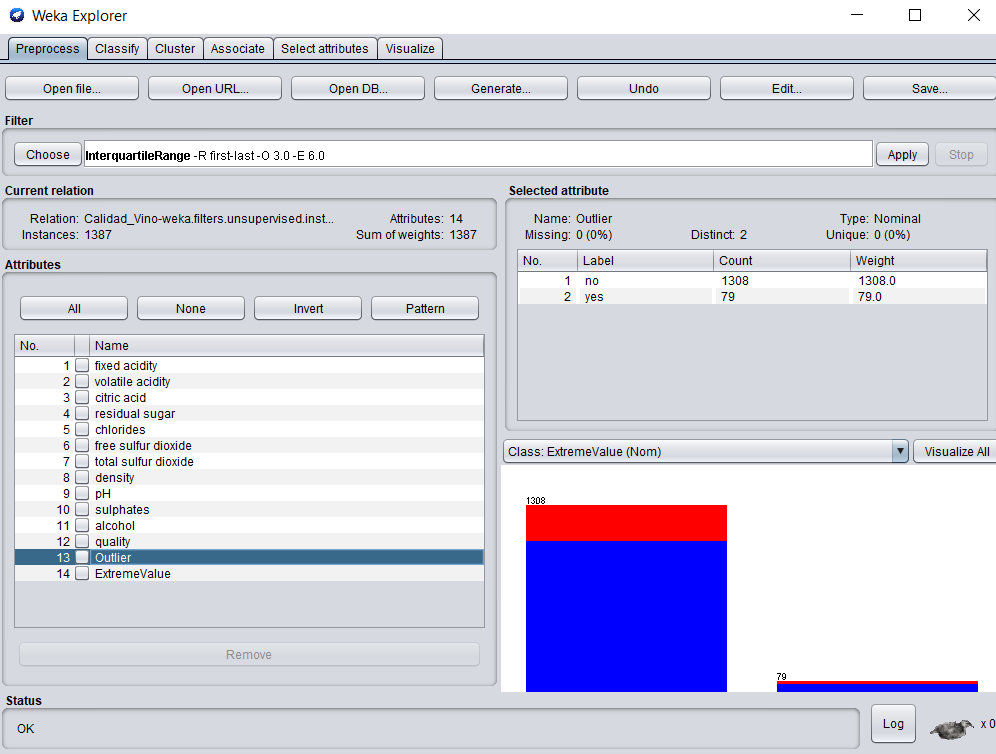
Una vez se aplica el filtro se evidencia que ya no existen valores faltantes en ninguna variable:

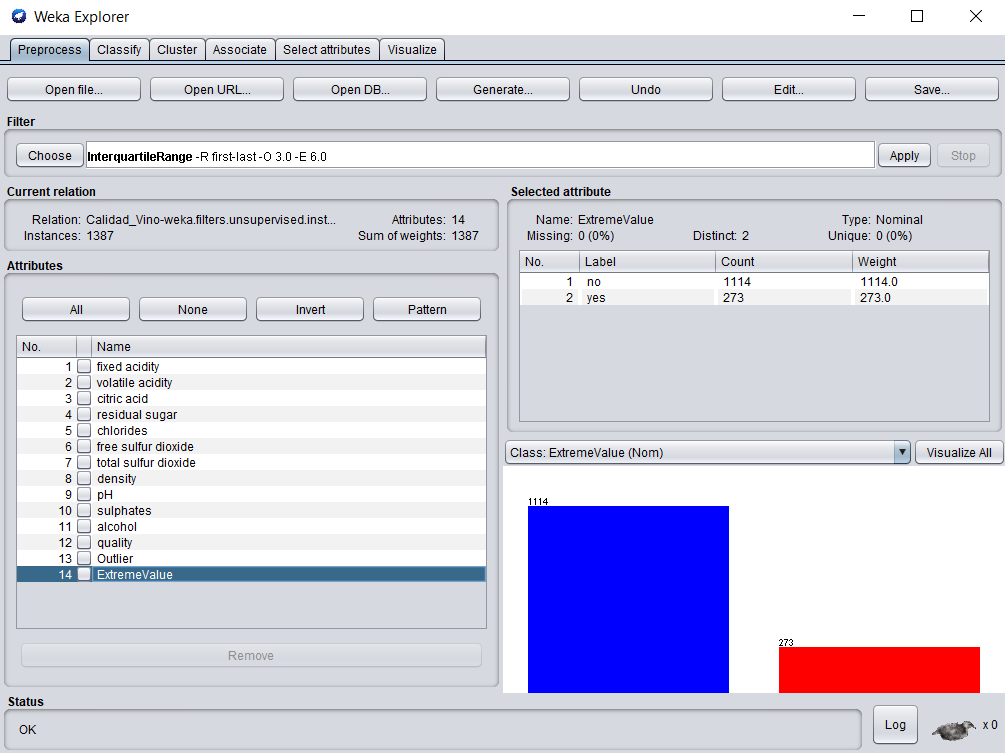


* 1. **Manejo de Outliers y valores extremos:**

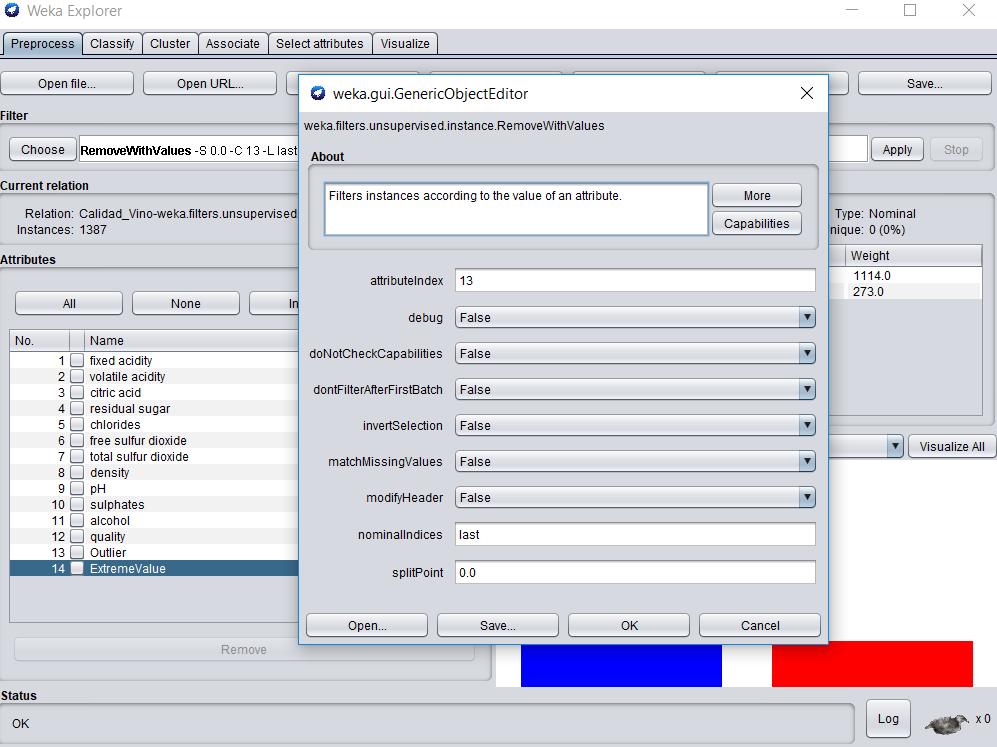


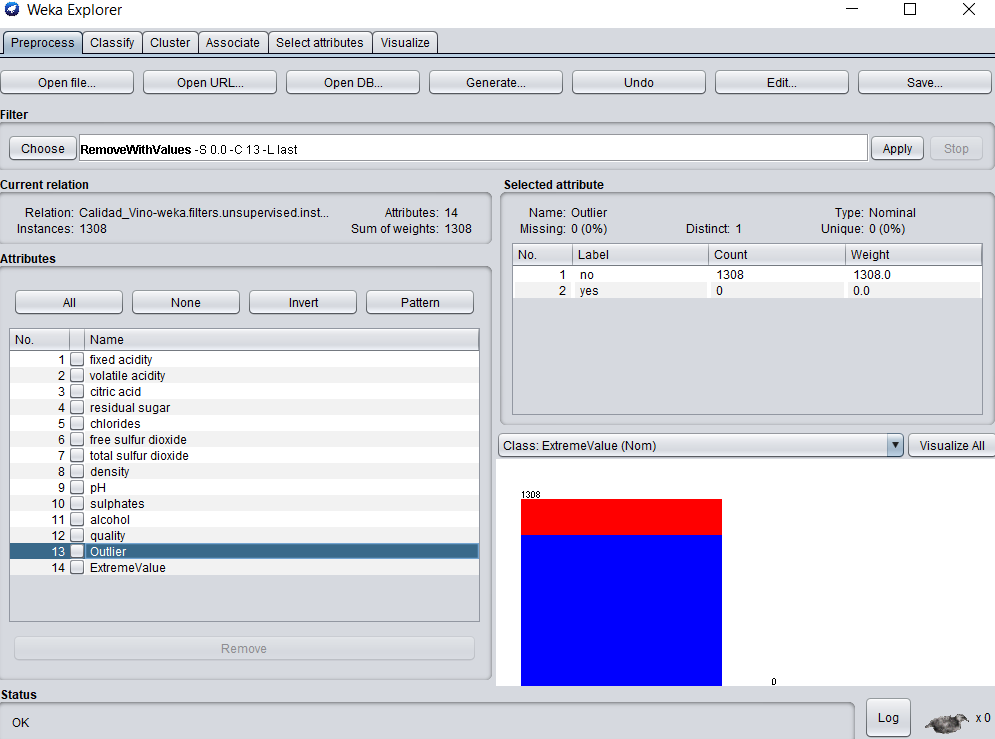
Una vez se aplica el filtro, este genera dos nuevas variables en las cuales se especifican los valores extremos y los outliers:



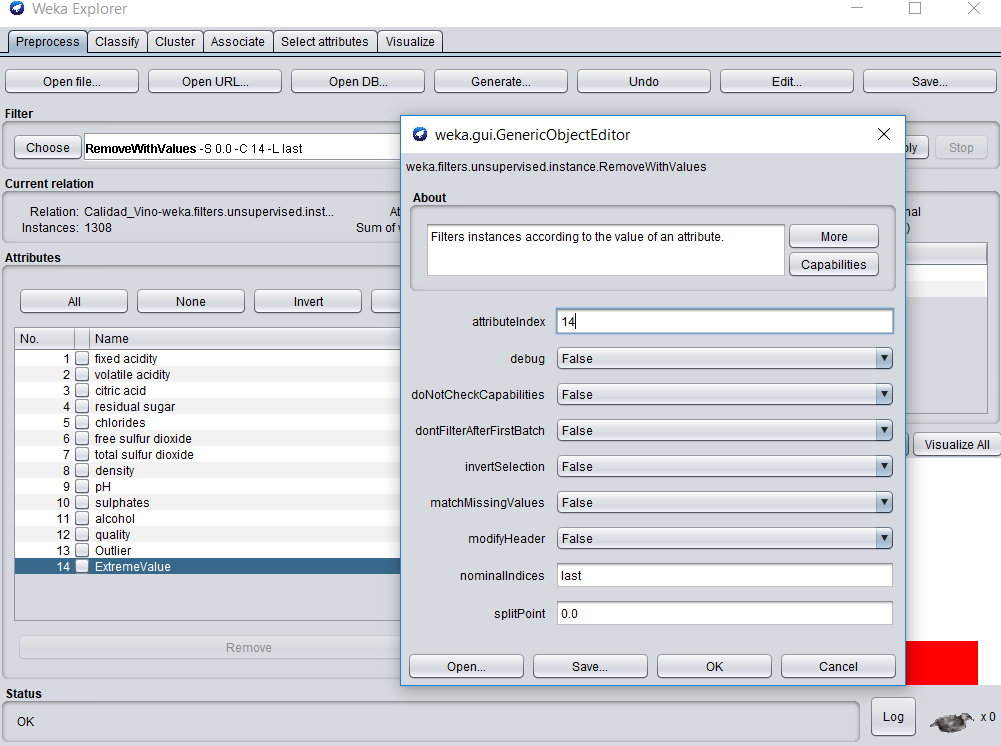


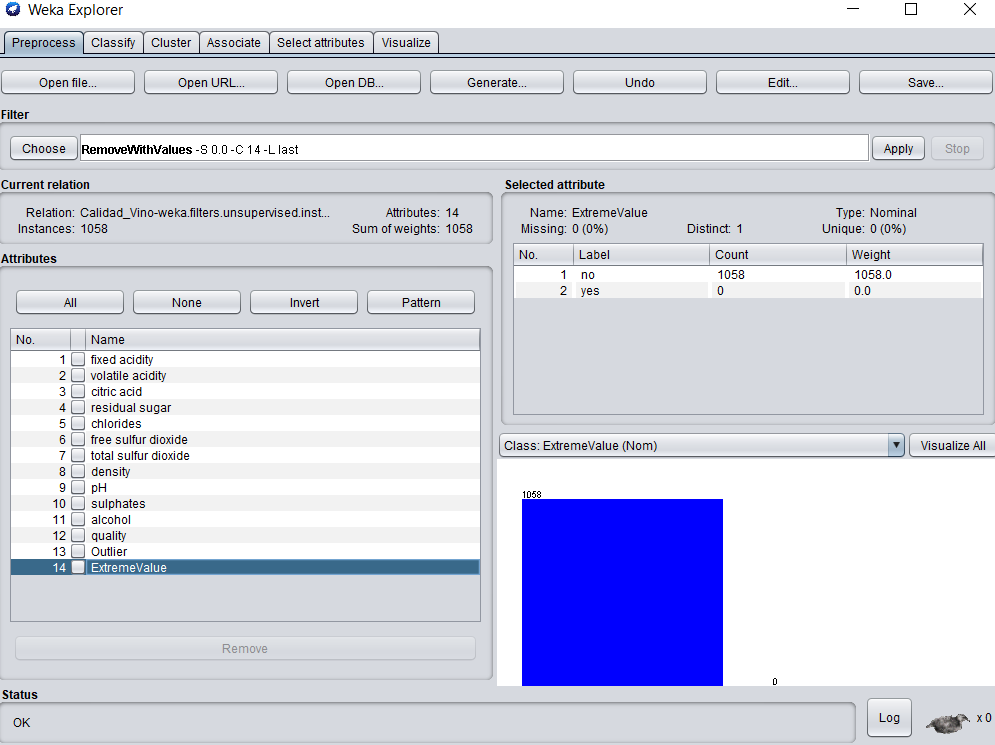
Se procede a eliminar los valores atípicos y extremos, con el filtro RemoveWithValues:





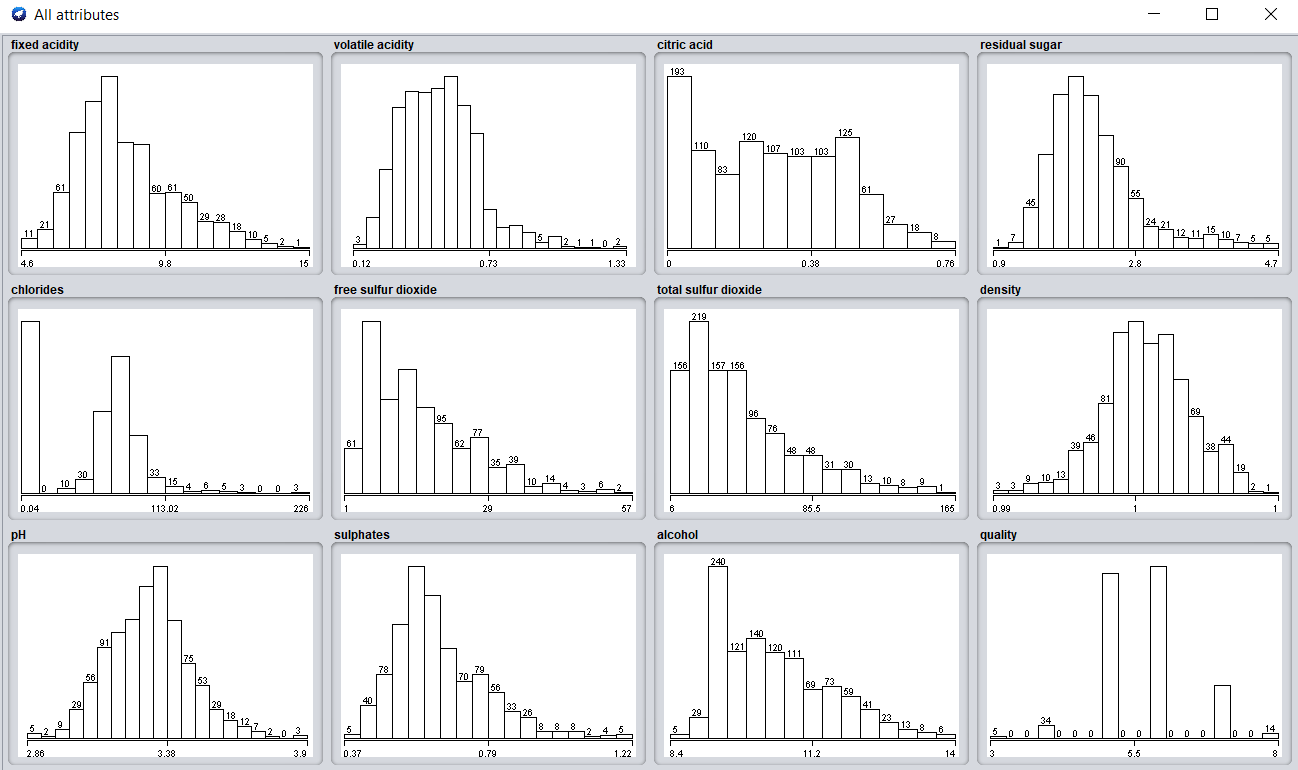
Se realiza un proceso similar con los valores extremos:



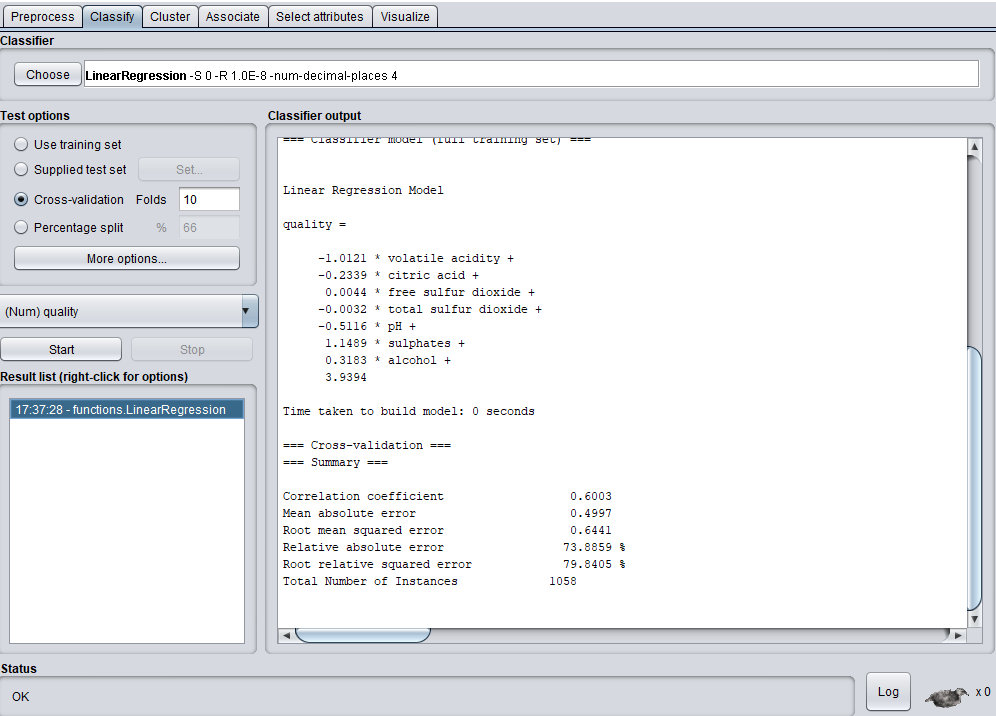


Una vez se eliminan estos valores, se procede a realizar la visualización de los datos:

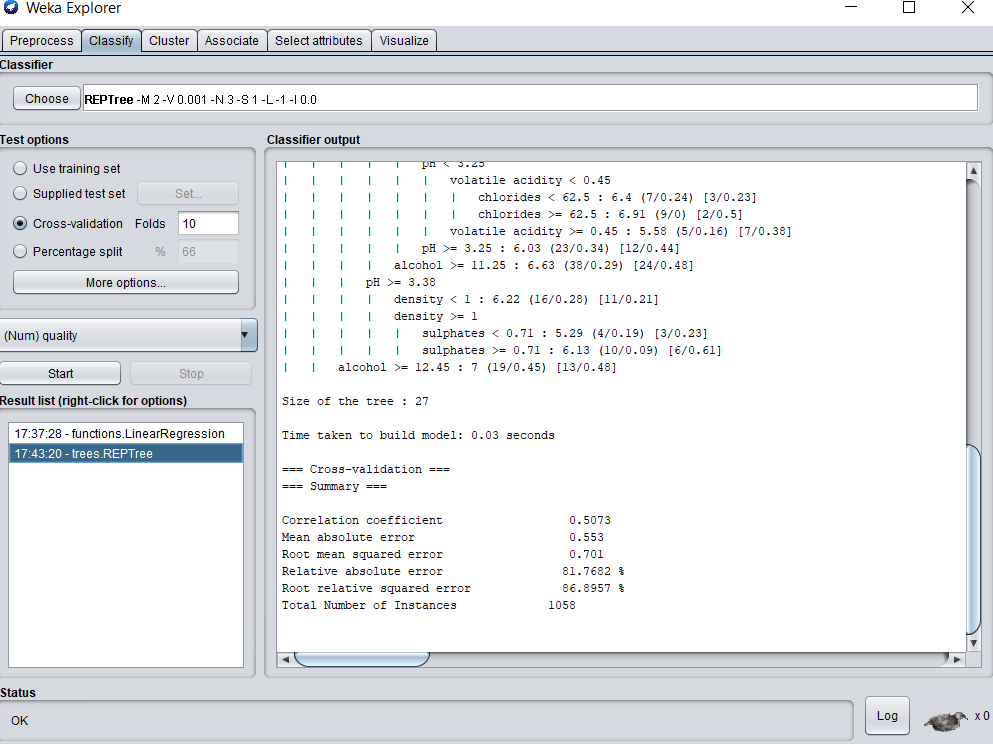




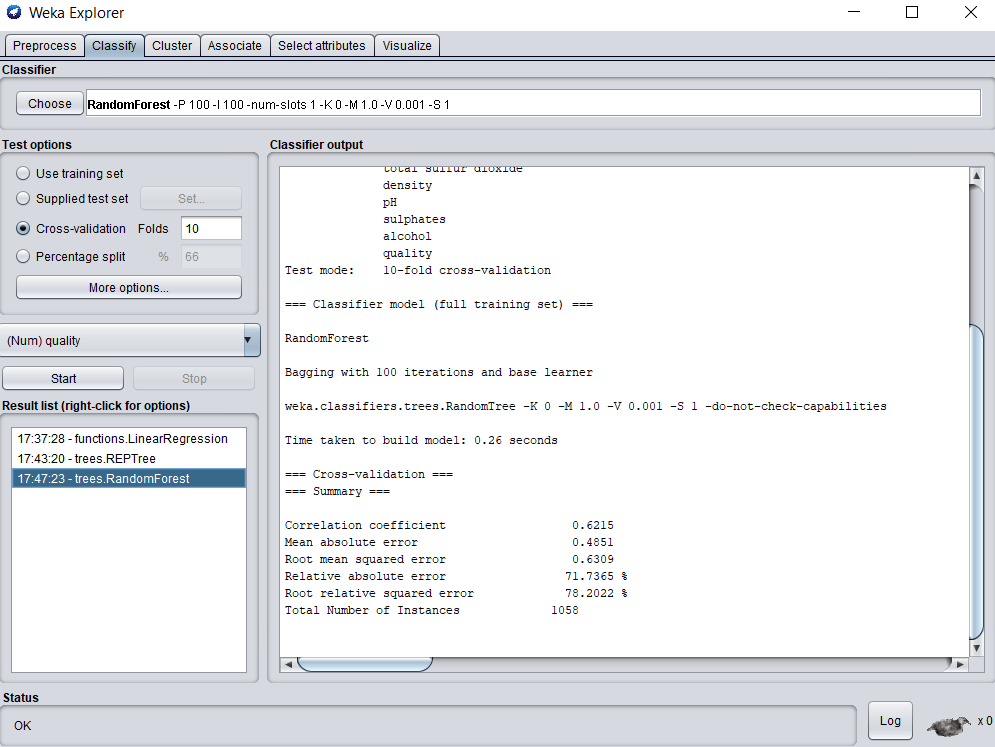
1. **Algoritmos de regresión**
   1. **Regresión Lineal**



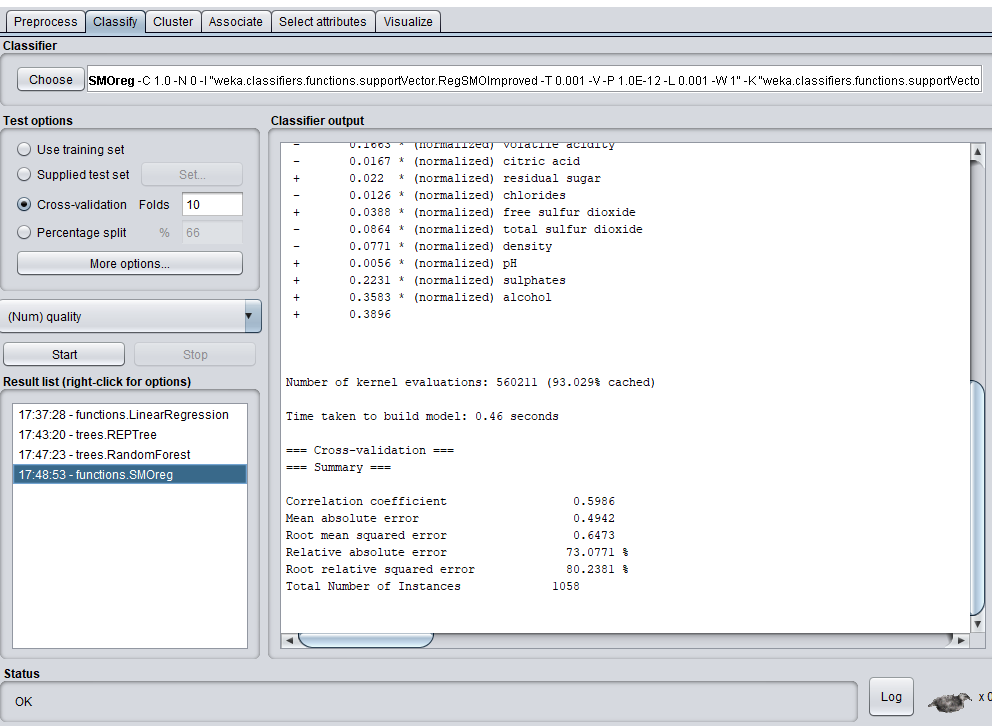
* 1. **Decision Tree**



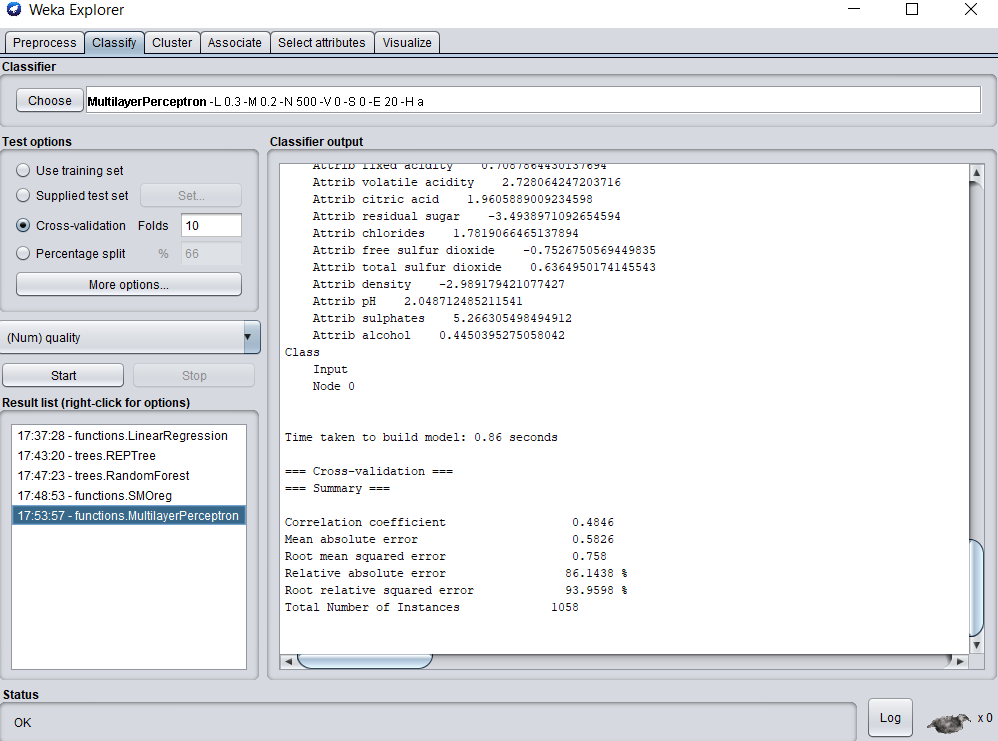
* 1. **Random Forest**



* 1. **SVR**

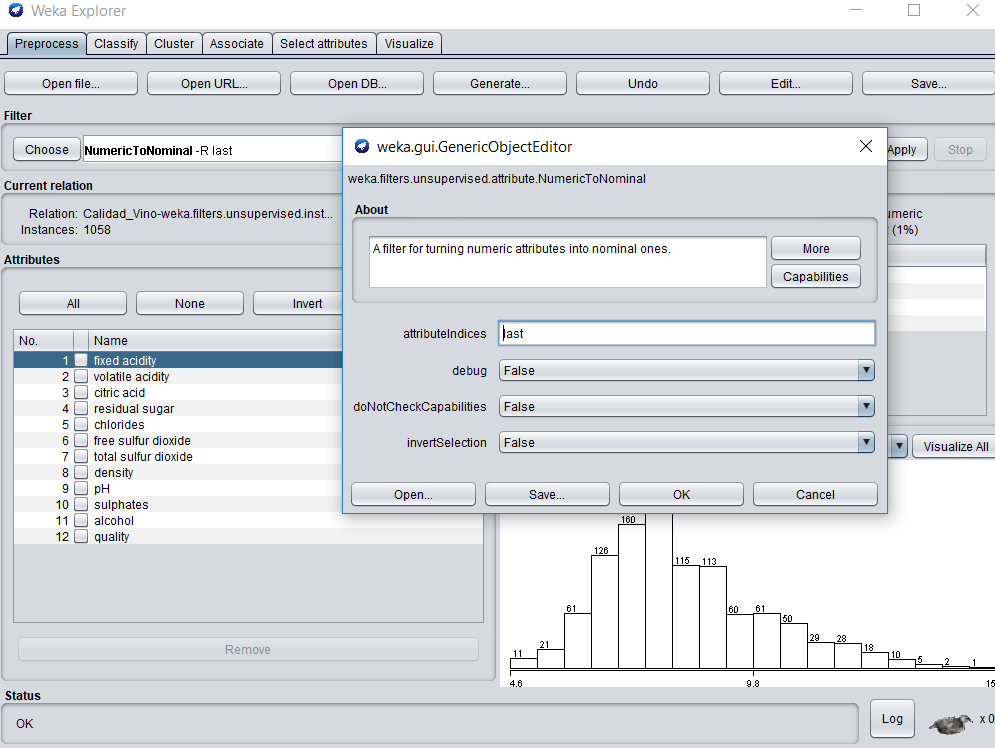


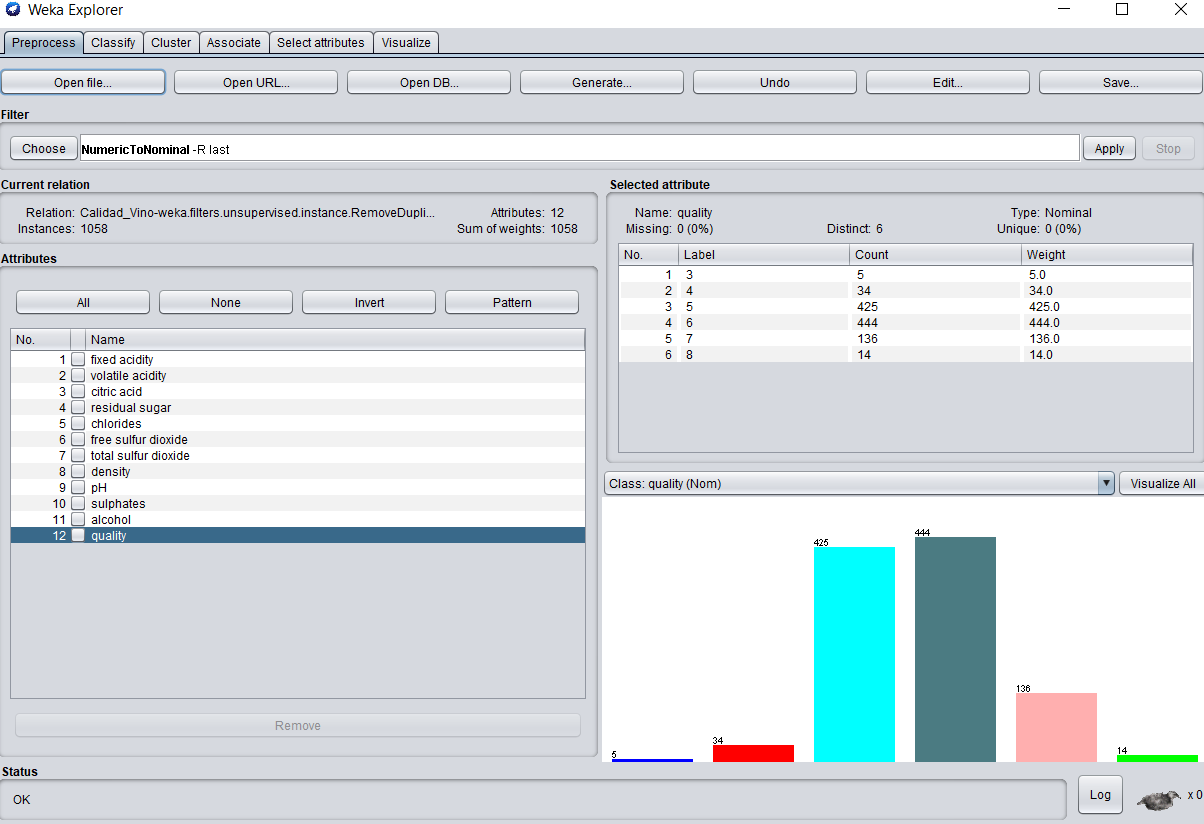
* 1. **Redes Neuronales**



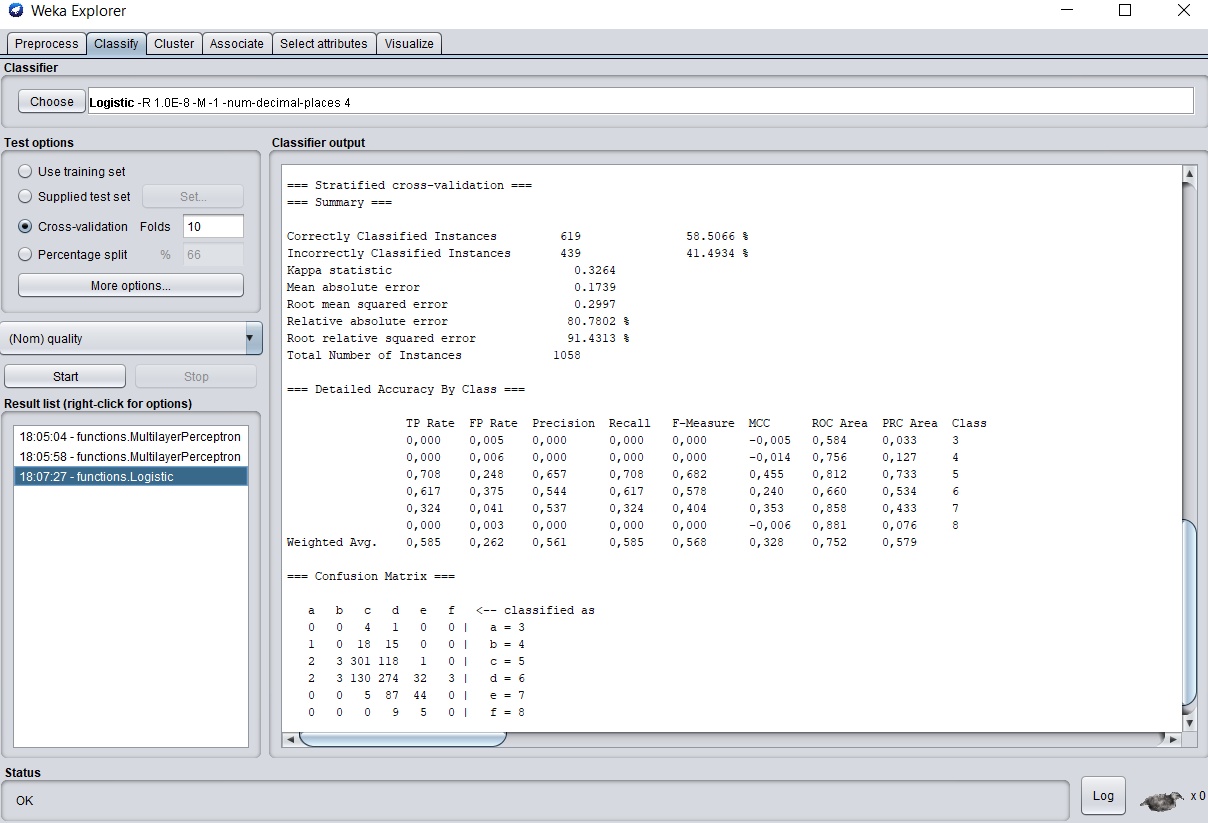
1. **Algoritmos de clasificación**
   1. **Preparación de los datos**

Para aplicar los algoritmos de clasificación, es necesario aplicar una transformación de tipo a la variable de salida. Esto se hace con el filtro NumericToNominal:

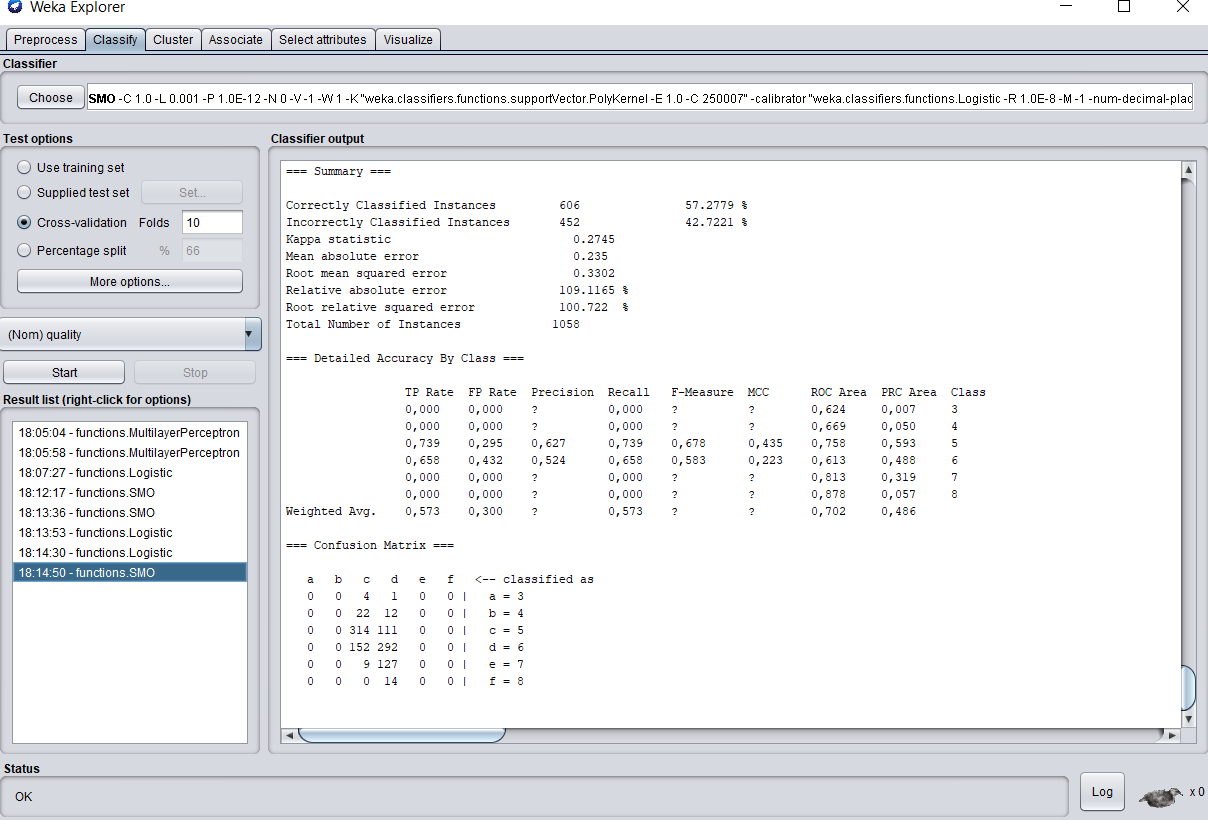




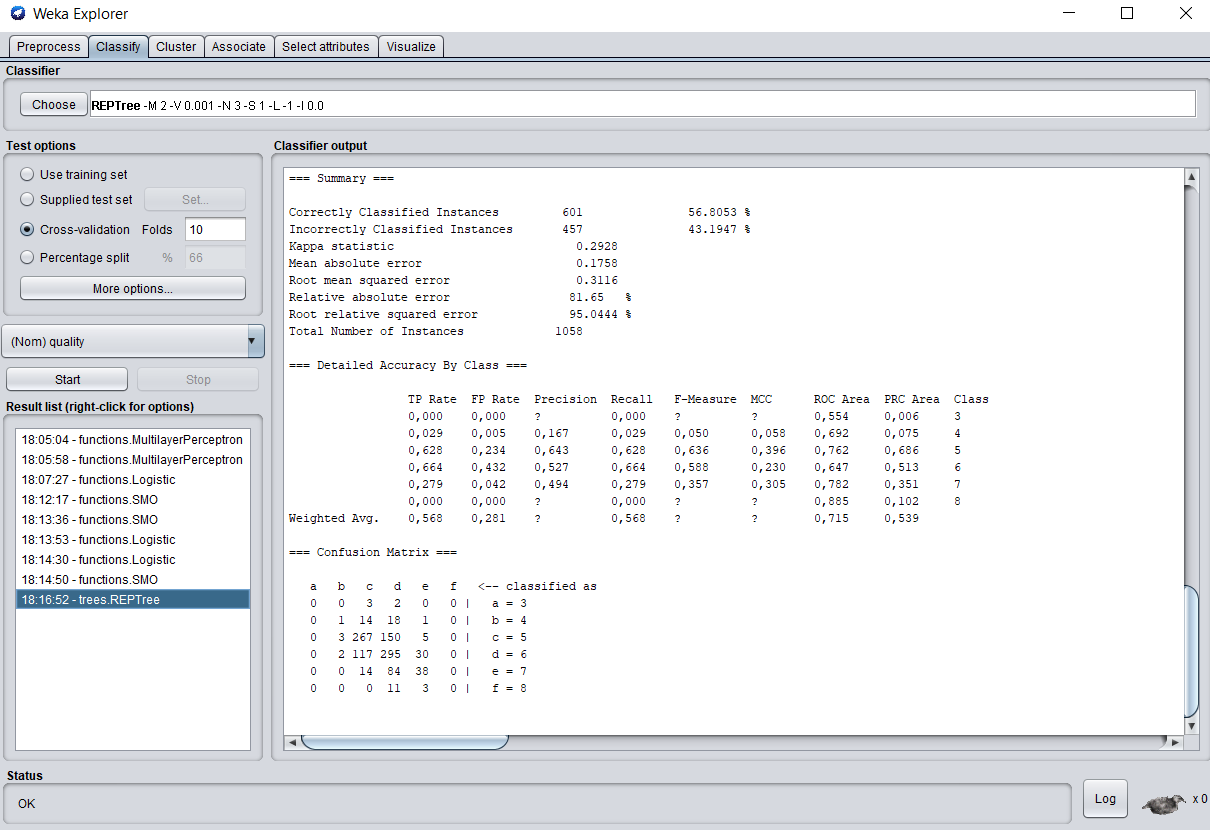
* 1. **Regresión Logística**



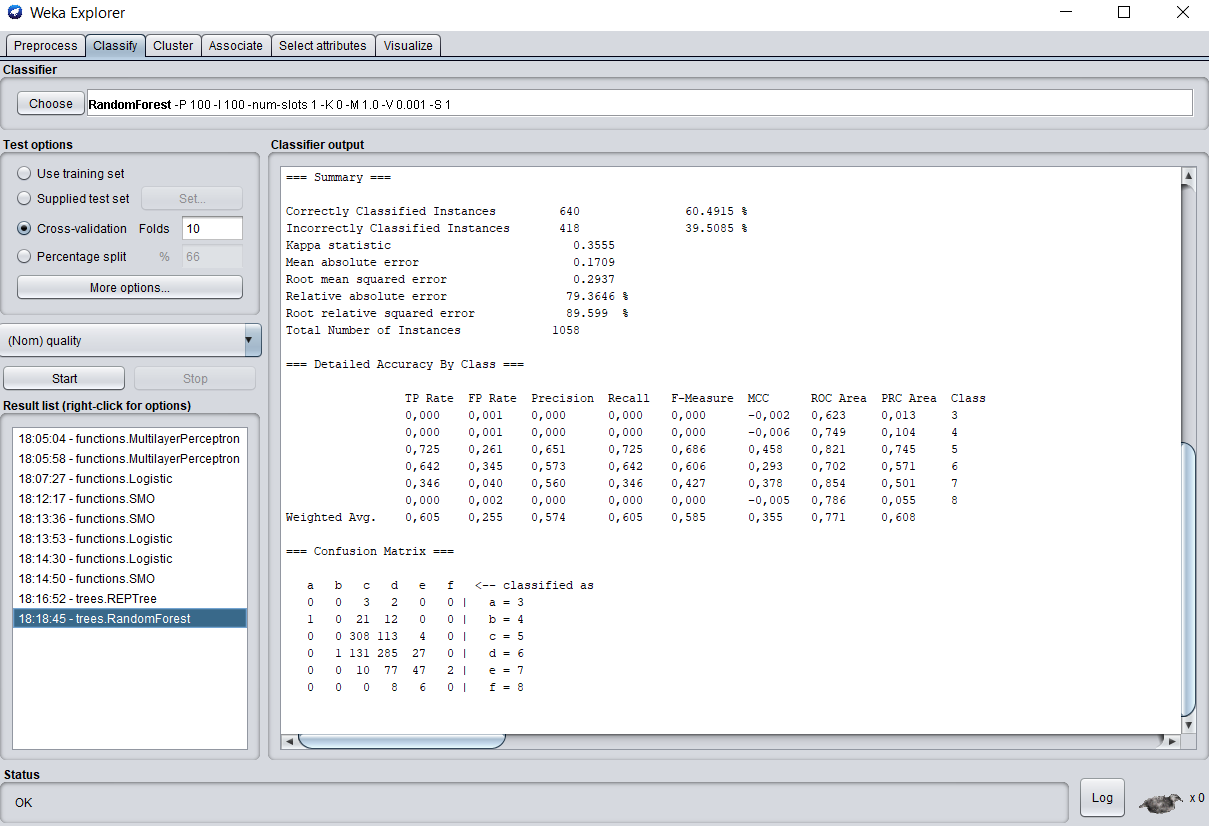
* 1. **SVM**



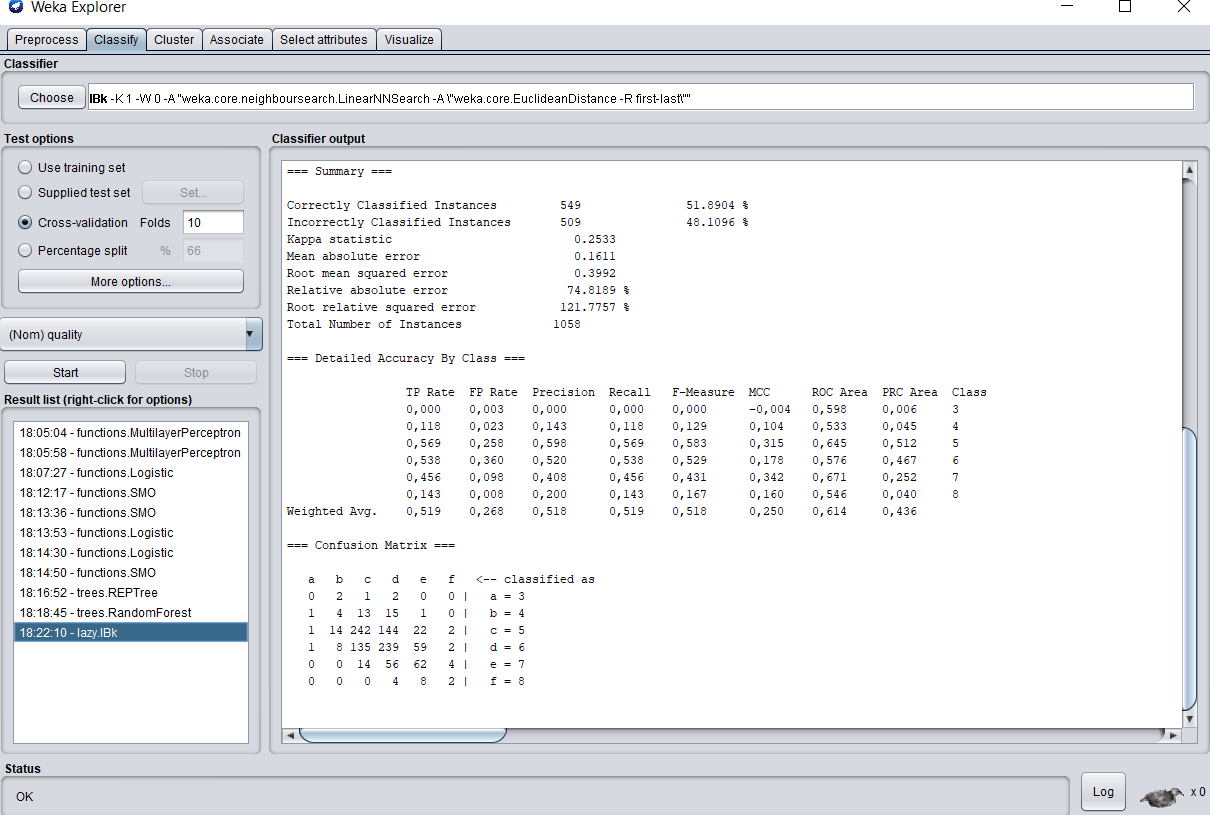
* 1. **Decision Tree**



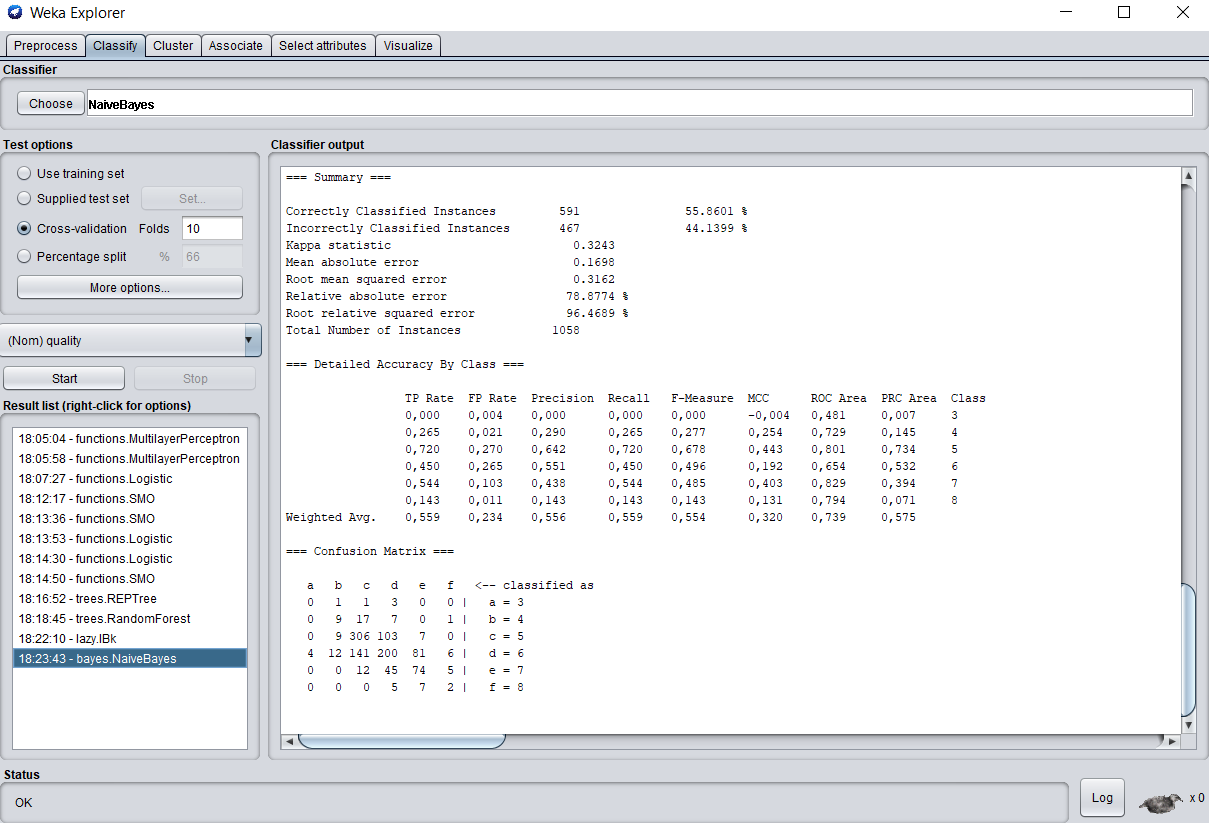
* 1. **Random Forest**



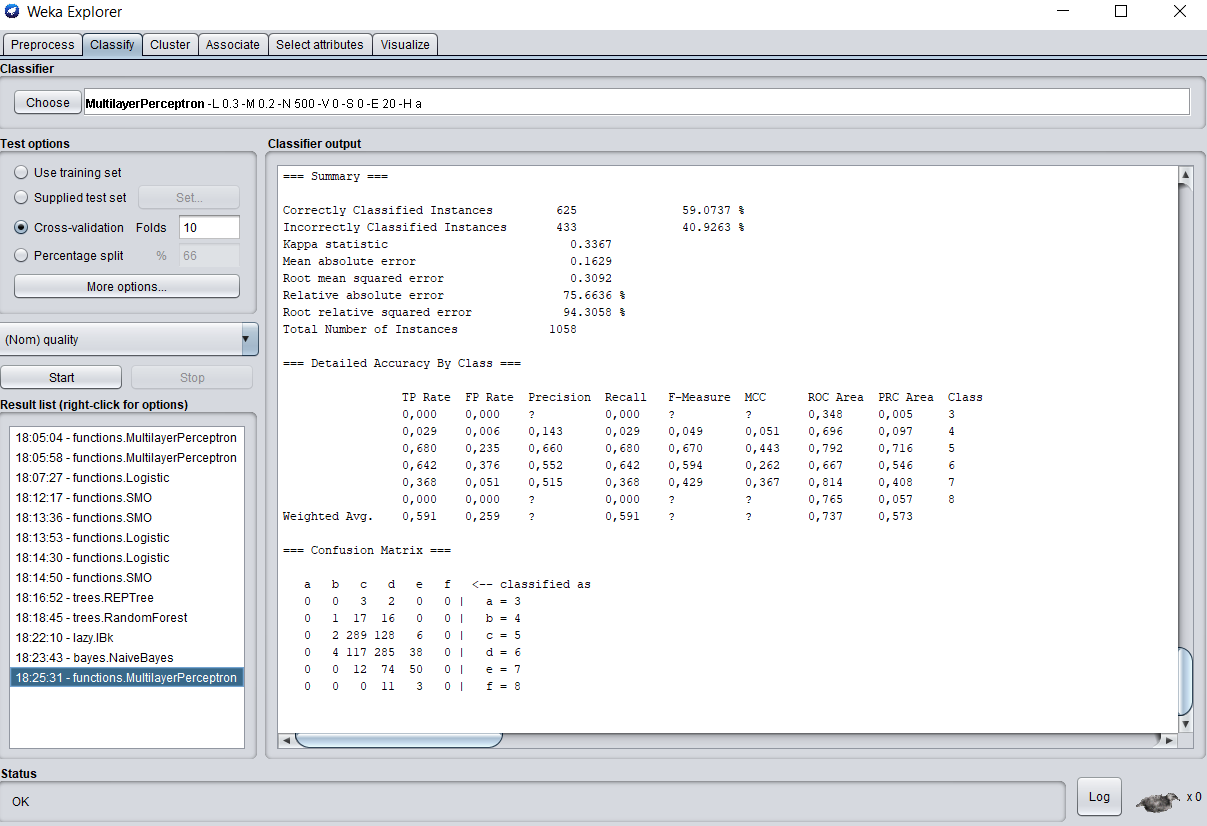
* 1. **K-NN**



* 1. **Naive-Bayes**

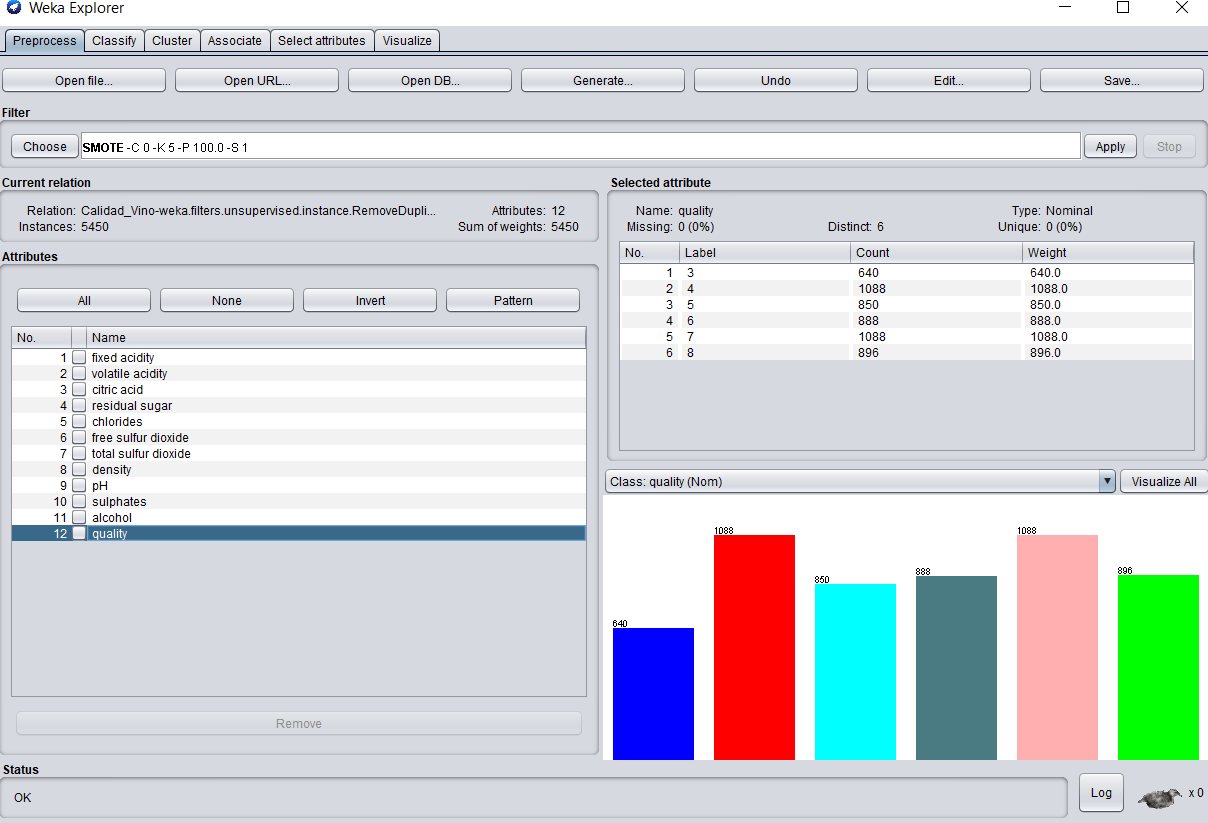


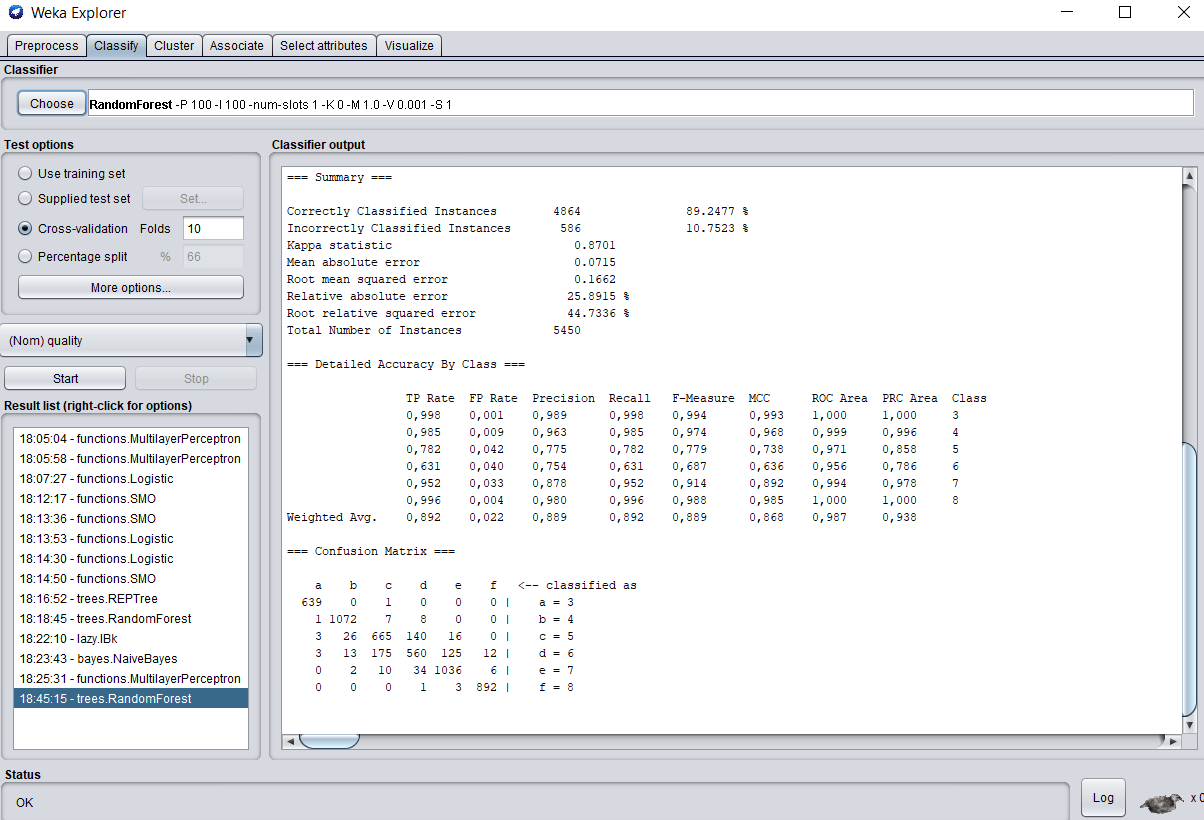
* 1. **Redes Neuronales**

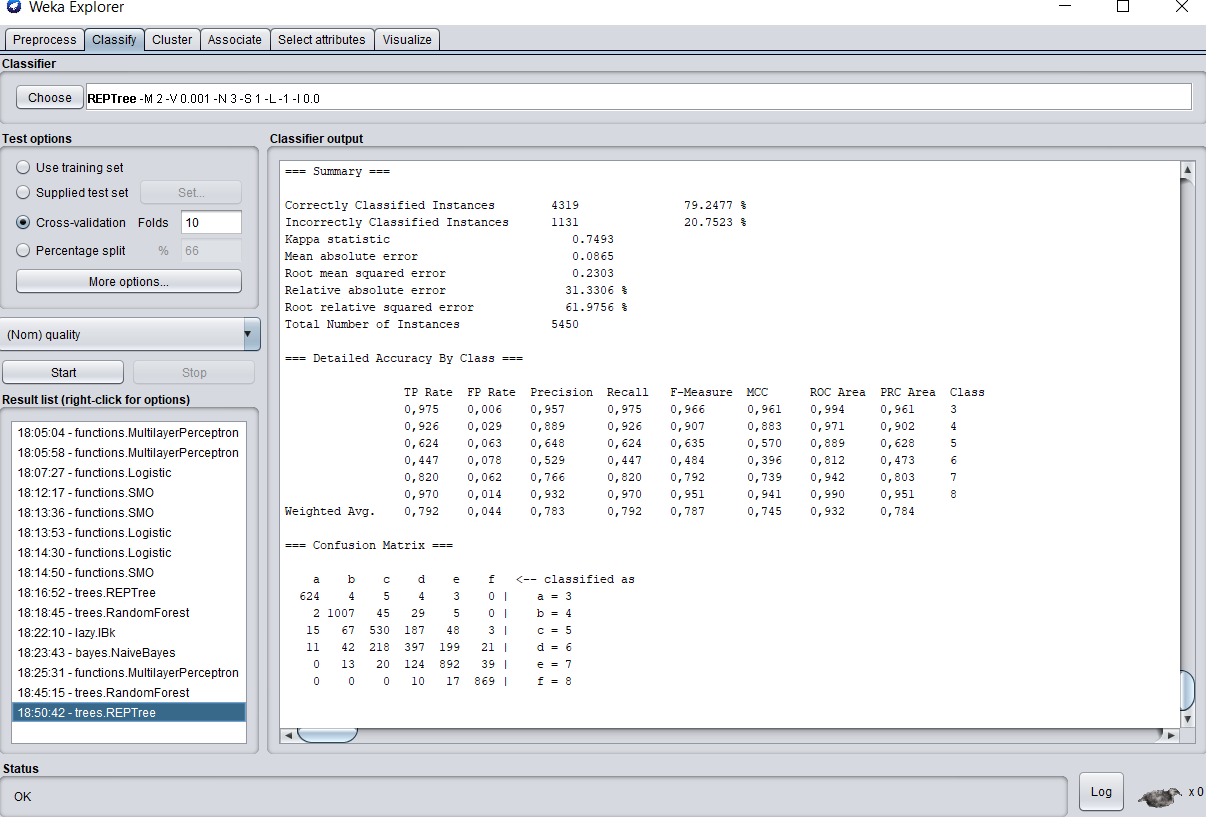


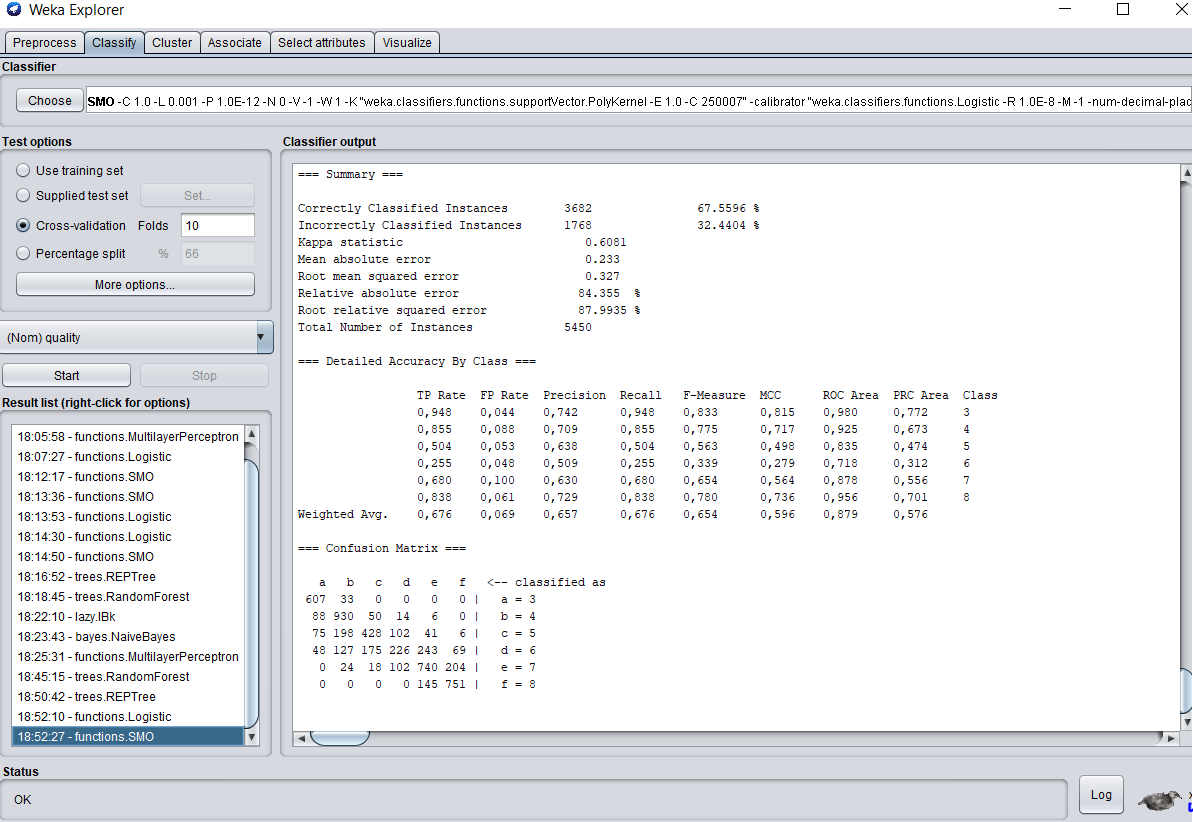
1. **Conclusiones**

* En general, los métodos de clasificación obtuvieron un mejor desempeño que los de regresión. Teniendo en cuenta que los algoritmos de clasificación se calificaron con respecto a su precisión y los de regresión con respecto al RMSE
* El valor de RMSE obtenido para todos los algoritmos de regresión se encuentra muy cercano.
* El algoritmo que mejor se comportó, tanto para regresión como para clasificación fue el random forest. Su precisión en clasificación fue del 60.49% y el RMSE para regresión fue de 0.6309
* Los datos evidentemente se encuentran desbalanceados y esto puede explicar que los valores de precisión y RMSE sean relativamente malos. Se hizo una prueba balanceando el dataset y algunos de los resultados obtenidos se muestran a continuación (cantidad de instancias: 5450):









En general se nota que los rendimientos se incrementaron, sin embargo, hay que tener en cuenta que el balanceo generó más del doble de instancias que se tenían inicialmente.

* En general se observa que en Python se obtuvieron mejores resultados que en WEKA. Esto se puede que explicar porque Python es un poco más flexible para manipular los datos, por tanto, se pudo intervenir más en el proceso de preparación.