



## **Tesina de grado**

Licenciatura en Sistemas de Información.

**Título:** Módulo de predicción con minería de datos aplicado al mercado de divisas.

**Directora:** Daniela López De Luise

**Alumno:** Matias Acevedo

**Fecha de entrega:** 17/02/2022

<b>Desarrollo de tesis</b>	<b>4</b>
Introducción	4
Planteo del Problema	5
Objetivos	5
Propósitos de transferencia	5
Marco Conceptual	6
Mercado de divisas	6
Inteligencia Artificial (IA)	6
Minería de datos	7
Antecedentes	7
Metodología a utilizar	8
Obtener DataSet Inicial	10
Desarrollo de modelos de datos	11
Algoritmo 1 para modelo de datos: Agrupando precios de a 15	11
Algoritmo 2 para modelo de datos: Restando el primer elemento	11
Módulo 1: Bitácora de procesamiento de datos	12
Estudio de datos sobre DataSet_1	12
Utilizando herramienta de calificación de atributos	14
Minería de datos (Estudio del modelo)	17
Utilizando árbol de inducción J48	17
Con redes neuronales: MultilayerPerceptron	24
Conclusión	30
Estudio de datos sobre DataSet_2	31
Utilizando herramienta de selección de atributos	32
Minería de datos (Estudio del modelo)	34
Utilizando árbol de inducción J48	34
Con redes neuronales: MultilayerPerceptron	37
Con el agrupador: MakeDensityBasedClusterer	39
Conclusión	40
Comparación de los Datasets	42
Módulo 2: Validación de modelos predictivos	45
Introducción	45
Generar DataSets de validación	46
Convertir DataSets de formato csv a arff	46
Cargar Datos y Modelo en Weka	48
Resultado de predicción de red neuronal: MultilayerPerceptron	53
Resultado de predicción de metacluster: MakeDensityBasedClusterer	54
Conclusión final	57
Trabajos a futuros	59
Código fuente en Python	60
Anexos Materiales y métodos	60
Evaluador de atributos	60

Selector de atributos	61
Clasificadores de predicción	62
Cronograma	68
Referencias	70

# Desarrollo de tesis

## Introducción

Según Eduardo Estay G. (2011), Los Mercados Financieros, es el lugar donde confluye la oferta y la demanda de Activos Financieros. Si de la confluencia de la oferta y demanda existe acuerdo en el precio, cantidad y fecha de liquidación, nace formalmente una operación financiera negociando el activo financiero.

Estos activos financieros dependen del tipo de mercado al cual pertenecen y se clasifican según los activos negociados:

### **Mercados de Activos Tradicionales:**

- Mercados de depósitos interbancarios
- Mercados de Renta Fija
- Mercados de Renta Variable
- Mercados de Divisas

### **Mercados de Instrumentos Derivados:**

- Mercados de Futuros
- Mercados de Opciones
- Mercados de Swaps

Este trabajo de investigación se centrará en el mercado de activos tradicionales de divisas Forex. De acuerdo con el sitio EFXTO (2021), Forex es el mercado internacional de divisas. Forex es el acrónimo de Foreign Exchange Market. Es el mercado financiero más grande del mundo, con un volumen de negocio superior a los 5.4 trillones de dólares diarios. Forex consiste en la compra-venta de divisas (monedas), esto es, dinero. Las divisas son comerciadas a través de un broker<sup>1</sup> y son comerciadas en pares, por ejemplo euro y dólar estadounidense: (par EUR/USD). En este mercado los traders<sup>2</sup> son quienes operan en el mismo, estos pueden ser, instituciones públicas y privadas como bancos y empresas o personas inversores. Estos actores que son quienes operan en este mercado deben anticiparse a la suba o baja del precio de la divisas teniendo en cuenta diferentes factores que afectan o alteran al precio.

En este trabajo se buscará automatizar este proceso de anticipación al precio, a través del análisis y procesamiento de datos mediante técnicas de inteligencia artificial como son las series temporales, todo esto conlleva al desarrollo de un módulo inteligente para tal fin.

---

<sup>1</sup> Un bróker es una empresa financiera que se encarga de la negociación de instrumentos financieros como divisas conectando las operaciones de los traders con el mercado internacional de divisas.

<sup>2</sup> Un trader es una persona o una entidad financiera que opera en los mercados financieros.

## Planteo del Problema

El mercado de Forex está influenciado por múltiples acontecimientos mundiales que afectan a la oferta y la demanda del mismo. De esta manera pronosticar la suba o baja del precio no solo depende de ciertas variables como, un análisis sobre patrones técnicos a lo largo del tiempo, sino también de otros datos fundamentales como sucesos sociales, decisiones políticas de un país o un banco central, el PBI de un país, etc.

El problema se basa en las múltiples variables que un trader debe tener en cuenta a la hora de tomar decisiones en el mercado de divisas, cada una de estas variables influyen de forma drásticamente en el comportamiento del precio, es decir, influyen en si el precio va a subir o bajar. A estas variables nombradas anteriormente se le puede agregar otro tipo de variables técnicas que son los patrones técnicos. De acuerdo a Francisca Serrano (2019), los patrones técnicos “son figuras geométricas que el precio va creando en un gráfico con sus movimientos alcistas, bajistas o laterales”.

Estas múltiples variables son un problema para los traders a la hora de ingresar al mercado, no solo por la cantidad sino también por la complejidad de interpretación que tienen. Es por eso que surge la necesidad de poder automatizar el proceso de análisis del mercado de divisas con el objetivo de a partir de estas variables, generar aproximaciones o tendencias del precio. Esto le generará al trader tener mayor seguridad al momento de realizar una operación de compra (si el precio baja) o venta (si el precio sube).

## Objetivos

Objetivo principal:

- Realizar recomendaciones de compras o ventas a traders que operan en el mercado de divisas Forex, mediante diferentes modelos de predicción.

Objetivos específicos:

- A partir de los datos iniciales seleccionar variables relevantes y compilar datos.
- Analizar datos y desarrollar dos modelos predictivos de adaptación a series temporales.
- Mediante weka utilizar los modelos de predicción desarrollados mostrando B (si el precio baja), I (si se mantiene) o S (si sube).

## Propósitos de transferencia

Implementar una herramienta liviana y práctica, fácilmente parametrizable y con características de autoajuste para ponerla a disposición de la industria y el público como un servicio.

# Marco Conceptual

## Mercado de divisas

Según Efrain Cazar M. (2021), las divisas son activos financieros mantenidos por residentes de un país y que constituyen una obligación por parte de un residente de otro país emisor de una moneda diferente. Las divisas se negocian en el mercado internacional de divisas, que es donde se encuentran oferentes y demandantes, es decir es donde se compran y venden las divisas. Además aquí se establece el valor de cambio de las monedas en que se van a realizar los flujos monetarios internacionales.

De acuerdo a Asturias Corporación Universitaria (2012), existen algunas características especiales como:

- Es un mercado OTC (Over the Counter). No es un mercado organizado.
- No tiene un horario definido y opera todos los días y en cualquier horario.
- Es un mercado descentralizado y no existe un único lugar de encuentro (se mueve alrededor del mundo).
- Se configura como un mercado interbancario, con negociación telefónica, que puede ser directa entre las entidades participantes o con la intermediación de un broker.
- El tipo de cambio se establece libremente por el cruce entre oferta y demanda. La información sobre los precios circula automáticamente a través de plataformas de información (Reuters, Bloomberg, Telerate...)
- Es un mercado muy volátil (grandes variaciones en el precio)

## Inteligencia Artificial (IA)

De acuerdo a Julio Cesar Ponce (2014), actualmente la IA es un área de la ciencia de gran interés por ser un área multidisciplinaria donde se realizan sistemas que tratan de hacer tareas y resolver problemas como lo hace un humano, así mismo se trata de simular de manera artificial las formas del pensamiento y cómo trabaja el cerebro.

A través de la IA, se pueden desarrollar módulos aplicados a los mercados financieros, estos pueden tomar decisiones basadas en la gran cantidad de información y de datos que tiene disponible. Analiza todos estos datos complejos de los mercados financieros a los cuales tiene acceso y actualiza los algoritmos de forma automática en base a los resultados anteriores, mejorando así la precisión en el proceso y la toma de decisiones.

Para resolver el problema planteado en este trabajo de investigación, aplicaremos específicamente la minería de datos que se basa en inteligencia artificial, Machine Learning y estadística computacional.

## Minería de datos

Según MSc. Mabel Gonzáles Castellanos (2013), La minería de datos es el proceso de descubrir nuevas correlaciones significativas, modelos y tendencias, filtrando grandes cantidades de datos, a través del uso de tecnologías de reconocimiento de modelos así como de técnicas de estadística y matemáticas.

El objetivo de este proceso es revelar patrones desconocidos a partir de los datos y resolver problemas con enormes conjuntos de datos y relaciones muy complejas entre ellos.

Para resolver el problema de este trabajo, hay que recurrir a la **minería de datos para series temporales** ya que se requiere predecir el movimiento del precio en un determinado rango de tiempo. También la naturaleza de las series temporales hace que su tratamiento se diferencie de los métodos tradicionales de minería de datos. Diferencias como: alta numerosidad, gran número de dimensiones y una constante actualización de sus datos al transcurrir el tiempo.

La minería de datos para series temporales es una contribución importante a los campos de estudio de la minería de datos y de las series temporales.

Según MSc. Mabel Gonzáles Castellanos Lic. César Soto Valero (2013), requiere tener claramente definidos cuáles serán los eventos que se van a “minar”. De manera similar es necesario definir las formaciones que apuntan a eventos significativos, para nuestro problema estos casos pueden ser sucesos sociales que afecten al precio de alguna divisa. En el contexto de la minería de datos para series temporales estas formaciones son llamadas patrones temporales. Un patrón temporal puede estar asociado a un evento por lo cual es necesario en la predicción de los eventos.

Además es necesario recurrir a diferentes métodos de tratamientos de datos los cuales se clasifican en categorías, que agrupan los diferentes tipos de tareas existentes para la minería de datos, correspondiendo a los objetivos del análisis y los tipos de problemas que enfrentan.

- Clasificación
- Agrupamiento
- Asociación
- Predicción
- Regresión

Para desarrollar este módulo inteligente, es necesario contar con dos categorías de algoritmos de resolución de problemas como son: Métodos de agrupación y Métodos de predicción Según Ian H. Witten; Eibe Frank; Mark A. Hall. (2011).

## Antecedentes

Existen antecedentes de trabajos similares al este mismo que se han desarrollado mediante técnicas de minería de datos con series temporales. A continuación describimos algunas de esos ejemplos:

- De acuerdo a Andrés Pablo Moggi de la Universidad de Buenos Aires (2013), las series temporales fueron implementadas en el sector comercial más específicamente en venta de pasajes.

En diferentes industrias surge la necesidad de pronosticar eventos con el objetivo de poder planear de un modo más eficiente las decisiones a tomar. En particular, el sector aerocomercial ve esa necesidad con mayor avidez ya que presenta muchos actores, escasos márgenes de ganancias y altos costos, fundamentalmente los producidos por el combustible. Por tal motivo, poder pronosticar la demanda de pasajeros consiste en un trabajo fundamental. Este tipo de información permite tomar acertadamente decisiones tales como aumento y/o disminución de frecuencias, cambios de equipos, inversión en equipos nuevos, aumento y/o disminución de tarifas, entre otros.

Para modelar este módulo, se introducen las series de tiempo estacionarias y no estacionarias.

- Resultados de Edwin Melo Mayta de la Facultad de Ingeniería estadística e informática. (2016). Uso de la metodología *Box – Jenkins*<sup>3</sup> para modelar series temporales.

En el Hospital Regional Manuel Núñez Butrón-Puno, es una institución pública que está dedicada exclusivamente a la actividad de servicio de recuperación y rehabilitación de los pacientes, para lo cual una de las prioridades de estudio es saber las futuras proyecciones del comportamiento de la mortalidad general intrahospitalaria en el Hospital Regional “Manuel Núñez Butrón” para prever recursos humanos, de infraestructura, equipamiento, tecnológicos y financieros y así reducir el incremento de muertes, motivo por el cual se tiene como objetivo determinar el mejor modelo de predicción mensual que se ajuste a la serie original para hacer predicciones a corto plazo. La metodología usada fue Box – Jenkins para el modelamiento de la serie del número de mortalidad general intrahospitalaria, la cual se desarrolló en las siguientes etapas de exploración de la serie, para la identificación del modelo, estimación de los parámetros del modelo, verificación de modelo y finalmente usar el modelo apropiado para la predicción, el resultado de la estimación del mejor modelo univariante para la predicción de la serie original de mortalidad intrahospitalaria, en el Hospital Regional Manuel Núñez Butrón.

## Metodología a utilizar

El módulo inteligente se desarrollará aplicando métodos de agrupación natural (específicamente Expectation Maximization y K-means), contrastando con SubsetEval como mecánica de selección y validación de las variables a trabajar.

Luego se construirá el modelo con un árbol de inducción (de la familia ID3) o con redes Naive Bayes, dependiendo de la confiabilidad que arroje el método luego de su aplicación sobre los datos compilados.

A los fines de la predicción temporal, los datos serán modificados en formato acorde para secuencias temporales en el marco de tiempo parametrizado (15 minutos).

Entre otras cuestiones se evaluará la necesidad de filtrar datos de baja relevancia y la posibilidad de desarrollar módulos de software complementarios.

---

<sup>3</sup> La metodología Box-Jenkins permitirá predecir valores futuros de una serie basándose en valores pasados de una sola variable o más.



Para los desarrollos se emplearán las librerías de Python<sup>4</sup> y las plataforma WEKA<sup>5</sup> para la validación de resultados.

Además de los datos históricos/técnicos que tienen que ver con el comportamiento del precio asociados a la oferta y demanda, existen otros factores importantes que afectarán de alguna u otra manera al comportamiento del precio, como el impacto social de algún evento como por ejemplo una pandemia, el PBI de un país, decisiones políticas, entre otras. En este trabajo se tendrán en cuenta estos últimos factores que afectan a los mercados financieros sólo conceptualmente y no a modo de implementar un módulo que obtenga todos estos datos.

Si bien esta herramienta que desarrollaremos tendrá en cuenta el precio histórico de este mercado pero no tendrá en cuenta sucesos o acontecimientos sociales, por lo tanto esta herramienta solo será un método de confirmación que el trader como usuario utilizará para poder operar en el mercado de divisas de una manera más segura.

---

<sup>4</sup> Python es un lenguaje de programación, de propósito general, de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web.

<sup>5</sup> Weka proporciona implementaciones de algoritmos de aprendizaje que puede aplicar fácilmente a su conjunto de datos. También incluye una variedad de herramientas para transformar conjuntos de datos

## Obtener DataSet Inicial

El primer paso de este desarrollo es obtener el dataset inicial que permitirá construir el modelo. Se buscó datos sobre el mercado de divisas de Forex. Según Forex Tester (2021), los datos recolectado en el dataset se realizan cada minuto, es decir, que en cada rango de tiempo de un minuto releva los movimientos del precio.

Este dataset inicial contiene las siguientes columnas: (Date, Hour, Open, High, Low, Close).

Columnas del DataSet:

**Divisa:** es en nombre de la divisa ej. EUR-USD.

**Date:** Es la fecha en la cual se obtiene el precio.

**Hour:** Es la hora en la cual se obtiene el precio.

**Open:** Es el precio de apertura en un determinado tiempo.

**High:** Es el precio más alto en el cual llegó en un determinado tiempo, entre el precio Open y el precio Close.

**Low:** Es el precio más bajo en el cual llegó en un determinado tiempo, entre el precio Open y el precio Close.

**Close:** Es el precio de cierre de la divisa en un determinado tiempo.

Los datos obtenidos en el dataset inicial corresponden a el EUR-USD únicamente, esto significa que el modelo final estará basado y listo para predecir el EUR-USD..

Este dataset tiene una columna que es la más importante, es el precio de cierre (Close), debido que muestra el precio con el que finalizó en ese rango de tiempo, que en este caso es de 1 minuto. Esto permite poder desarrollar un módulo en python, agrupando los precios de la columna Close en vectores por ejemplo de 15 elementos, para el caso de querer predecir la suba o baja del precio en un tiempo de 15 minutos, aprovechando la ventaja de que el dataset registra el movimiento de una divisa cada 1 minuto.

Para identificar este dataset a lo largo de este trabajo le asignaremos un nombre, se llamará *DataSet\_Inicial.csv*

Para poder realizar agrupaciones de datos de a 15 elementos, mediante python se desarrollan dos módulos que son dos algoritmos diferentes para poder obtener dos dataset y de esta manera realizar comparaciones a medida que se vaya creando el modelo.

## Desarrollo de modelos de datos

### Algoritmo 1 para modelo de datos: Agrupando precios de a 15

Para este algoritmo se utilizó únicamente la columna Close, que indica el precio de cierre en el rango de tiempo de 1 minuto.

El algoritmo desarrollado en Python consiste en, a partir del primer elemento, lista en un vector el primero más los siguientes 14 elementos, luego en un segundo vector lista el segundo elemento más los 14 elementos que los siguen, luego con el tercero. De esta manera continúa listando hasta finalizar la lista de elementos.

Al finalizar cada iteración se agrega una columna llamada "Class", que consiste en un dato de variación calculado a partir de la diferencia entre el primer y último elemento del vector generado. Esta última columna contendrá una "S" (Sube) si el último es mayor al primero, "I" (Igual) si el primero es igual al último y "B" (Baja) si el primer elemento es mayor al último.

En el módulo de python se utiliza el método *GenerarDataSet(1, 'DataSet\_1.csv', 'DataSet\_Inicial.csv')* donde se le pasa como parámetro en datasetNum el número 1, el segundo parámetro (fileNameGenerated) es el nombre del archivo que se genera y el tercer parámetro (fileNameToProces) es el nombre del archivo inicial el cual buscará para generar el dataset. Este módulo carga el dataset *DataSet\_Inicial* y a partir de esto procesa toda la información de la columna "Close" con la cual genera un archivo csv que en este trabajo se llamará **DataSet\_1**.

	A	B	C	D	E	F	G	H	
1	Col1,Col2,Col3,Col4,Col5,Col6,Col7,Col8,Col9,Col10,Col11,Col12,Col13,Col14,Col15,Class								
2	12.230,12.230,12.229,12.229,12.230,12.230,12.231,12.232,12.233,12.234,12.233,12.233,12.233,12.232,12.232,S								
3	12.230,12.229,12.229,12.230,12.230,12.231,12.232,12.233,12.234,12.233,12.233,12.233,12.232,12.232,12.232,S								
4	12.229,12.229,12.230,12.230,12.231,12.232,12.233,12.234,12.233,12.233,12.233,12.232,12.232,12.232,12.231,S								
5	12.229,12.230,12.230,12.231,12.232,12.233,12.234,12.233,12.233,12.233,12.232,12.232,12.232,12.231,12.230,S								
6	12.230,12.230,12.231,12.232,12.233,12.234,12.233,12.233,12.233,12.232,12.232,12.232,12.231,12.230,12.230,I								
7	12.230,12.231,12.232,12.233,12.234,12.233,12.233,12.233,12.232,12.232,12.232,12.231,12.230,12.230,12.229,B								
8	12.231,12.232,12.233,12.234,12.233,12.233,12.233,12.232,12.232,12.232,12.231,12.230,12.230,12.229,12.229,B								
9	12.232,12.233,12.234,12.233,12.233,12.233,12.232,12.232,12.232,12.231,12.230,12.230,12.229,12.229,12.229,B								
10	12.233,12.234,12.233,12.233,12.233,12.232,12.232,12.232,12.231,12.230,12.230,12.229,12.229,12.229,12.229,B								
11	12.234,12.233,12.233,12.233,12.232,12.232,12.232,12.231,12.230,12.230,12.229,12.229,12.229,12.229,12.230,B								
12	12.233,12.233,12.233,12.233,12.233,12.233,12.233,12.231,12.230,12.230,12.230,12.230,12.230,12.230,12.230,B								

Imagen 0 (*DataSet\_1.csv*)

### Algoritmo 2 para modelo de datos: Restando el primer elemento

Para este algoritmo se utiliza la columna Close, comenzando desde el primer elemento (precio). En la primera iteración se tendrá como entrada un vector con los primeros 16 elementos.

Ejemplo. Columna Close de 15 elementos (10, 5, 45, 8, 6, 7, 12, 14, 32, 15, 11, 18, 10, 14, 15, 10)

Comenzando desde el primer elemento, se le restará el segundo y se lo agrega a un vector (vector de salida), luego se le restará el tercero y se lo agrega al vector de salida, y así

sucesivamente hasta llegar al número 16 llamada. En la última columna llamada "Class" se agrega un dato de variación calculado a partir de la diferencia entre la primera columna y la última. Esta última columna contendrá una "S" (Sube) si el último es mayor al primero, "I" (Igual) si el primero es igual al último y "B" (Baja) si el primer elemento es mayor al último. Posteriormente se elimina el primer elemento y se agrega uno al final (el siguiente) y se procederá con las restas del primer elemento de turno.

Vector de salida (5, -35, 2, 4, 3, -2, -4, -22, -5, -1, -8, 0, -4, -5, 0, B)

En este caso en la última columna, el algoritmo pondrá B, ya que el último elemento es menor al primero (Baja).

Para este algoritmo se desarrolló un módulo en python. Se Utiliza el método *GenerarDataSet(2, 'DataSet\_2.csv', 'DataSet\_Inicial.csv')* donde se le pasa como parámetro en datasetNum el número 2, el segundo parámetro (fileNameGenerated) es el nombre del archivo que se genera y el tercer parámetro (fileNameToProces) es el nombre del archivo inicial en el cual se basará para generar el dataset. Este módulo carga el dataset *DataSet\_Inicial* y a partir de esto procesa toda la información de la columna "Close" con la cual se genera un archivo csv que en este trabajo lo llamaremos **DataSet\_2**.

Ejemplo tomado de DataSet\_2.csv

*Col1,Col2,Col3,Col4,Col5,Col6,Col7,Col8,Col9,Col10,Col11,Col12,Col13,Col14,Col15,Class*

*0.0,0.00100000000000012221,0.00100000000000012221,0.0,0.0,-0.0009999999999994458,-0.00199999999999988916,-0.0030000000000001137,-0.0039999999999995595,-0.0030000000000001137,-0.0030000000000001137,-0.0030000000000001137,-0.00199999999999988916,-0.00199999999999988916,-0.00199999999999988916,B*

*0.00100000000000012221,0.00100000000000012221,0.0,0.0,-0.0009999999999994458,-0.00199999999999988916,-0.0030000000000001137,-0.0039999999999995595,-0.0030000000000001137,-0.0030000000000001137,-0.0030000000000001137,-0.00199999999999988916,-0.00199999999999988916,-0.00199999999999988916,-0.0009999999999994458,B*

## Módulo 1: Bitácora de procesamiento de datos

### Estudio de datos sobre DataSet\_1

Se comienza con el primer dataset el cual se llama **DataSet\_1** obtenido a partir de la ejecución del algoritmo 1. Lo primero que se hizo fue cargar el archivo csv a weka obteniendo los siguientes resultados:

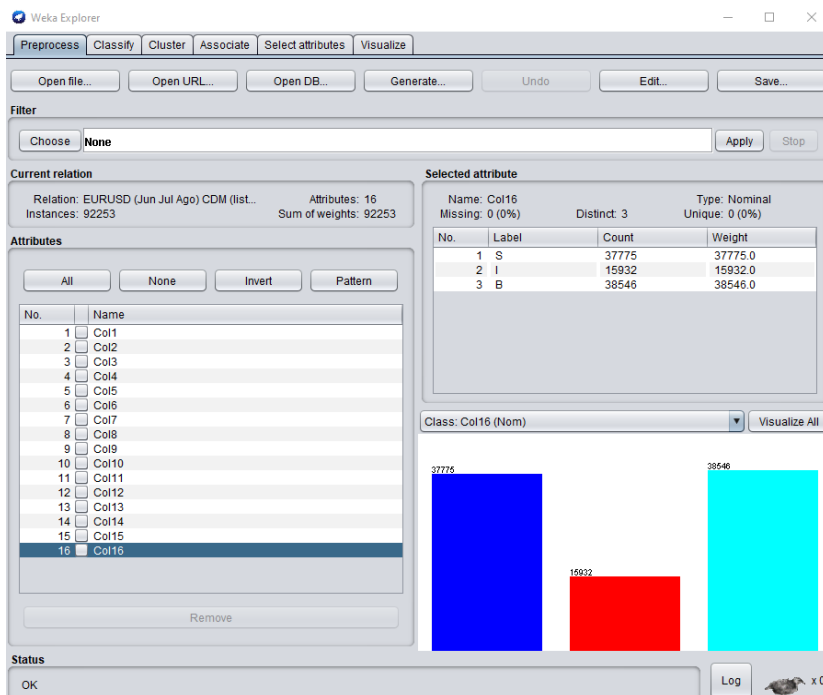


Imagen 1

La información que muestra Weka en la imagen 1, luego de cargar el archivo es que tiene 92.253 (Instances) registros con 16 columnas (Attributes).

Si se selecciona la Col16 de tipo Nominal, se puede ver los tres valores que contiene la columna que son “S”, “I” y “B” con la cantidad de datos que van a predecir cada uno. Para S encontré 37.775, para I 15.932 y para B 38.546.

Para las demás columnas, desde la columna 1 (Col1) a la columna 15 (Col15) Weka detecta un máximo de 12.252 un mínimo de 11.663 y un precio medio de 11.879. También detecta que es de tipo Numérico (Numeric), Datos perdidos 0% (Missing), datos distintos 551 (Distinct), datos únicos 27 (Unique) y el Name que indica el nombre de la columna.

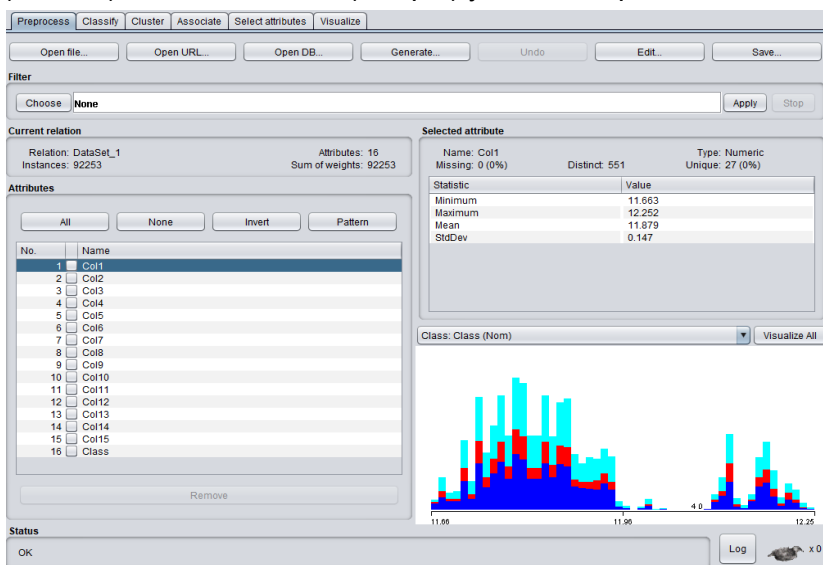
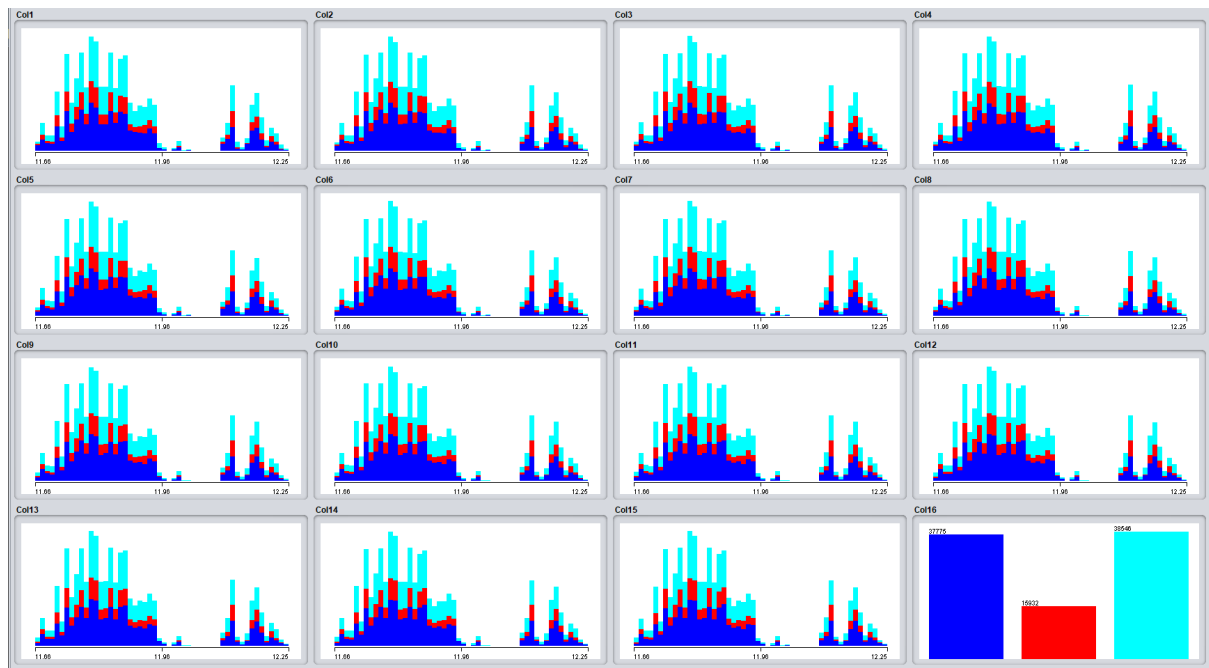


Imagen 2

Se puede notar que en el gráfico tenemos un comportamiento específico llamado multimodal, esto significa que hay una gran distribución de los datos por lo que existe una amplia variación en una misma columna de los mismos. También se nota que si se mueve de la columna 1 a la 15 no hay una variación marcada del gráfico entre los datos de las columnas, sino que los datos son muy similares unos de otros representados en los gráficos.

Como se puede ver en la *Imagen 3* la variación de datos va desde la columna 1 a la 15 y la columna 16 muestra los datos “S”, “I” y “B”.



*Imagen 3*

### Utilizando herramienta de calificación de atributos

Este módulo de análisis de datos de Weka se utiliza para determinar qué tan relevantes son las columnas que se tiene, que luego serán utilizadas para predecir, para este caso de estudio es la suba o la baja del precio.

Se utilizó el selector de atributos Ranker para poder saber qué columnas son las más relevantes al momento de predecir.

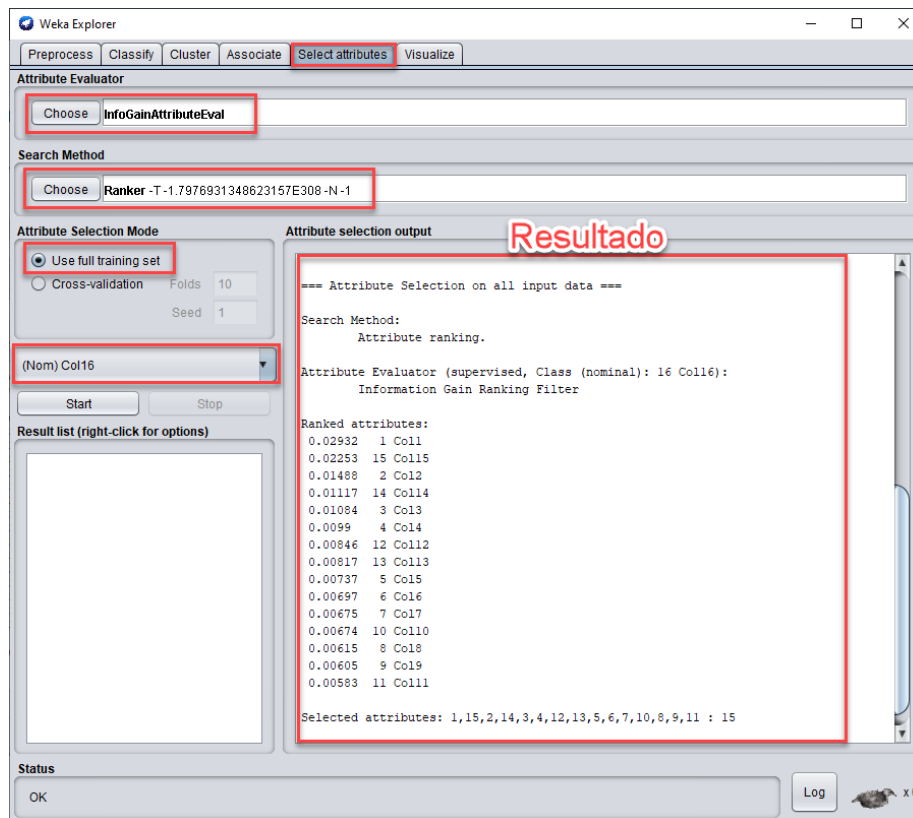


Imagen 4

Para eso se selecciona el “Attribute Evaluator” llamado “InfoGainAttributeEval”, luego en “Search Method” se selecciona “Ranker”, en “Attribute Selection Mode” se lo deja en “Use full training set”<sup>6</sup>, y por último se selecciona la única columna de tipo Nominal 16 (contiene “S” o “I” o “B”) para realizar la clasificación, es decir, es la columna en la cual el Ranker se basa para realizar la clasificación. De esta manera la herramienta Ranker retorna el ranking de columnas según cuánto podrían contribuir al momento de predecir. En el primer puesto queda la columna 1 llamada “Col1” y en el segundo puesto la columna 15 llamada “Col15”, esto tiene lógica ya que para obtener la columna 16 del DataSet\_1, se ha tenido en cuenta la columna 1 y 15, es decir, la columna 1 y 15 se están utilizando para calcular la columna Class, que es “S” o “I” o “B” dependiendo si el precio de la columna 15 baja, sube o queda igual con respecto al precio de la columna 1.

Dicho selector arrojó la siguiente información que muestra el ranking de atributos: (Imagen 4.1)

<sup>6</sup> La opción “Use full training set” significa que se va a usar todos los datos para generar el ranking de atributos relevantes. A diferencia del “Cross-validation” utiliza menos cantidad de datos y obtiene una validación para ver que tan buenos son esos datos.

```

=== Attribute Selection on all input data ===

Search Method:
  Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 16 Col16):
  Information Gain Ranking Filter

Ranked attributes:
0.02932   1 Col1
0.02253  15 Col15
0.01488   2 Col2
0.01117  14 Col14
0.01084   3 Col3
0.0099    4 Col4
0.00846  12 Col12
0.00817  13 Col13
0.00737   5 Col5
0.00697   6 Col6
0.00675   7 Col7
0.00674  10 Col10
0.00615   8 Col8
0.00605   9 Col9
0.00583  11 Col11

Selected attributes: 1,15,2,14,3,4,12,13,5,6,7,10,8,9,11 : 15

```

#### *Imagen 4.1*

Obsérvese también que el atributo 16 no aparece ya que es el target seleccionado para calcular el ranking de atributos.

En el siguiente caso, la prueba que se hizo fue eliminar la columna 1 y 15 para poder ver el comportamiento de las demás columnas con respecto a la columna 16 (columna target). En la Imagen 4.2, podemos ver que el orden de las columnas a partir de la 1 y 15, que ya no existen, es el mismo, esto significa que todas las columnas (excepto la 1 y 15) están aportando de forma análoga de la misma manera que el caso anterior (cuando se tenía la columna 1 y 15), a la columna 16.

```

=== Attribute Selection on all input data ===

Search Method:
  Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 14 Col16):
  Information Gain Ranking Filter

Ranked attributes:
0.01488   1 Col2
0.01117  13 Col14
0.01084   2 Col3
0.0099    3 Col4
0.00846  11 Col12
0.00817  12 Col13
0.00737   4 Col5
0.00697   5 Col6
0.00675   6 Col7
0.00674   9 Col10
0.00615   7 Col8
0.00605   8 Col9
0.00583  10 Col11

Selected attributes: 1,13,2,3,11,12,4,5,6,9,7,8,10 : 13

```

#### *Imagen 4.2*



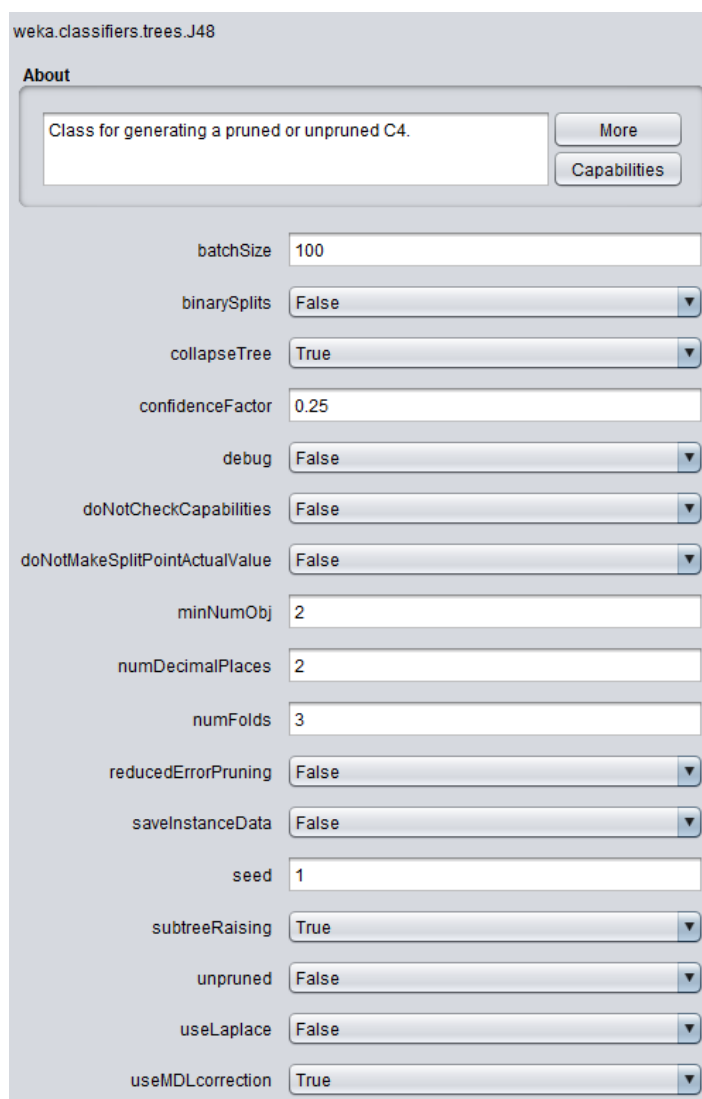
## Minería de datos (Estudio del modelo)

Para determinar si un modelo es bueno o no, es importante algunos datos como el Correctly classified instance (instancias clasificadas correctamente), Incorrect classified instance (instancias clasificadas incorrectamente) y el kappa statistics que dice que tan bien va a funcionar el modelo, este número va a estar siempre entre -1 y 1, se analiza interpretando que cuanto más cerca del cero esté peor es el modelo y cuanto más cerca del uno es mejor y si es 1 está describiendo con la máxima precisión pero no está infiriendo aunque en minería de datos si describe perfectamente no es nada bueno.

Utilizando árbol de inducción J48

### Caso 1

En la primera ejecución del clasificador J48 se hizo una configuración de la siguiente manera.



The image shows the configuration window for the J48 classifier in Weka. The window title is "weka.classifiers.trees.J48". It has an "About" tab selected, which contains a text box with "Class for generating a pruned or unpruned C4." and two buttons: "More" and "Capabilities". Below this, there are various configuration options:

- batchSize: 100
- binarySplits: False
- collapseTree: True
- confidenceFactor: 0.25
- debug: False
- doNotCheckCapabilities: False
- doNotMakeSplitPointActualValue: False
- minNumObj: 2
- numDecimalPlaces: 2
- numFolds: 3
- reducedErrorPruning: False
- saveInstanceData: False
- seed: 1
- subtreeRaising: True
- unpruned: False
- useLaplace: False
- useMDLcorrection: True

### Imagen 5

El resultado que se ha logrado con esta configuración es que el árbol que crea, tiene demasiadas hojas y ramas, lo cual no es algo beneficioso para el modelo que se quiere crear. El J48 que se ha configurado, el “minNumObj” en 2, con esto se está indicando que no haga ninguna subrama que tenga menos de 2 objetos y el confidenceFactor en 0.25 que es el nivel de confianza.

Este árbol generado tiene 5.047 niveles y el tamaño es de 10.093, como se puede ver en la imagen 6. El gran tamaño de este árbol indica que no ha podido inducir correctamente y que hay demasiado nivel de detalle

```
=== Run information ===

Scheme:      weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:    EURUSD (Jun Jul Ago) CDM (listando de a 15)-weka.filters.unsupervised.attribute.Remove-R1,15
Instances:   92253
Attributes:  14
              Col2
              Col3
              Col4
              Col5
              Col6
              Col7
              Col8
              Col9
              Col10
              Col11
              Col12
              Col13
              Col14
              Col16
Test mode:   10-fold cross-validation

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

Number of Leaves :    5047

Size of the tree :    10093

Time taken to build model: 56.55 seconds
```

### Imagen 6.

En la siguiente imagen podemos ver el tamaño del árbol desde una perspectiva gráfica.



## Caso 2

En el siguiente paso se trata de mejorar el árbol cambiando la configuración, más específicamente aumentando el “minNumObj” a 500 (esto indica que el árbol no va a tener ninguna subrama con menos de 500 objetos), el confidenceFactor seguirá en 0.25, y ejecutarlo con el “Percentage Split” en 66% para que la ejecución sea un poco más rápida y no tan pesada.

```
=== Run information ===

Scheme:      weka.classifiers.trees.J48 -C 0.25 -M 500
Relation:    EURUSD (Jun Jul Ago) CDM (listando de a 15)-weka.filters.unsupervised.attribute.Remove-R1,15
Instances:   92253
Attributes:  14
             Col2
             Col3
             Col4
             Col5
             Col6
             Col7
             Col8
             Col9
             Col10
             Col11
             Col12
             Col13
             Col14
             Col16

Test mode:   split 66.0% train, remainder test
```

Imagen 8

```
=== Summary ===

Correctly Classified Instances  23103      73.6562 %
Incorrectly Classified Instances  8263      26.3438 %
Kappa statistic                0.5514

Mean absolute error            0.258
Root mean squared error        0.3618
Relative absolute error         61.6066 %
Root relative squared error     79.0218 %
Total Number of Instances      31366

=== Detailed Accuracy By Class ===
```

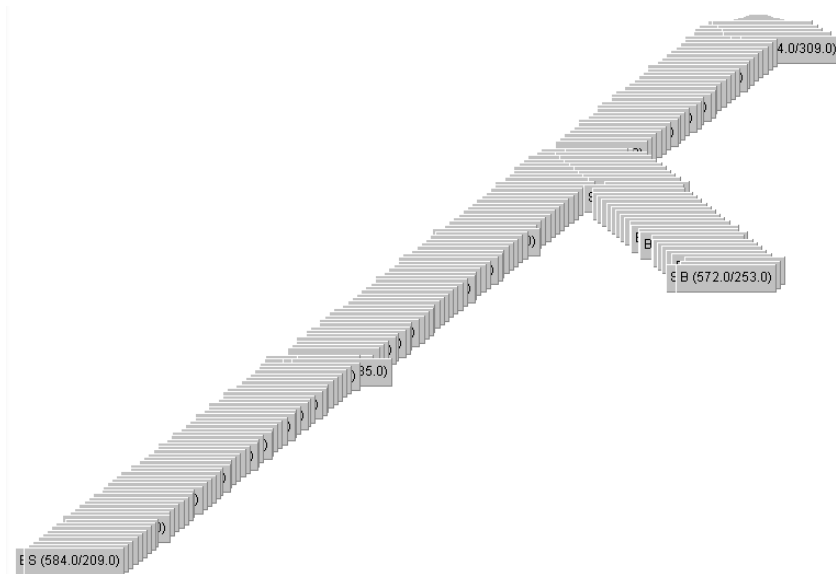
	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0,904	0,239	0,725	0,904	0,805	0,654	0,895	0,807	S	
0,000	0,000	?	0,000	?	?	0,585	0,220	I	
0,880	0,209	0,748	0,880	0,809	0,661	0,897	0,819	B	
Weighted Avg.	0,737	0,185	?	0,737	?	?	0,842	0,710	

```
=== Confusion Matrix ===

  a    b    c  <-- classified as
11665  0 1237 |   a = S
 2865  0 2607 |   b = I
 1554  0 11438 |   c = B
```

Imagen 9

Para esta configuración se logró tener una correcta clasificación en un 73,6% y una incorrecta clasificación en un 26,3%, también se puede ver que tiene 155 niveles y un tamaño de 309. También se destaca el kappa statistics en 0,55.



*Imagen 10*

Como se puede ver en la Imagen 10, el árbol es mucho más disminuido de acuerdo a estas modificaciones en la configuración. Este árbol contiene 155 niveles y un tamaño de 309. Al tener un árbol más reducido se puede decir que esto será más favorable para el modelo.

### **Caso 3**

Se sigue mejorando la configuración para que el modelo mejore, esta vez se aumentó el “minNumObj” a 900 y se obtuvo una mejora en el árbol pero disminuyó el porcentajes de aciertos.

En la imagen 11, se puede tomar como algo positivo de mejora la reducción de los niveles del árbol, pasó de ser de 155 a 112 y una reducción del tamaño de 309 a 223.

Lo negativo es que la disminución de la correcta clasificación, pasó de ser de un 63,7% a 59,7% y además aumentó la incorrecta clasificación de 36,3% a 40,2%. Esto último indica que no se ha podido mejorar el árbol de una manera esperada. También un kappa statistics de 0,31 que confirma que el modelo ha empeorado.

```
Number of Leaves : 112
Size of the tree : 223
```

Time taken to build model: 34.01 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.05 seconds

=== Summary ===

```
Correctly Classified Instances 18735 59.7303 %
Incorrectly Classified Instances 12631 40.2697 %
Kappa statistic 0.3141
Mean absolute error 0.3611
Root mean squared error 0.4268
Relative absolute error 86.2405 %
Root relative squared error 93.2102 %
Total Number of Instances 31366
```

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,696	0,317	0,606	0,696	0,648	0,374	0,734	0,624	S
	0,000	0,000	?	0,000	?	?	0,567	0,211	I
	0,751	0,369	0,590	0,751	0,661	0,376	0,738	0,646	B
Weighted Avg.	0,597	0,283	?	0,597	?	?	0,707	0,561	

=== Confusion Matrix ===

```
a b c <-- classified as
8983 0 3919 | a = S
2608 0 2864 | b = I
3240 0 9752 | c = B
```

Imagen 11

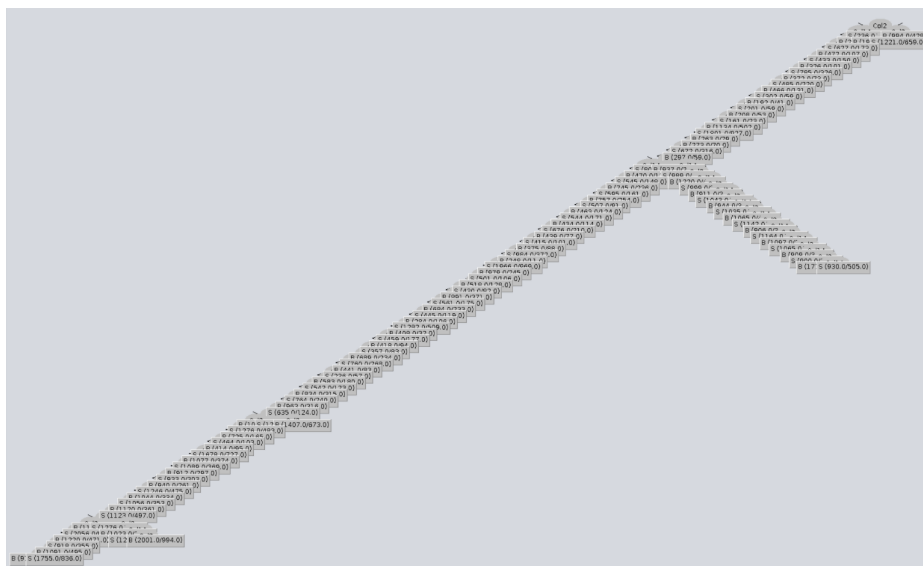


Imagen 12

## Caso 4

Para seguir mejorando el árbol de decisión conviene aumentar nuevamente el “minNumObj” a 1000 y cambiar el “batch-size” (cuantos datos toma a la vez) a 700.

Number of Leaves : 108  
Size of the tree : 215

Time taken to build model: 32.84 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.05 seconds

=== Summary ===

Correctly Classified Instances	18399	58.6591 %
Incorrectly Classified Instances	12967	41.3409 %
Kappa statistic	0.2959	
Mean absolute error	0.3675	
Root mean squared error	0.43	
Relative absolute error	87.7562 %	
Root relative squared error	93.9292 %	
Total Number of Instances	31366	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,692	0,336	0,590	0,692	0,637	0,350	0,719	0,611	S
	0,000	0,000	?	0,000	?	?	0,570	0,211	I
	0,729	0,368	0,583	0,729	0,648	0,356	0,725	0,632	B
Weighted Avg.	0,587	0,291	?	0,587	?	?	0,695	0,550	

=== Confusion Matrix ===

a	b	c	-- classified as
8925	0	3977	a = S
2684	0	2788	b = I
3518	0	9474	c = B

Imagen 13

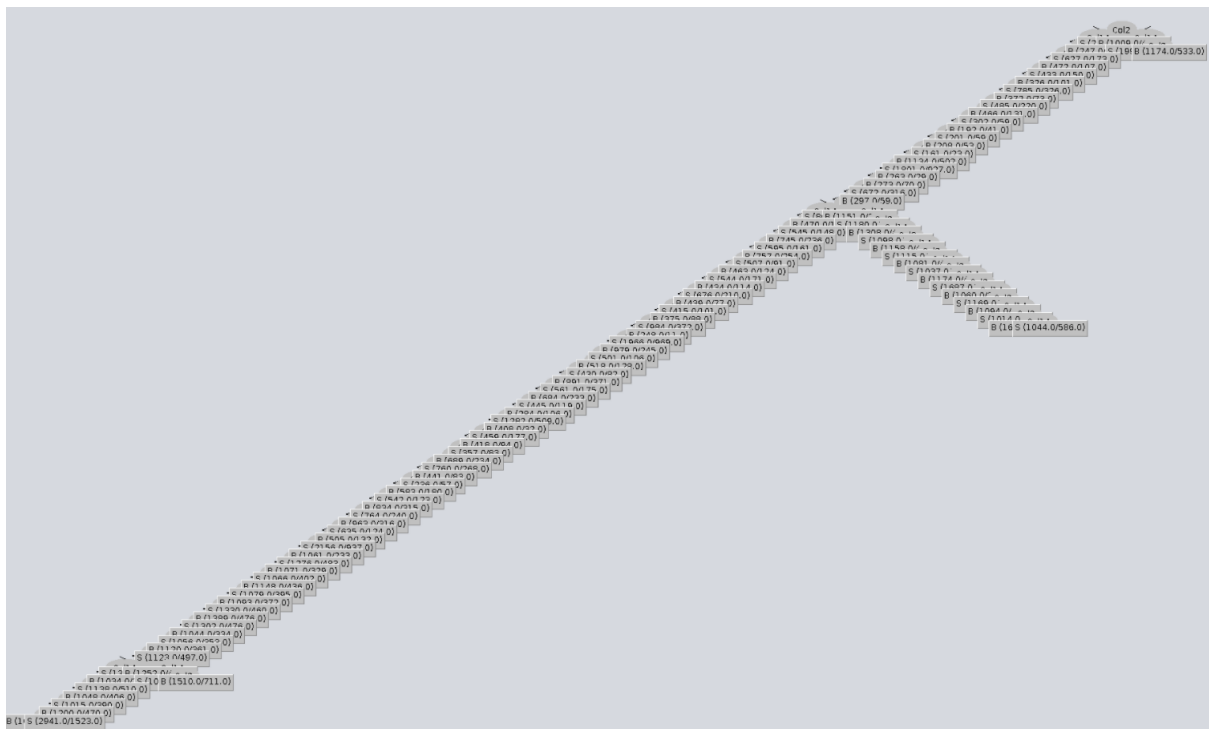


Imagen 14

Si bien como se ve en la imagen 13 no hay mejoras en la clasificación con respecto al anterior, la correcta clasificación ha bajado de 59,7% a 58,6 y por lo tanto desfavorablemente aumentó la incorrecta clasificación de 40,2% a 41,3%. También un kappa statistics en 0,29.

En conclusión, se podría decir que el mejor modelo que se ha logrado con este clasificador es en el **caso 2** , donde se obtiene el “minNumObj” en 500 y el confidenceFactor en 0.25. Se tiene una correcta clasificación de instancia en 63,7 %, incorrecta clasificación de instancia en 36,3% y el dato más importante el kappa statistics en 0,38.

Con redes neuronales: MultilayerPerceptron

## Caso 1

### Configuración de algunas opciones:

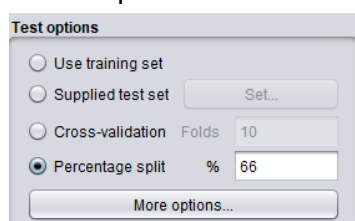
GUI: cómo false, ya que llevaría más tiempo debido a que esta opción muestra con una interfaz gráfica la creación del modelo.

autoBuild: En true, muestra cómo se arma la red neuronal.

batchSize: se ingresa 100, como tamaño del lote a procesar.

Con el Percentage split en 66%

Se define el porcentaje con el que se construirá el modelo y la parte restante con la que hará las pruebas.



*Imagen 16*

Al igual que en el árbol J48 se creará un modelo para predecir la columna 16 (Col16) del DataSet\_1 en base a las demás columnas, pero ahora se utilizará el método MultilayerPerceptron basado en redes neuronales. Las redes neuronales son mucho más favorables para este caso que el árbol de inducción J45, ya que se llevan mucho mejor con datos numéricos, por lo tanto, este método sería más favorable que el árbol de inducción.

Se configura el MultilayerPerceptron por defecto tal cual lo presenta weka, de la siguiente manera:



weka.classifiers.functions.MultilayerPerceptron

**About**

A classifier that uses backpropagation to learn a multi-layer perceptron to classify instances.

[More](#)  
[Capabilities](#)

GUI	False
autoBuild	True
batchSize	100
debug	False
decay	False
doNotCheckCapabilities	False
hiddenLayers	a
learningRate	0.3
momentum	0.2
nominalToBinaryFilter	True
normalizeAttributes	True
normalizeNumericClass	True
numDecimalPlaces	2
reset	True
resume	False
seed	0
trainingTime	500
validationSetSize	0
validationThreshold	20

*Imagen 15*

```

=== Run information ===

Scheme:      weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
Relation:    EURUSD (Jun Jul Ago) CDM (listando de a 15)-weka.filters.unsupervised.attribute.Remove-R1,15
Instances:    92253
Attributes:   14
              Col2
              Col3
              Col4
              Col5
              Col6
              Col7
              Col8
              Col9
              Col10
              Col11
              Col12
              Col13
              Col14
              Col16

Test mode:    split 66.0% train, remainder test

=== Evaluation on test split ===

Time taken to test model on test split: 0.11 seconds

=== Summary ===

Correctly Classified Instances      24320      77.5362 %
Incorrectly Classified Instances    7046      22.4638 %
Kappa statistic                    0.6387
Mean absolute error                 0.215
Root mean squared error             0.3313
Relative absolute error             51.3312 %
Root relative squared error         72.3572 %
Total Number of Instances          31366

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
          0,855    0,146    0,804     0,855    0,829      0,704    0,925     0,897     S
          0,400    0,091    0,483     0,400    0,437      0,334    0,791     0,415     I
          0,854    0,109    0,847     0,854    0,850      0,744    0,933     0,915     B
Weighted Avg.   0,775    0,121    0,766     0,775    0,769      0,656    0,905     0,820

=== Confusion Matrix ===
  a    b    c  <-- classified as
11037 1269 596 | a = S
1875 2188 1409 | b = I
822 1075 11095 | c = B

```

Imagen 17

Luego de correr el tenemos un resultado el MultilayerPerceptron se obtuvo los siguientes resultados:

El kappa statistic es 0,6387, indica que es bastante bueno, por lo tanto el 63% de las veces el modelo va a funcionar correctamente

También indica que clasificó correctamente 77,5% y las que clasificó incorrectamente fue 22,4%.

En la matriz de confusión en la imagen 18, se puede notar lo siguiente:

Se marca los aciertos que tuvo (marcados con verde), en la primera columna (a) cuando era S (suba), lo clasificó como S 1.1037 veces, en la segunda columna (b) cuando era I (igual), lo clasificó como I 2.188 veces y la tercera columna (c) era B (baja ), lo clasificó como B 11.095 veces.

Luego tenemos los desaciertos que están marcados con rojo:

En la primera columna (a), cuando era I, lo clasificó como S 1.875 y cuando era B, lo clasificó como S 8.22 veces.

En la segunda columna (b), cuando era S, lo clasificó como I 1.269 veces y cuando era B, lo clasificó como I 1.075 veces.

En la tercera columna (c), cuando era S, lo clasificó como B 5.96 veces y cuando era I, lo clasificó como B 1.409 veces.

=== Confusion Matrix ===

	a	b	c	<-- classified as
a	11037	1269	596	a = S
b	1875	2188	1409	b = I
c	822	1075	11095	c = B

Imagen 18

Teniendo en cuenta que cuando era S lo clasificó como B 596 veces y cuando era B lo clasificó como S 822, esto da la pauta de que este modelo se equivoca más cuando el precio sube que cuando el precio baja.

## Caso 2

En este caso se cambiará la configuración del MultilayerPerceptron para tratar de mejorar el modelo del caso anterior. Para esto se aumenta el **batchSize** en 200, esto permitirá tener mayor cantidad de instancias para procesar y el **hiddenLayers** en 8 (cantidad de nodos)

GUI	False
autoBuild	True
batchSize	200
debug	False
decay	False
doNotCheckCapabilities	False
hiddenLayers	8
learningRate	0.3
momentum	0.2
nominalToBinaryFilter	True
normalizeAttributes	True
normalizeNumericClass	True
numDecimalPlaces	2
reset	True
resume	False
seed	0
trainingTime	500
validationSetSize	0
validationThreshold	20

Imagen 19

```

=== Evaluation on test split ===

Time taken to test model on test split: 0.18 seconds

=== Summary ===

Correctly Classified Instances      31365          99.9968 %
Incorrectly Classified Instances      1          0.0032 %
Kappa statistic                    0.9999
Mean absolute error                 0.0036
Root mean squared error             0.015
Relative absolute error              0.8599 %
Root relative squared error          3.2751 %
Total Number of Instances           31366

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
          1,000    0,000    1,000     1,000    1,000     1,000    1,000    1,000    S
          1,000    0,000    1,000     1,000    1,000     1,000    1,000    1,000    I
          1,000    0,000    1,000     1,000    1,000     1,000    1,000    1,000    B
Weighted Avg.    1,000    0,000    1,000     1,000    1,000     1,000    1,000    1,000

=== Confusion Matrix ===

  a    b    c  <-- classified as
12902   0    0 |    a = S
  0 5471    1 |    b = I
  0    0 12992 |    c = B

```

## Imagen 20

En la Imagen 20 se puede ver el resultado de la red neuronal, indica que tiene un 99,99% de instancias clasificadas correctamente y 0,003 instancias clasificadas incorrectamente, también el kappa statistic indica 0,99. A simple vista esto parece ser que la red neuronal es casi perfecta y que casi no se confunde.

Se puede percibir en la matriz de confusión que sólo se equivocó una sola vez, cuando era I (igual) dijo que era B (baja). Estos resultados son un poco mentirosos, como se sabe cuando el kappa statistics es 1 o demasiado cerca al uno indica que está describiendo con la máxima precisión pero no está infiriendo, por otro lado la correcta clasificación de instancias es casi el 100% esto no es bueno, a esto se lo conoce como overfitting, esto es que el modelo se aprende el dataset casi de memoria.

Como se vio, estos resultados no son demasiado favorables para nuestra red neuronal, por lo tanto se puede decir que este caso (caso 2) no es mejor que el caso anterior (caso 1). Debido a esto seguiremos mejorando nuestro modelo de red neuronal.

## Caso 3

Para este caso se vuelve el **hiddenLayers** en "a" como en el Caso 1 y se mantiene el **batchSize** en 200.

=== Evaluation on test split ===

Time taken to test model on test split: 0.33 seconds

=== Summary ===

Correctly Classified Instances	28084	89.5364 %
Incorrectly Classified Instances	3282	10.4636 %
Kappa statistic	0.8335	
Mean absolute error	0.0772	
Root mean squared error	0.2115	
Relative absolute error	18.4393 %	
Root relative squared error	46.1988 %	
Total Number of Instances	31366	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	1,000	0,069	0,910	1,000	0,953	0,920	1,000	1,000	S
	0,695	0,062	0,702	0,695	0,699	0,635	0,953	0,671	I
	0,876	0,021	0,967	0,876	0,919	0,870	0,994	0,991	B
Weighted Avg.	0,895	0,048	0,897	0,895	0,894	0,849	0,989	0,939	

=== Confusion Matrix ===

a	b	c	<-- classified as
12900	2	0	a = S
1279	3804	389	b = I
0	1612	11380	c = B

## Imagen 21

Como se ve en la imagen 21, se obtiene una correcta clasificación de instancias de 89,5% y correcta clasificación de instancias de 10,5%, también un kappa statistics de 0,83. Esto da la pauta de que se tiene una mejora con respecto al mejor modelo, caso 1, de estos 3 casos que hemos creado.

En el caso 1 se tiene una correcta clasificación de instancias del 77,5% y una incorrecta clasificación del 22,5% además de un capa de 0,63%.

Claramente se nota que hay una mejora en este caso con respecto al caso 1.

## Conclusión

Comparando los modelos desarrollados hasta el momento, por un lado los resultados obtenidos a partir del mejor modelo utilizando el árbol de inducción J48 y los resultados obtenidos a partir del mejor modelo utilizando la red neuronal MultilayerPerceptron.

Para empezar se puede seleccionar el mejor modelo del J48 obtenido a partir del análisis de sus resultados, que es el caso 2 donde se tenía una correcta clasificación de instancia en 63,7%, incorrecta clasificación de instancia en 36,3% y el dato más importante el kappa statistics en 0,38.

Ahora el mejor modelo de red neuronal, que es el caso 3, donde se obtuvo una correcta clasificación de instancias de 89,5% y incorrecta clasificación de instancias de 10,5%, también un kappa statistics de 0,83.

Como conclusión se puede decir que se ha obtenido mejores resultados utilizando la red neuronal ya que existe una diferencia de más del doble en el kappa statistics (en J48 0,38 y en la red neuronal 0,83), también es mejor la matriz y el porcentaje de clasificación de instancias correcta, por lo tanto esto confirma que la red neuronal se equivocó menos veces que el árbol de inducción. Por todos estos motivos en la red neuronal se logró más del doble de mejora con respecto al modelo realizado en el árbol J48.

## Estudio de datos sobre DataSet\_2

En esta primera sesión se analiza con weka los datos del DataSet\_2, proveniente de ejecutar el algoritmo 2.

A continuación se carga el archivo DataSet\_2.csv en weka y se obtiene los siguientes resultados.

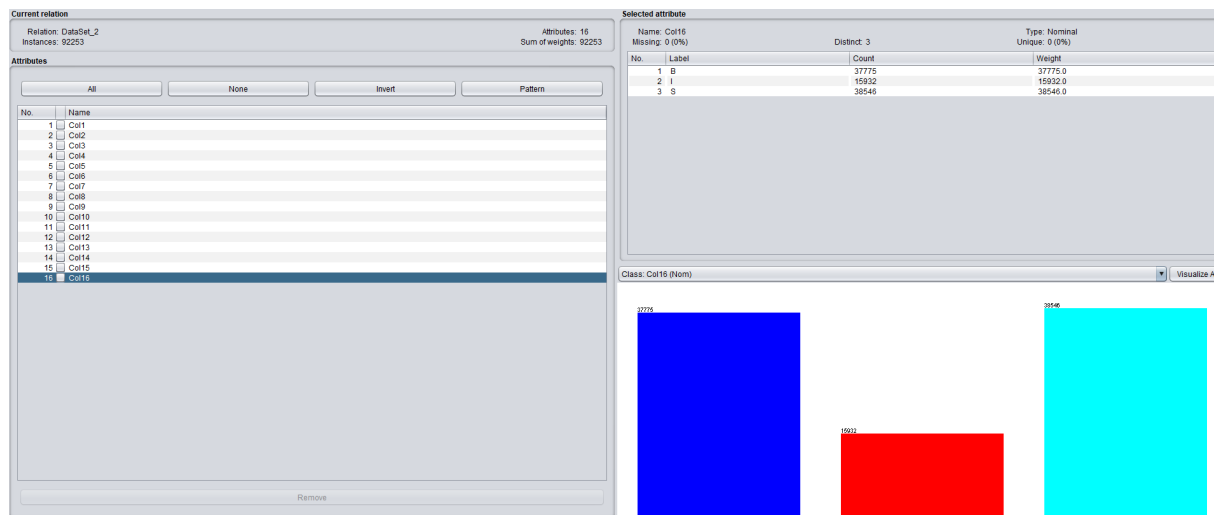


Imagen 22

La información que muestra Weka en la imagen 22, luego de cargar el archivo, es que existen 92.253 (Instances) registros con 16 columnas (Attributes).

Tal como se ve en la imagen 22, seleccionando la columna nominal 16, se puede ver los tres valores que contiene la columna que son "S", "I" y "B" con la cantidad de datos que van a predecir cada uno. Para S encontré 38.546, para I 15.932 y para B 37.775.

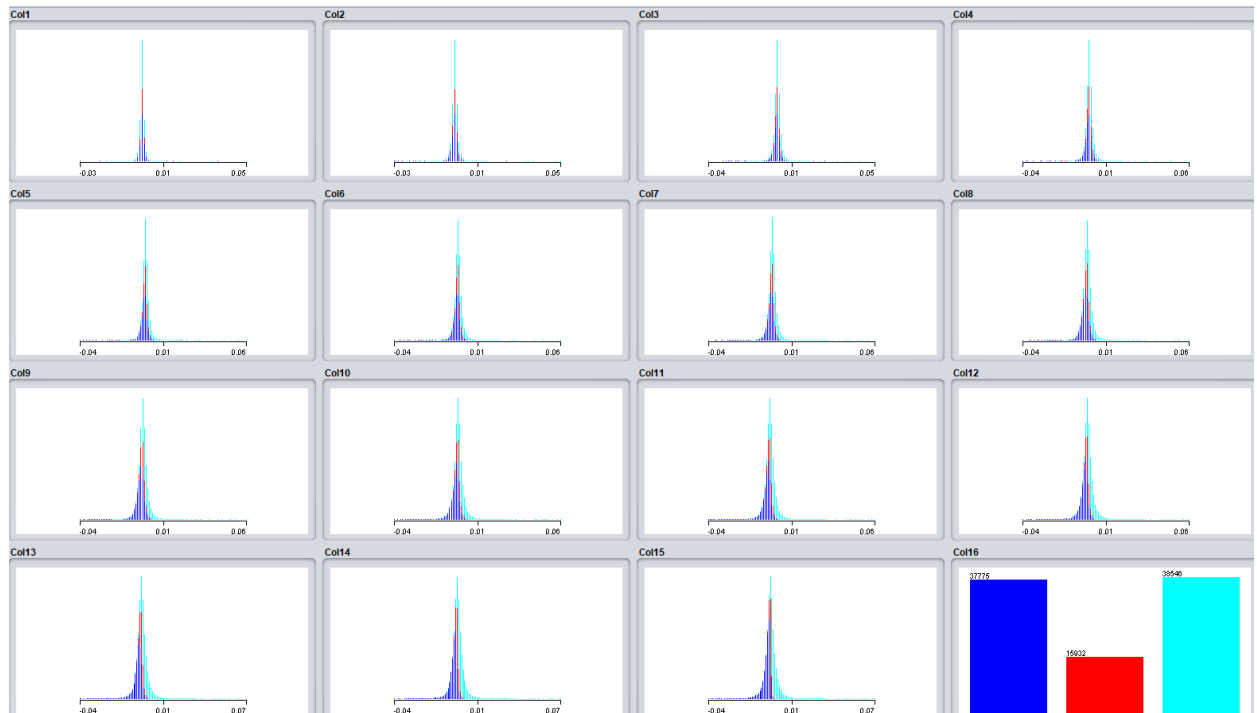


Imagen 23

En la Imagen 23 se muestra la distribución de los datos desde una vista gráfica. A simple vista se ve que desde la columna 1 a la 15 (columnas no nominal), los datos tienen un comportamiento similar. Se detectó que los gráficos tienen una forma monomodal, este es el nombre que se le da a estos tipos de gráficos donde los datos están agrupados de una forma más compacta, al principio hay un ascenso de los datos hasta llegar a la cima (tope) para luego bajar. Tener este tipo de gráfico es favorable para desarrollar el modelo ya que el precio no está tan disperso.

#### Utilizando herramienta de selección de atributos

La herramienta de selección de atributos de weka, Raker. Como ya se vio anteriormente en el primer dataset, esta herramienta permite saber los atributos (columnas) más relevantes o que más aportan a la columna nominal, que en este caso es la columna 16 (Col16).

Luego de ejecutar el Ranker en weka se obtienen los siguientes resultados que podemos ver en la imagen 24.

El Ranker arroja el ranking de las columnas más importantes:

15,14,13,12,11,10,9,8,7,6,5,4,3,2,1

Esto indica que la columna 15 es la más influyente y la columna 1 la menos influyente.



=== Run information ===

Evaluator: weka.attributeSelection.InfoGainAttributeEval  
Search: weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1  
Relation: DataSet\_2  
Instances: 92253  
Attributes: 16  
Col1  
Col2  
Col3  
Col4  
Col5  
Col6  
Col7  
Col8  
Col9  
Col10  
Col11  
Col12  
Col13  
Col14  
Col15  
Col16  
Evaluation mode: evaluate on all training data

=== Attribute Selection on all input data ===

Search Method:  
Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 16 Col16):  
Information Gain Ranking Filter

Ranked attributes:

0.92782	15	Col15
0.7166	14	Col14
0.59302	13	Col13
0.50098	12	Col12
0.42499	11	Col11
0.35775	10	Col10
0.30126	9	Col9
0.25003	8	Col8
0.2042	7	Col7
0.16199	6	Col6
0.1213	5	Col5
0.08402	4	Col4
0.0503	3	Col3
0.02165	2	Col2
0.0073	1	Col1

Selected attributes: 15,14,13,12,11,10,9,8,7,6,5,4,3,2,1 : 15

*Imagen 24*

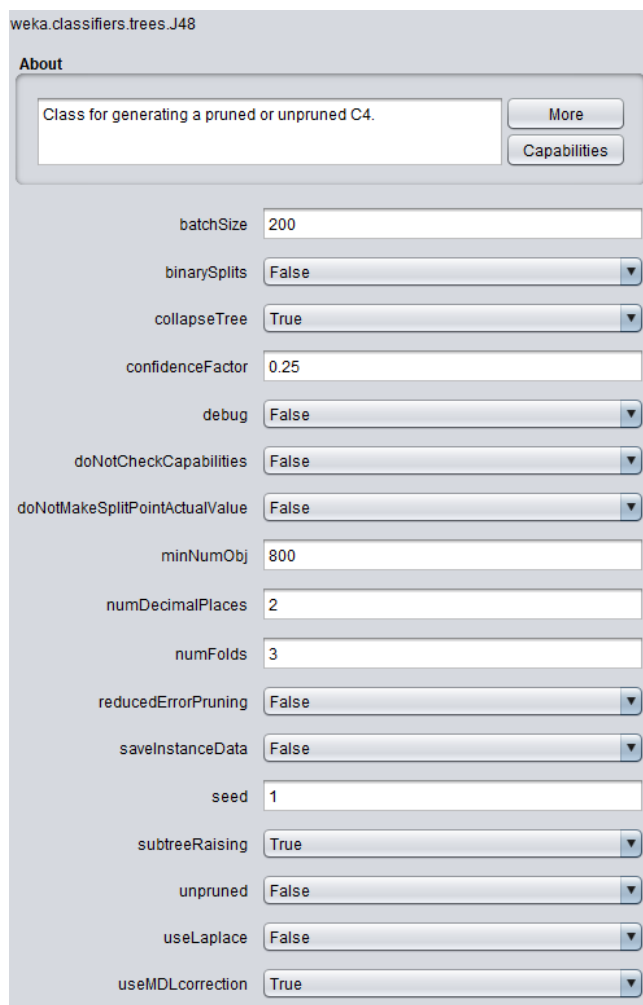
## Minería de datos (Estudio del modelo)

Para determinar si un modelo es bueno o no, es importante algunos datos que arroja weka, como el “Correctly classified instance” (instancias clasificadas correctamente), “Incorrect classified instance” (instancias clasificadas incorrectamente) y el “kappa statistics”, dice que tan bien va a funcionar el modelo, este número va a estar siempre entre -1 y 1, se analiza interpretando que cuanto más cerca del cero esté peor es el modelo y cuanto más cerca del uno es mejor y si es 1 está describiendo con la máxima precisión pero no está infiriendo aunque en minería de datos si describe perfectamente no es nada bueno.

Utilizando árbol de inducción J48

### Caso 1

Para este primer caso, más allá de los parámetros que viene por defecto, se comienza con configurar el batchSize en 200 y el minNumObj en 800.



The image shows the 'weka.classifiers.trees.J48' settings window. At the top, there is an 'About' section with a text box containing 'Class for generating a pruned or unpruned C4.' and two buttons: 'More' and 'Capabilities'. Below this, a list of parameters is shown with their current values: batchSize (200), binarySplits (False), collapseTree (True), confidenceFactor (0.25), debug (False), doNotCheckCapabilities (False), doNotMakeSplitPointActualValue (False), minNumObj (800), numDecimalPlaces (2), numFolds (3), reducedErrorPruning (False), saveInstanceData (False), seed (1), subtreeRaising (True), unpruned (False), useLaplace (False), and useMDLcorrection (True). Each parameter has a corresponding input field or dropdown menu.

*Imagen 25*

```

=== Evaluation on test split ===

Time taken to test model on test split: 0.02 seconds

=== Summary ===
Correctly Classified Instances      30635      97.6695 %
Incorrectly Classified Instances    731        2.3305 %
Kappa statistic                    0.963
Mean absolute error                 0.0277
Root mean squared error             0.1186
Relative absolute error              6.6124 %
Root relative squared error          25.9051 %
Total Number of Instances          31366

=== Detailed Accuracy By Class ===

```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,986	0,013	0,981	0,986	0,984	0,972	0,998	0,994	B
	0,941	0,012	0,946	0,941	0,943	0,931	0,991	0,960	I
	0,983	0,011	0,985	0,983	0,984	0,973	0,998	0,994	S
Weighted Avg.	0,977	0,012	0,977	0,977	0,977	0,965	0,997	0,988	

```

=== Confusion Matrix ===

```

a	b	c	<-- classified as
12675	142	42	a = B
172	5181	152	b = I
67	156	12779	c = S

*Imagen 26*

Como resultado, se puede ver en la imagen 26, se ha desarrollado un muy buen modelo, teniendo en cuenta las instancias clasificadas correctamente en 97,6%, las instancias clasificadas incorrectamente 2,3% y el kappa statistics en 0,963. Esto nos dice que el modelo se ha equivocado solo el 2,3% de las veces.

## Caso 2

En este segundo caso se deja como esta el batchSize en 200 pero se actualiza el minNumObj en 500.

weka.classifiers.trees.J48

**About**

Class for generating a pruned or unpruned C4. More  
Capabilities

batchSize 200

binarySplits False

collapseTree True

confidenceFactor 0.25

debug False

doNotCheckCapabilities False

doNotMakeSplitPointActualValue False

minNumObj 500

numDecimalPlaces 2

numFolds 3

reducedErrorPruning False

saveInstanceData False

seed 1

subtreeRaising True

unpruned False

useLaplace False

useMDLcorrection True

Imagen 27

=== Evaluation on test split ===

Time taken to test model on test split: 0.02 seconds

=== Summary ===

Correctly Classified Instances	30742	98.0106 %
Incorrectly Classified Instances	624	1.9894 %
Kappa statistic	0.9686	
Mean absolute error	0.0202	
Root mean squared error	0.0997	
Relative absolute error	4.8134 %	
Root relative squared error	21.7637 %	
Total Number of Instances	31366	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,978	0,001	0,998	0,978	0,988	0,980	0,999	0,999	B
	0,996	0,023	0,903	0,996	0,947	0,937	0,997	0,981	I
	0,975	0,000	0,999	0,975	0,987	0,978	0,999	0,998	S
Weighted Avg.	0,980	0,005	0,982	0,980	0,980	0,972	0,999	0,995	

=== Confusion Matrix ===

a	b	c	<-- classified as
12576	280	3	a = B
14	5485	6	b = I
10	311	12681	c = S

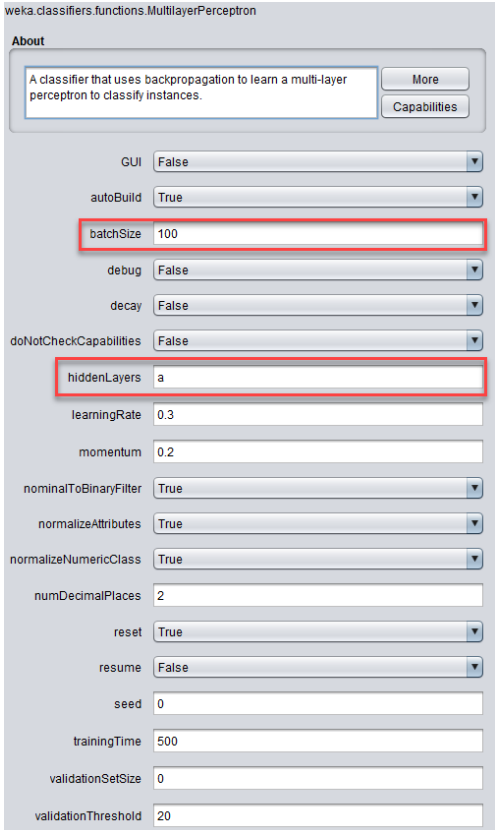
Imagen 28

Viendo la Imagen 28 se puede ver el resultado, se ha desarrollado un muy buen modelo, teniendo en cuenta las instancias clasificadas correctamente en 98,0%, las instancias clasificadas incorrectamente 1,9% y el kappa statistics en 0,968.

Con este resultado se puede ver una mejora con respecto al caso 1, donde se tiene mejores resultados en el kappa statistics y en instancias clasificadas correctamente, por lo tanto se establece este caso (caso 2) como el mejor modelo dentro del clasificador J48.

Con redes neuronales: MultilayerPerceptron

## Caso 1



**Resultado**

```

=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances  92253      100 %
Incorrectly Classified Instances  0         0 %
Kappa statistic                1
Mean absolute error            0.0007
Root mean squared error        0.0014
Relative absolute error        0.1702 %
Root relative squared error     0.2997 %
Total Number of Instances      92253

=== Detailed Accuracy By Class ===

```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	B
1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	I
1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	S
Weighted Avg.	1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000	

```

=== Confusion Matrix ===
      a   b   c  <-- classified as
37775  0   0   1   a = B
 0 15932  0   1   b = I
 0   0 38546  1   c = S

```

Imagen 29

Después de varias pruebas y cambios en los diferentes parámetros y de obtener siempre estos mismo resultados, como el kappa statistic en 1. Debido a esto, se llega a la conclusión de que el DataSet\_2 tiene datos demasiados simples por lo que la red neuronal no puede desarrollar un modelo confiable que pueda predecir. Cuando se dice que los datos son simples, implica que estos son perfectos dentro de un rango y no tiene datos tan dispersos como la del DataSet\_1, por lo que para las redes neuronales no es muy favorable, además para superar esto es necesario mayor cantidad de datos.

Para poder mejorar este dataset, se aplica un proceso de Normalización de datos. Según Santiago Morante (2018), para que funcionen mejor muchos algoritmos de Machine Learning usados en Data Science, hay que normalizar las variables de entrada al algoritmo. Normalizar significa, en este caso, comprimir o extender los valores de la variable para que estén en un rango definido.

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Imagen 30

De acuerdo a la fórmula de normalización, X es el dato que ingresa a la función comenzando desde el primer elemento del dataset hasta el último, Xmin es el menor de todos los números del dataset y Xmax es el mayor elemento de todo el dataset.

Para este algoritmo se ha desarrollado un módulo en python donde se utiliza el *GenerarDataSet*(3, 'DataSet\_3.csv', 'DataSet\_2 .csv'). Se le pasa como parámetro en datasetNum el número 3, el segundo parámetro (fileNameGenerated) es el nombre del archivo que se genera y el tercer parámetro (fileNameToProces) es el nombre del archivo en el cual se basará para generar el dataset. Este módulo carga el dataset *DataSet\_2* y a partir de esto se genera un archivo csv que en este trabajo se llama **DataSet\_3** ya que es generado y es la continuación del dataset *DataSet\_2*.

Para procesar este dataset *DataSet\_3* en la aplicación de weka se utiliza *MakeDensityBasedClusterer*.

	A	B	C	D	E	F	G
1	Col1,Col2,Col3,Col4,Col5,Col6,Col7,Col8,Col9,Col10,Col11,Col12,Col13,Col14,Col15,Class						
2	0.37,0.38,0.38,0.37,0.37,0.36,0.35,0.35,0.34,0.35,0.35,0.35,0.35,0.35,0.35,B						
3	0.38,0.38,0.37,0.37,0.36,0.35,0.35,0.34,0.35,0.35,0.35,0.35,0.35,0.35,0.36,B						
4	0.37,0.36,0.36,0.35,0.35,0.34,0.33,0.34,0.34,0.34,0.35,0.35,0.35,0.35,0.36,B						

Imagen 31.1 (extracto de archivo *DataSet\_3.csv*)

Al cargar el archivo *DataSet\_3*, weka da una perspectiva de la composición de los datos.

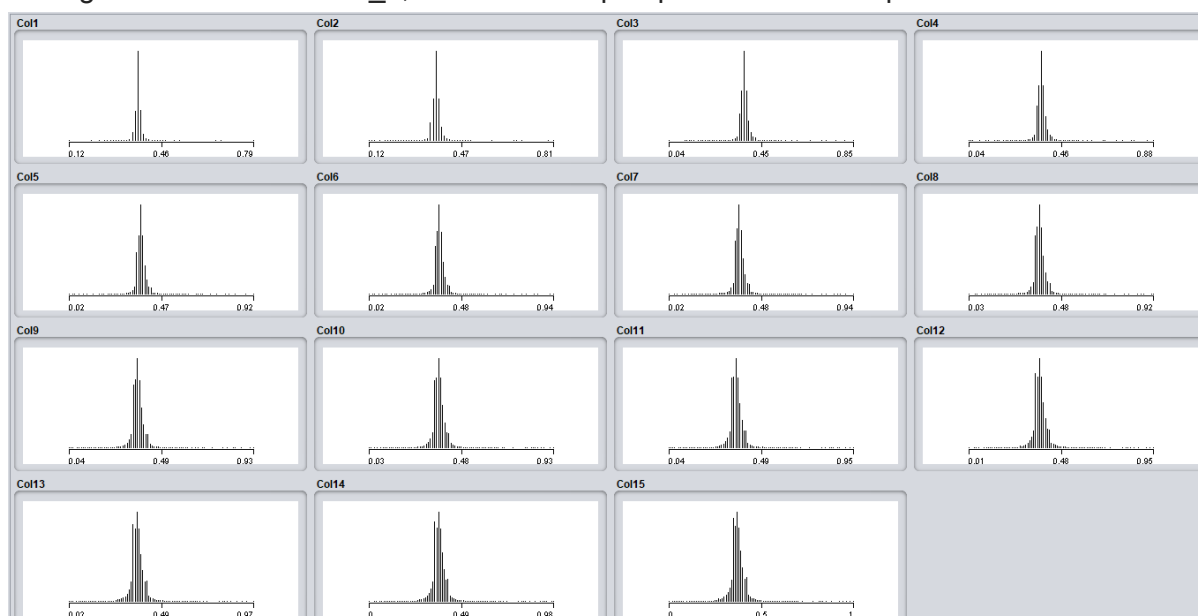


Imagen 31.2

Con el agrupador: MakeDensityBasedClusterer

Esta herramienta envuelve un algoritmo de agrupamiento para que devuelva una distribución de probabilidad y densidad. A cada conglomerado y atributo se le ajusta una distribución discreta o una distribución normal simétrica (de la cual la desviación estándar mínima es un parámetro).

### Caso 1

Se ejecuta el DataSet\_3 en weka con el MakeDensityBasedClusterer. Se Deja los parámetros seteados por defecto como se ve en la imagen 32. Es necesario aclarar que tiene como defecto seteado 2 clusters en

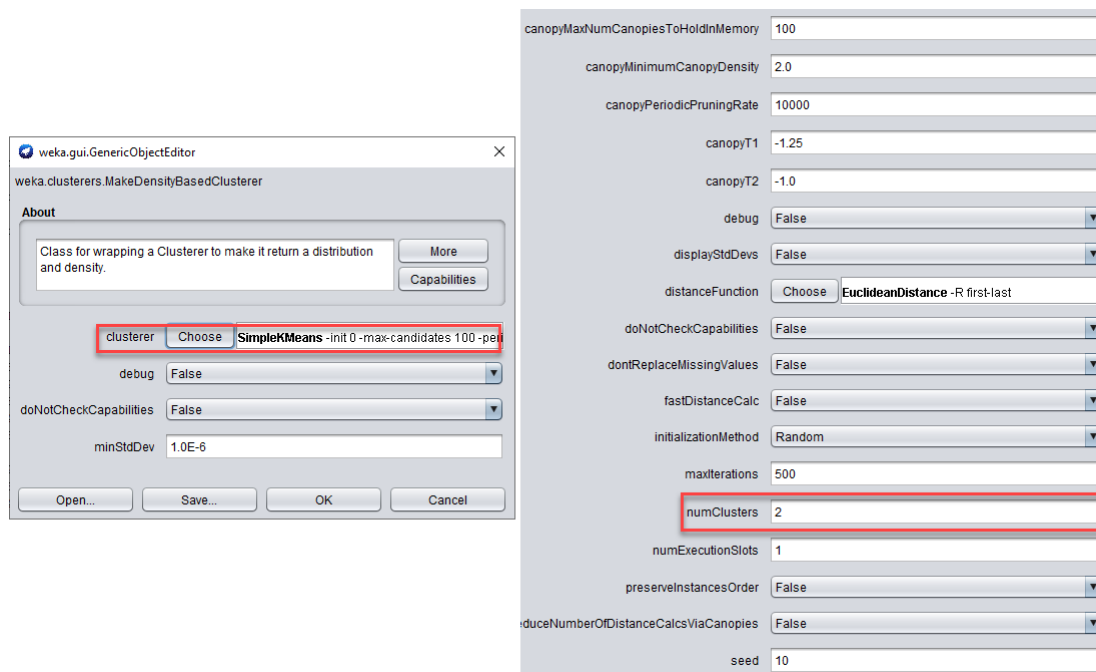


Imagen 32

#### Clustered Instances

0	10925 ( 35%)
1	20442 ( 65%)

Log likelihood: 37.56792

Imagen 33

Como resultado se obtiene lo que se ve en la imagen 33, lo que el metaclusterer ha podido generar con 2 clusters, 35% en el primero y 65% en el segundo y un Log likelihood de 37.56 que es una medida relativa que se utiliza para ajustar parámetros.

## Caso 2

En este caso se busca mejorar el Log likelihood aumentando la cantidad de cluster a 10.

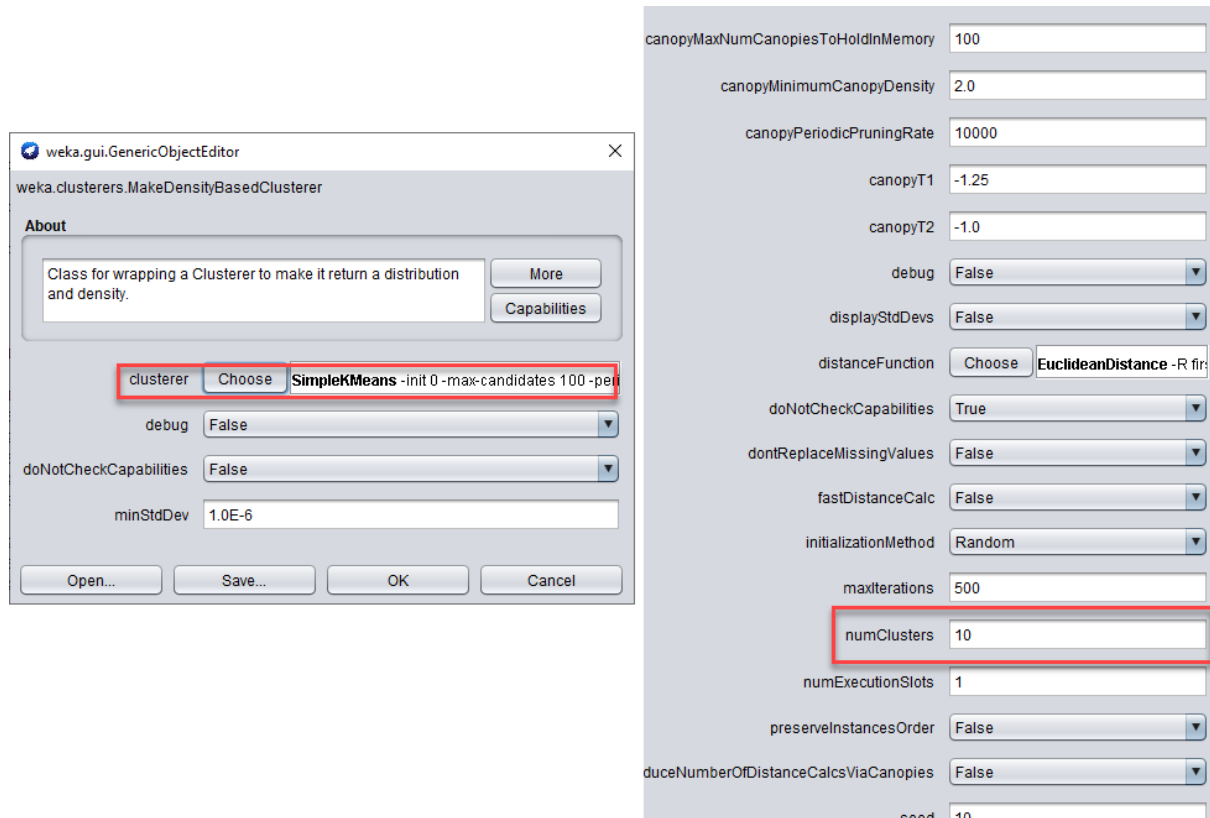


Imagen 34

Como resultado, imagen 35, se ha podido mejorar el log likelihood a 44.9, aumentando la cantidad de clusters a 10.

### Clustered Instances

0	8197 ( 26%)
1	3301 ( 11%)
2	2213 ( 7%)
3	5318 ( 17%)
4	4402 ( 14%)
5	5300 ( 17%)
6	42 ( 0%)
7	294 ( 1%)
8	1810 ( 6%)
9	490 ( 2%)

Log likelihood: 44.89992

Imagen 35

Conclusión



En esta conclusión se comparan los resultados obtenidos en el DataSet\_2 y el DataSet\_3.

Para el caso del DataSet\_2 como mejor resultado que se obtuvo fue con la ejecución del árbol J48 con buenos resultados (instancias clasificadas correctamente en 98,0%, las instancias clasificadas incorrectamente 1,9% y el kappa statistics en 0,968), se podría decir que es el mejor resultado que se obtuvo ya que con las redes neuronales no se ha podido obtener un buen resultado (kappa statistic igual a 1).

Para el caso del DataSet\_3 se utilizó el MakeDensityBasedClusterer el cual obtuvo mejores resultados en el caso 2, con 10 clusters y un log likelihood de 44.9.

## Comparación de los Datasets

En esta sesión de este trabajo se hará una comparación entre el DataSet\_1, DataSet\_2 y DataSet\_3.

El primer criterio que se utilizará para compararlos es a través de la composición de los datos que estos poseen. El DataSet\_1 tiene datos representados por gráficos multimodal, esto significa que los datos están más dispersos a diferencia de los dataset DataSet\_2 y DataSet\_3 que sus datos representados en gráficos tiene una forma monomodal, significa que estos datos están más compactos y más agrupados.

Para entender mejor el concepto de Bimodal/Multimodo y Monomodo, en la imagen 36 podemos ver la diferencia entre uno y otro.

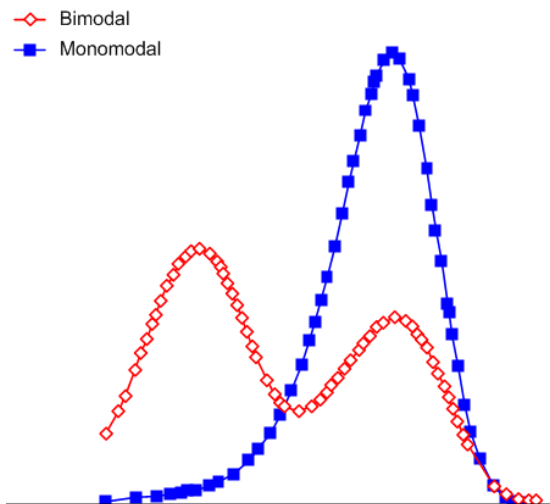


Imagen 36

La siguiente comparación tiene que ver con la curtosis. La curtosis indica que tan confiable es la moda con respecto al promedio. Existen 3 tipos de curtosis, Leptocúrtica donde los datos están más compactos, Mesocúrtica es el escenario ideal donde los datos están más estables y Platicúrtica indica dispersión de valores por lo tanto esto lleva a la ambigüedad que no es muy bueno para el modelo.

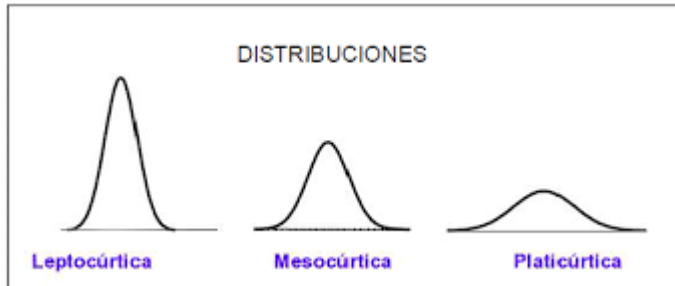


Imagen 37

Para el DataSet\_1 tiene una curtosis Mesocúrtica, esto es favorable incluso fue el dataset con el que menos problemas hubo en el proceso de minería de datos.

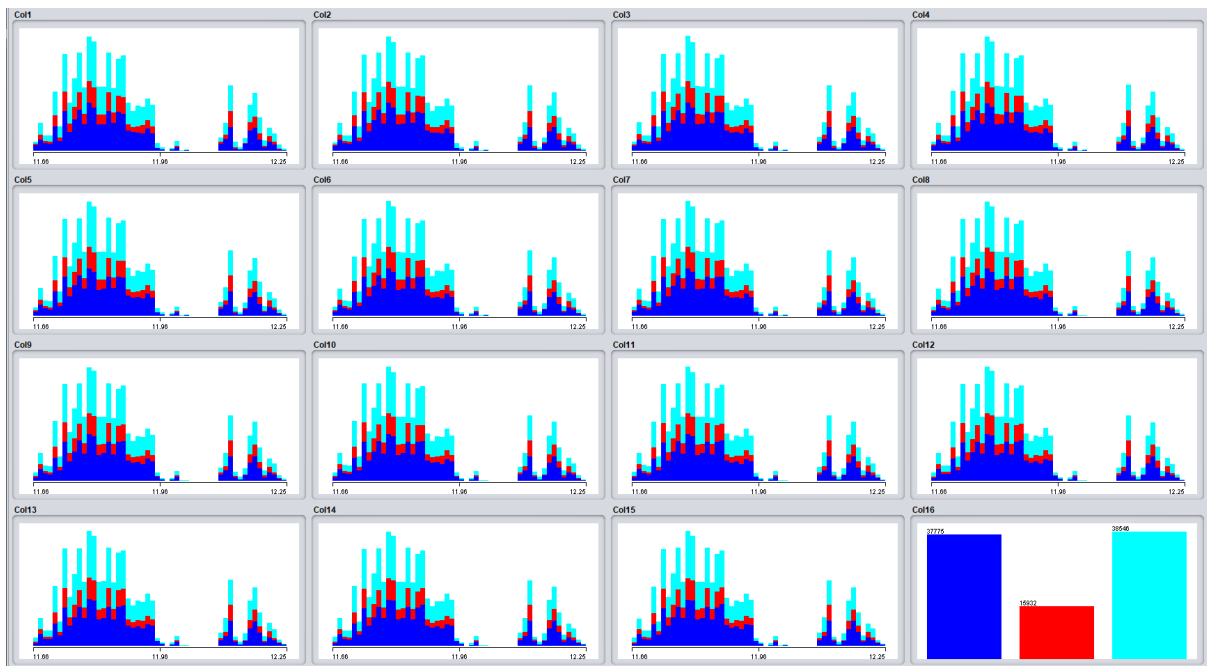


Imagen 37.1 (DataSet\_1)

En cambio para el DataSet\_2 tiene una curtosis leptocúrtica como se ve en la Imagen 37.2, por lo que se tuvo que aplicar el proceso de normalización de los datos.

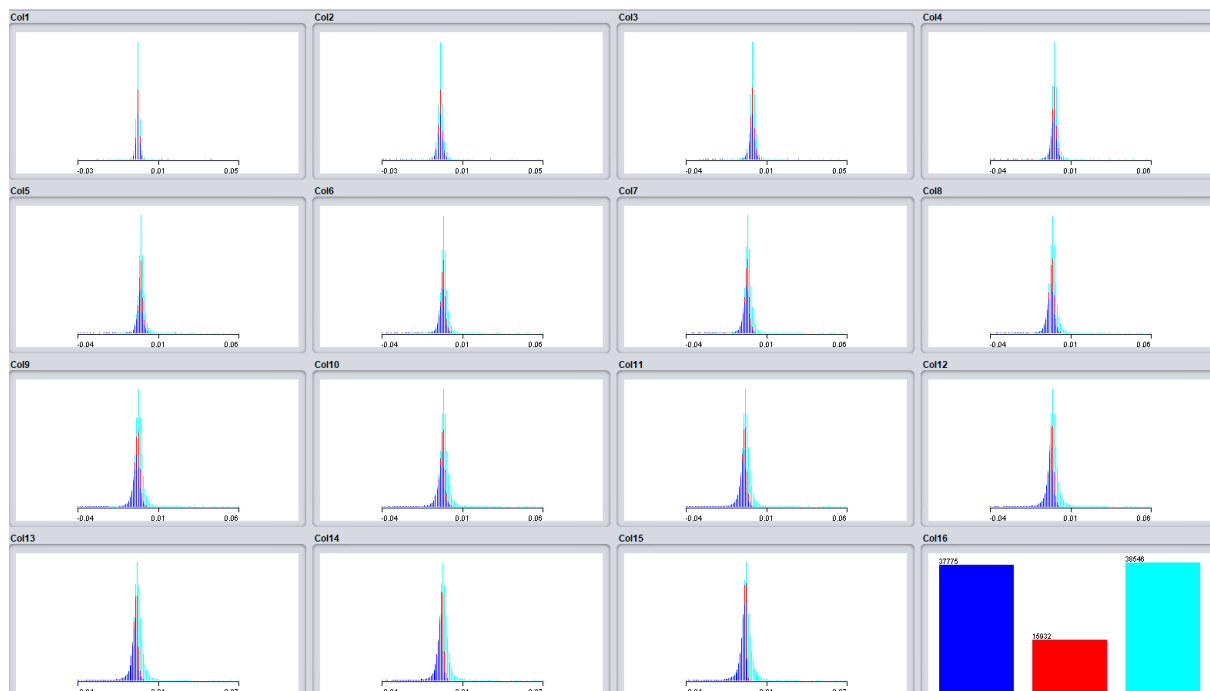


Imagen 37.2 (DataSet\_2)

Luego de aplicar el proceso de normalización de datos para el DataSet\_2 se obtuvo el siguiente resultado (Image 37.3) con una curtosis mesocúrtica, con lo cual favoreció en el proceso de crear el modelo.

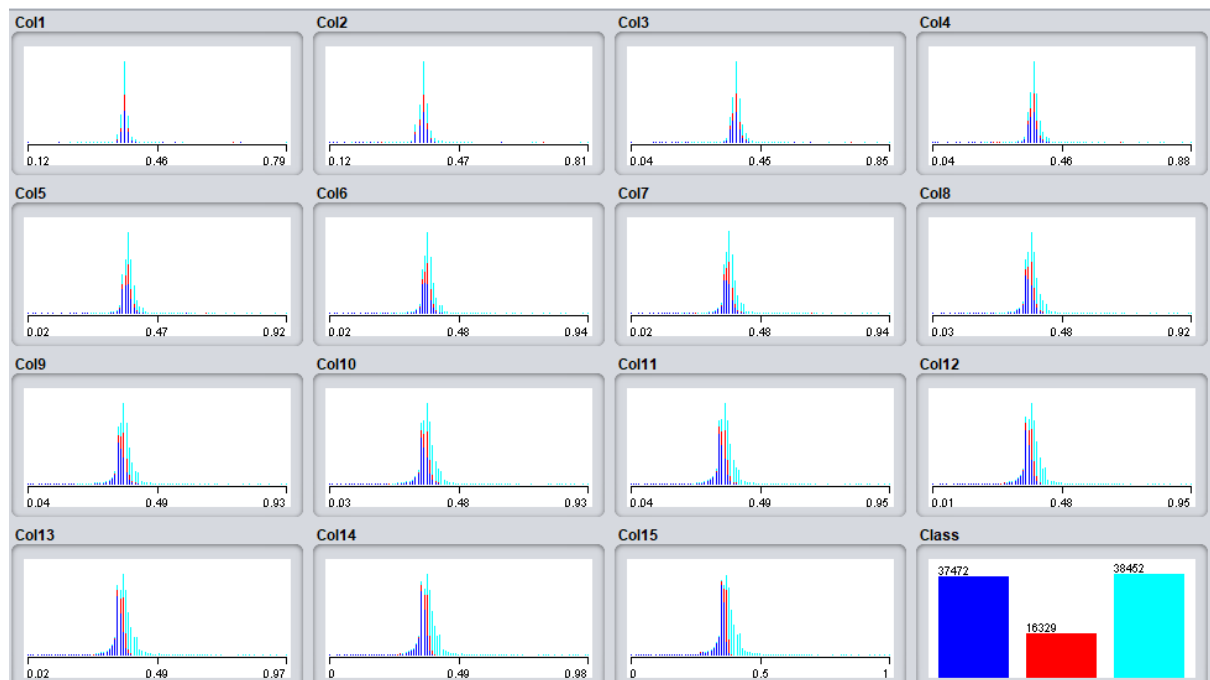


Imagen 37.3

## Módulo 2: Validación de modelos predictivos

### Introducción

En este módulo se desarrolla un proceso de validación. Este proceso permitirá saber qué tan bueno es el modelo al momento de predecir. Esto significa testear el modelo mediante datos actualizados que se ingresan al modelo para que este prediga la suba o baja del precio. Los datos que se utilizan para testear los modelos son los mismos datos históricos que se utilizan para crear el modelo pero actualizados, de diciembre.

Para comenzar con este proceso se selecciona un modelo de cada dataset seleccionado en el módulo 1. Para el DataSet\_1 se elige el mejor modelo de MultilayerPerceptron de red neuronal en el paso 3, y para el DataSet\_2 se elige el modelo generado mediante el DataSet\_3 (generado del DataSet\_2) con MakeDensityBasedClusterer en el caso 2.

Para iniciar este paso de validación es necesario generar los archivos .model que se generan en weka de la siguiente manera como se ve en la Imagen 38.

Click derecho en “Result list” y “Save model”. El archivo se guarda con la extensión .model.

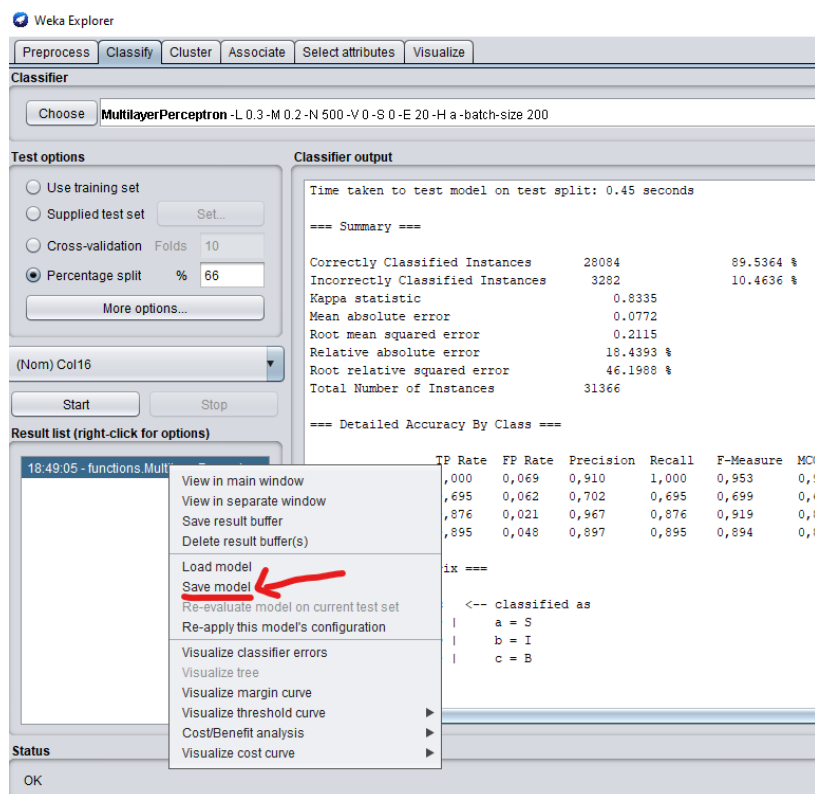


Imagen 38

## Generar DataSets de validación

Para poder validar que tan bueno es este modelo se debe tener los datos con lo que probar el modelo, para eso se busca del mismo sitio donde se recolectó los datos para los dataset iniciales y se realiza el mismo proceso que se hizo con los DataSet iniciales. Este paso de procesar los datos es importante ya que los DataSets de testeo deben tener el mismo formato y columnas que los datasets que se utilizan para crear los modelos.

Para testear el modelo obtenido con el DataSet\_1 se utiliza el método *GenerarDataSet(1, 'DataSet\_1\_Test.csv', 'DataSet\_Inicial\_Test.csv')* donde se le pasa como parámetro en datasetNum el número 1, el segundo parámetro (fileNameGenerated) es el nombre del archivo que se genera y el tercer parámetro (fileNameToProces) es el nombre del archivo inicial en el cual se basará para generar el dataset. Este método crea un nuevo archivo llamado DataSet\_1\_Test.csv.

Para el caso del DataSet\_2 el proceso es más complejo ya que se utilizó 2 métodos para llegar al DataSet\_3. Primero el método *GenerarDataSet(2, 'DataSet\_2\_Test.csv', 'DataSet\_Inicial\_Test.csv')*, donde el primer parámetro que se le pasó en datasetNum es el número 2, el segundo parámetro (fileNameGenerated) es el nombre del archivo que se genera y el tercer parámetro (fileNameToProces) es el nombre del archivo inicial en el cual se basará para generar el dataset. Este método generará un archivo que se llamará DataSet\_2\_Test.csv. Este último archivo se vuelve a procesar de nuevo ya que debe quedar igual a el archivo DataSet\_3. Para esto se utiliza el mismo método nuevamente, *GenerarDataSet(3, 'DataSet\_3\_Test.csv', 'DataSet\_2\_Test.csv')*. El primer parámetro que se le pasa en datasetNum es el número 2, el segundo parámetro (fileNameGenerated) es el nombre del archivo que se genera y el tercer parámetro (fileNameToProces) es el nombre del archivo inicial en el cual se basará para generar el dataset. Este método devolverá el dataset final llamado DataSet\_3\_Test.

## Convertir DataSets de formato csv a arff

Al momento de realizar las validaciones de los modelos, es necesario que los datos con los que se van a hacer las pruebas estén en un formato arff. Para eso se debe cambiar los formatos ya que los archivos que se generan son csv.

Esto significa solo remover la primera fila que indica el nombre de las columnas en el archivo csv y agregar etiquetas @relation que indica el nombre que identifica a los datos, @attribute para describir las columnas y la sesión de @data que es a partir de donde se encuentran los datos. Esto aplica para los dos archivos csv tanto el DataSet\_1.csv que se llamará DataSet\_1.arff y el DataSet\_3.csv que se llamará DataSet\_3.arff.

Imagen 39 (Archivo csv del DataSet 1 Test)

```
@attribute Col1 numeric
@attribute Col2 numeric
@attribute Col3 numeric
@attribute Col4 numeric
@attribute Col5 numeric
@attribute Col6 numeric
@attribute Col7 numeric
@attribute Col8 numeric
@attribute Col9 numeric
@attribute Col10 numeric
@attribute Col11 numeric
@attribute Col12 numeric
@attribute Col13 numeric
@attribute Col14 numeric
@attribute Col15 numeric
@attribute Class { S, I, B }
```

[illegible]

Imagen 40 (Archivo arff del DataSet 1 Test)

## Cargar Datos y Modelo en Weka

### DataSet\_1

El primer paso es generar el modelo (archivo .model), para esto haremos lo mismo que en el **Caso 3** del módulo 1 para el **MultilayerPerceptron** que fue el mejor modelo que se obtuvo para este mismo. Primero se carga el archivo **DataSet\_1.csv**, luego se setea el **hiddenLayers** en "a", el **batchSize** en 200 y por último, el botón **start**.

Una vez terminado el proceso se guarda el modelo de la siguiente manera como se ve en la Imagen 41 y se le asigna como nombre **Model\_DataSet\_1.model**.

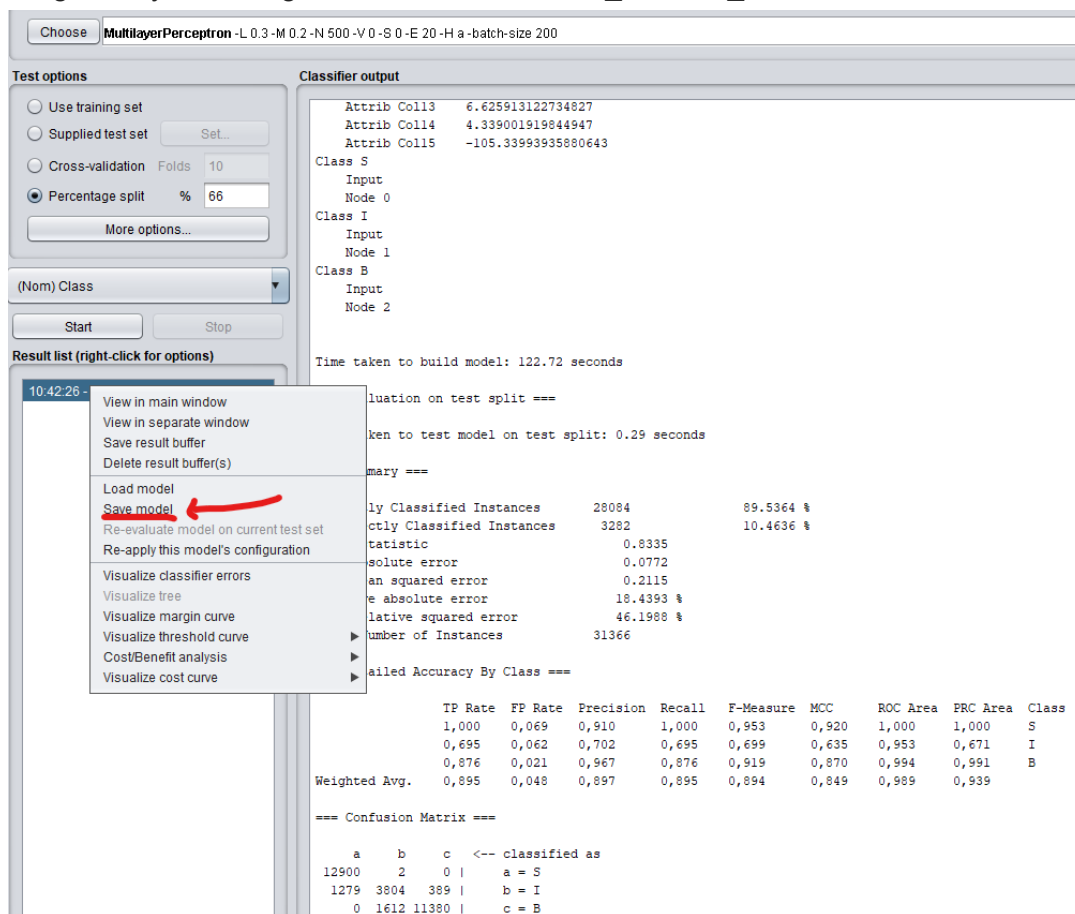


Imagen 41

Después de generar el modelo se debe cargarlo y también los datos que se utilizará para testear.

Primero se carga el modelo generado anteriormente (**Model\_DataSet\_1.model**) como se ve en la Imagen 42.



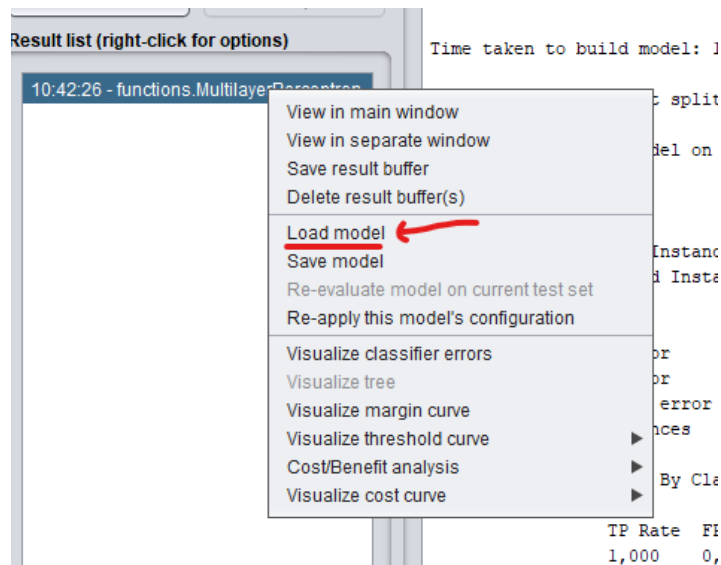


Imagen 42

Luego se carga el archivo con los datos a testear (*DataSet\_1\_Test.arff*) como se ve en la Imagen 43, se configura la carpeta de salida donde arrojará el archivo con las predicciones (con nombre *Predictions\_DataSet\_1.rdp*) como se ve en la Imagen 44 y por ultimo start.

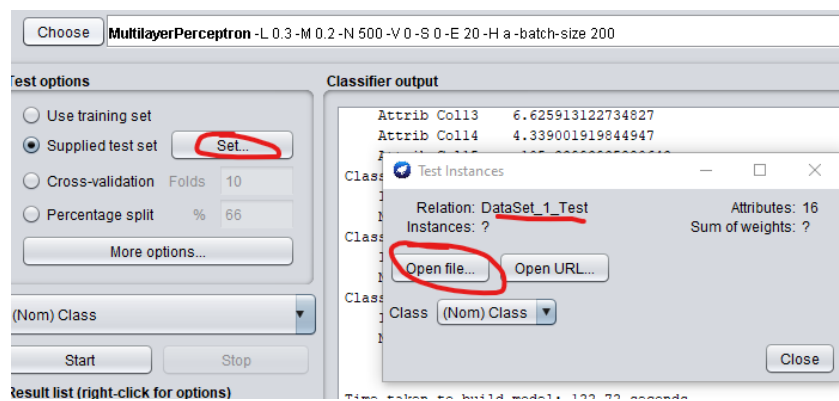


Imagen 43

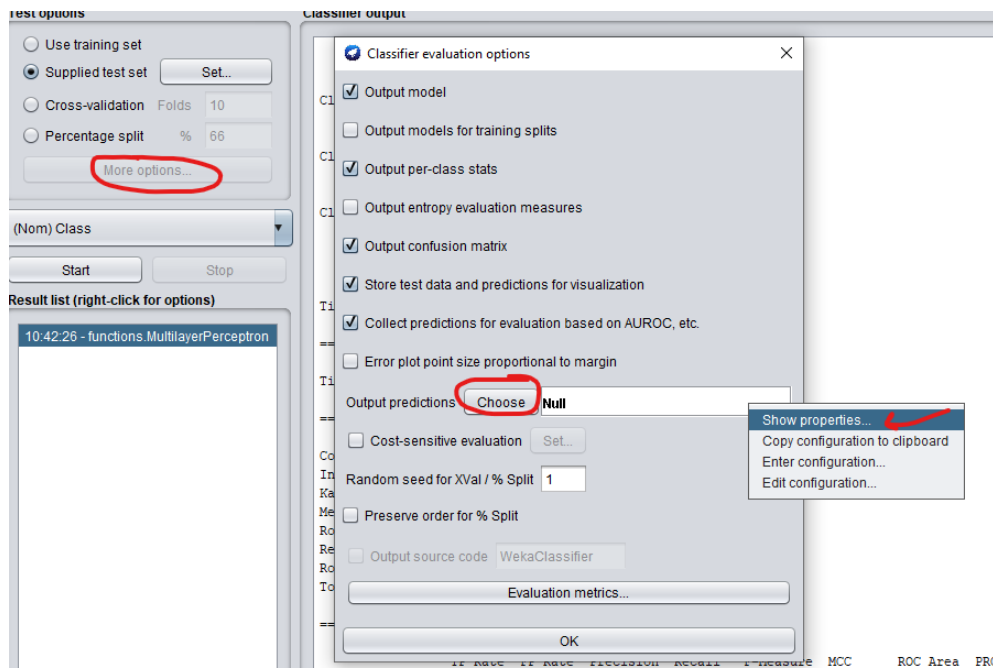


Imagen 44 (Acá se configura el archivo de salida, este archivo tiene como extensión rdp.)

### DataSet\_3

Para generar este modelo (archivo .model) se debe hacer lo mismo que en el **Caso 2** del **MakeDensityBasedClusterer** del módulo 2, luego se genera el modelo. Una vez que se tiene el modelo se debe cargar como en la Imagen 45.

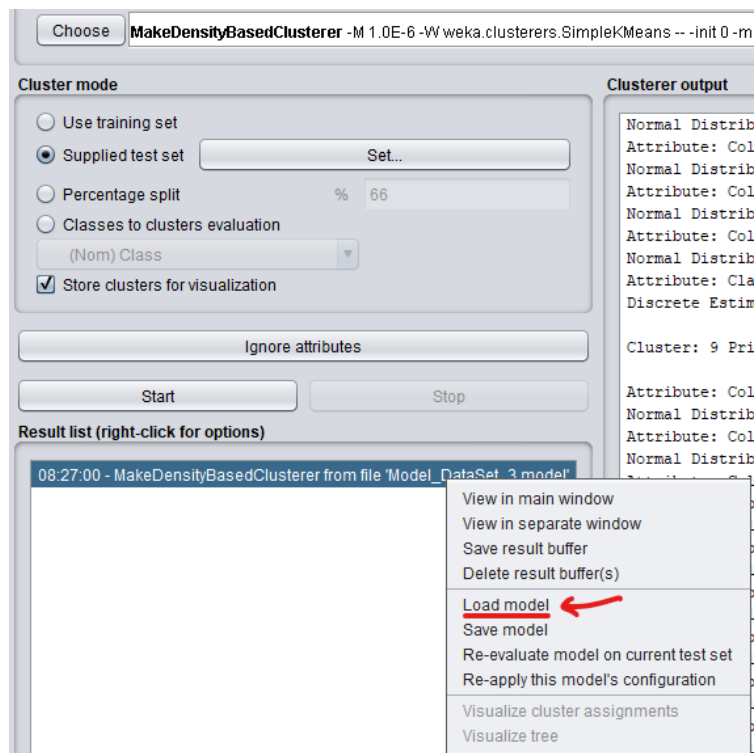


Imagen 45

Luego se cargan los datos que se utilizarán para testear.

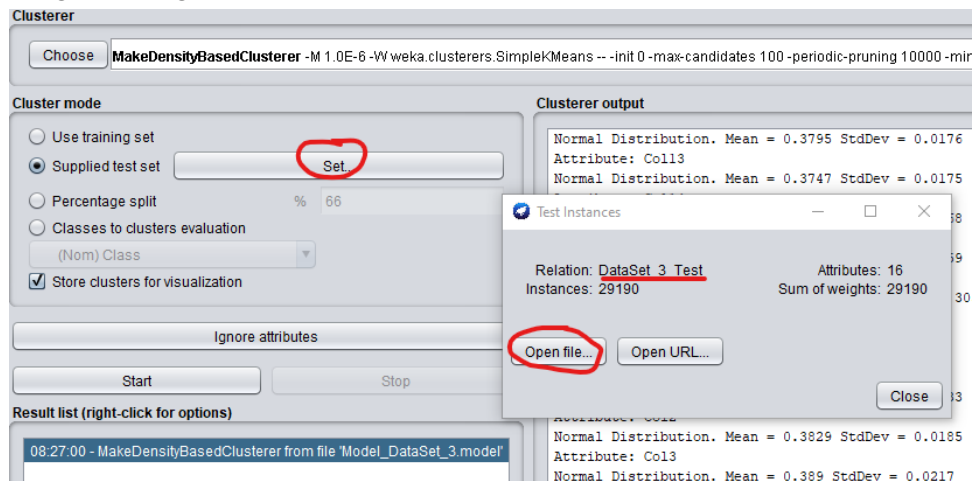


Imagen 46 (archivo DataSet\_3\_Test.arff)

Por último se configura la cantidad de clusters en “numClusters” y se inicia el proceso dando click en start.

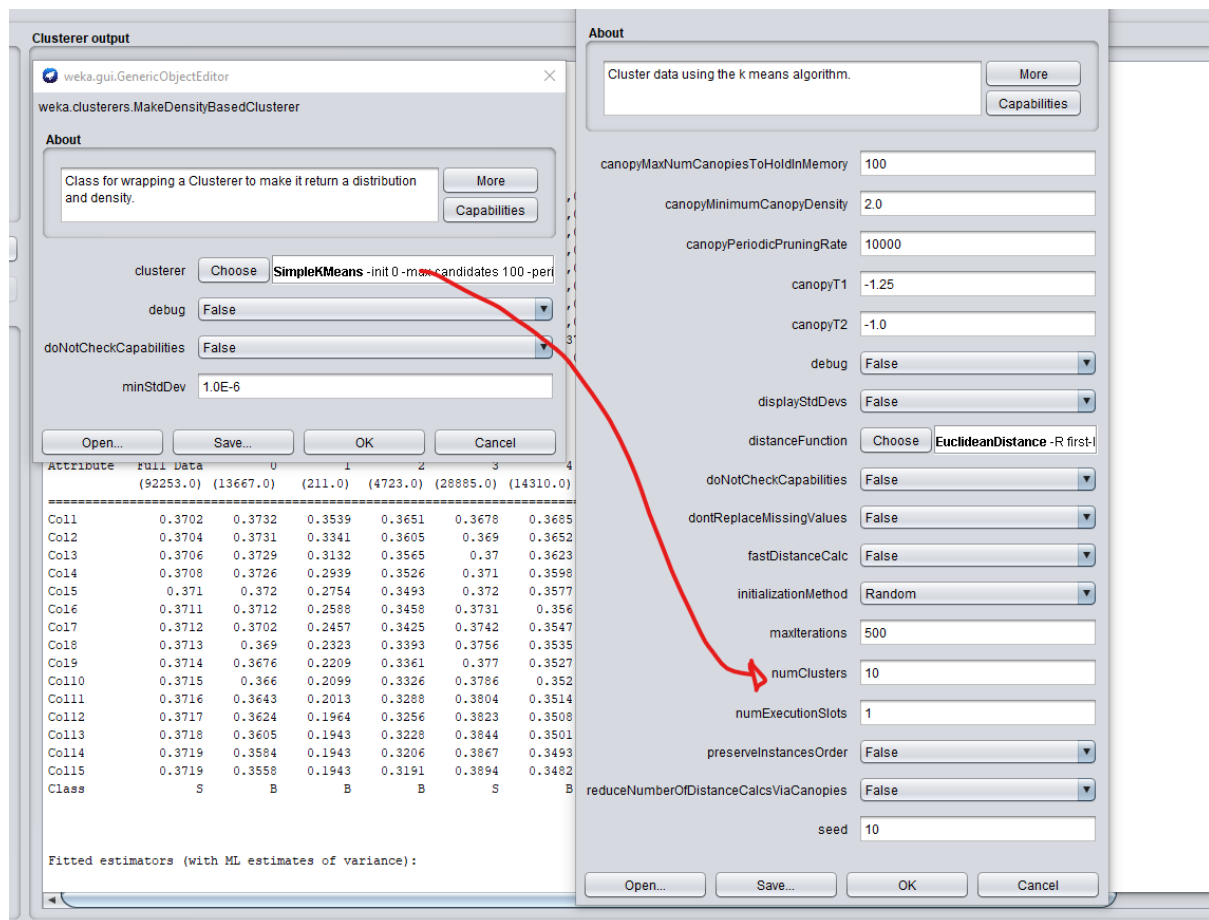


Imagen 47

Una vez que finaliza el proceso se guarda el resultado buffer de la siguiente manera (Imagen 48), con el nombre de Predictions\_DataSet\_3.txt.

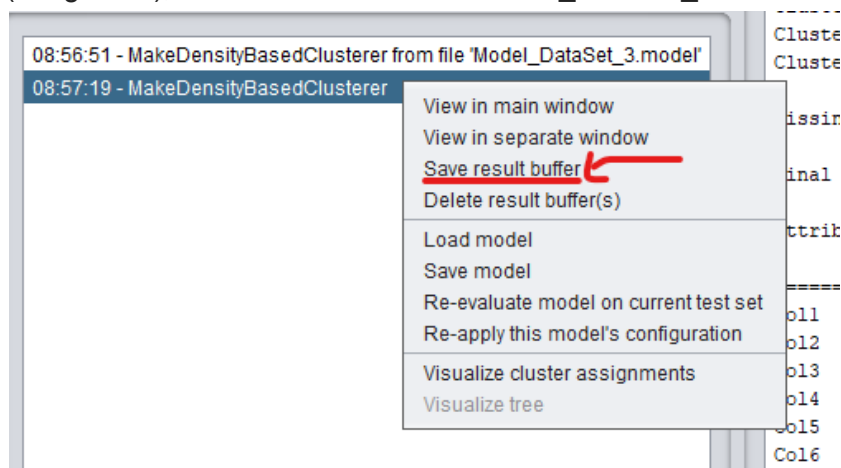


Imagen 48

## Resultado de predicción de red neuronal: **MultilayerPerceptron**

Como resultado de predicciones para este modelo se obtuvo el siguiente archivo mostrado en la Imagen 49. La columna “inst” muestra el número de instancias, “predicted” es el valor que predijo (Puede ser B, I o S) y “prediction” indica el valor de predicción de cero a uno. El valor 1 significa que su acierto fue de un 100%.

inst	predicted	prediction
1	3:B	1
2	3:B	1
3	3:B	1
4	3:B	1
5	3:B	1
6	3:B	1
7	3:B	1
8	3:B	1
9	3:B	1
10	3:B	0.999
11	1:S	0.999
12	1:S	1
13	1:S	0.999
14	1:S	1
15	1:S	1
16	1:S	1
17	1:S	1
18	1:S	1
19	1:S	1

Imagen 49 (archivo de 29190 registros llamado *Predictions\_DataSet\_1.rdp*)

Luego de obtener el archivo con los datos prededidos, se optó por procesar este archivo desarrollando una función en python llamada *CalcularPromedioDePrediccionesDataSet\_1()* para obtener datos como se ve en la siguiente imagen:

```
-----Resultados de modelo MultilayerPerceptron-----
Cantidad de 1: 18582
Cantidad: 29190
Promedio: 99.87
-----Prediccion de Bajas-----
Cantidad de B: 12164
Promedio: 99.94
-----Prediccion de Iguales-----
Cantidad de I: 4847
Promedio: 99.614
-----Prediccion de Subas-----
Cantidad de S: 12179
Promedio: 99.895

Process finished with exit code 0
```

Imagen 50 (captura obtenida luego de ejecutar la función en python)

## Resultado de predicción de metacluster: **MakeDensityBasedClusterer**

Como resultado del modelo generado mediante MakeDensityBasedClusterer, el cual se ha generado utilizando 10 clusters, se obtuvo los siguientes resultados como se ve en la imagen 51 y 52, los 10 clusters y las predicciones para cada uno.

Number of iterations: 88

Within cluster sum of squared errors: 411.0547036141147

Initial starting points (random):

Cluster 0: 0.38,0.38,0.38,0.39,0.38,0.38,0.39,0.39,0.38,0.38,0.38,0.38,0.38,0.37,B  
Cluster 1: 0.37,0.37,0.37,0.38,0.38,0.36,0.36,0.35,0.35,0.35,0.35,0.35,0.35,0.35,B  
Cluster 2: 0.38,0.37,0.36,0.36,0.36,0.36,0.36,0.36,0.36,0.35,0.35,0.35,0.36,0.36,0.35,B  
Cluster 3: 0.37,0.36,0.36,0.36,0.36,0.37,0.37,0.37,0.38,0.38,0.38,0.37,0.38,0.38,0.38,S  
Cluster 4: 0.38,0.38,0.36,0.37,0.36,0.37,0.37,0.37,0.37,0.37,0.37,0.38,0.37,0.37,0.35,B  
Cluster 5: 0.38,0.37,0.37,0.37,0.36,0.35,0.35,0.35,0.37,0.37,0.36,0.36,0.35,0.34,0.34,B  
Cluster 6: 0.37,0.38,0.38,0.38,0.38,0.38,0.38,0.38,0.37,0.37,0.37,0.37,0.37,0.37,I  
Cluster 7: 0.36,0.37,0.36,0.35,0.37,0.35,0.35,0.35,0.35,0.35,0.35,0.35,0.35,0.35,I  
Cluster 8: 0.39,0.38,0.39,0.39,0.39,0.38,0.39,0.4,0.4,0.4,0.39,0.39,0.37,0.36,0.36,B  
Cluster 9: 0.37,0.38,0.38,0.38,0.38,0.38,0.38,0.38,0.38,0.38,0.39,0.39,0.4,0.4,S

Missing values globally replaced with mean/mode

Final cluster centroids:

Attribute	Cluster#										
	Full Data (92253.0)	0 (13667.0)	1 (211.0)	2 (4723.0)	3 (28885.0)	4 (14310.0)	5 (1510.0)	6 (9927.0)	7 (6402.0)	8 (3051.0)	9 (9567.0)
Col1	0.3702	0.3732	0.3539	0.3651	0.3678	0.3685	0.3623	0.373	0.3644	0.3837	0.3762
Col2	0.3704	0.3731	0.3341	0.3605	0.369	0.3652	0.3545	0.3742	0.3629	0.3874	0.3829
Col3	0.3706	0.3729	0.3132	0.3565	0.37	0.3623	0.3464	0.3752	0.3615	0.39	0.389
Col4	0.3708	0.3726	0.2939	0.3526	0.371	0.3598	0.3382	0.3761	0.3604	0.3919	0.3947
Col5	0.371	0.372	0.2754	0.3493	0.372	0.3577	0.3296	0.3768	0.3596	0.393	0.4001
Col6	0.3711	0.3712	0.2588	0.3458	0.3731	0.356	0.3215	0.3772	0.359	0.3934	0.405
Col7	0.3712	0.3702	0.2457	0.3425	0.3742	0.3547	0.3132	0.3775	0.3585	0.3931	0.4096
Col8	0.3713	0.369	0.2323	0.3393	0.3756	0.3535	0.3057	0.3776	0.3583	0.3918	0.4135
Col9	0.3714	0.3676	0.2209	0.3361	0.377	0.3527	0.299	0.3775	0.3583	0.3895	0.4172
Col10	0.3715	0.366	0.2099	0.3326	0.3786	0.352	0.2942	0.3772	0.3587	0.387	0.4203
Col11	0.3716	0.3643	0.2013	0.3288	0.3804	0.3514	0.2905	0.3766	0.3593	0.3834	0.4228
Col12	0.3717	0.3624	0.1964	0.3256	0.3823	0.3508	0.2875	0.3759	0.3603	0.3795	0.4248
Col13	0.3718	0.3605	0.1943	0.3228	0.3844	0.3501	0.2851	0.3751	0.3615	0.3747	0.4264
Col14	0.3719	0.3584	0.1943	0.3206	0.3867	0.3493	0.2836	0.3741	0.3628	0.3694	0.4276
Col15	0.3719	0.3558	0.1943	0.3191	0.3894	0.3482	0.2826	0.373	0.3644	0.3632	0.4285
Class	S	B	B	B	S	B	B	I	I	B	S

*Imagen 51*

Fitted estimators (with ML estimates of variance):

Cluster: 0 Prior probability: 0.1481

Attribute: Col1  
Normal Distribution. Mean = 0.3732 StdDev = 0.0071  
Attribute: Col2  
Normal Distribution. Mean = 0.3731 StdDev = 0.009  
Attribute: Col3  
Normal Distribution. Mean = 0.3729 StdDev = 0.0095  
Attribute: Col4  
Normal Distribution. Mean = 0.3726 StdDev = 0.0098  
Attribute: Col5  
Normal Distribution. Mean = 0.372 StdDev = 0.0097  
Attribute: Col6  
Normal Distribution. Mean = 0.3712 StdDev = 0.0096  
Attribute: Col7  
Normal Distribution. Mean = 0.3702 StdDev = 0.0096  
Attribute: Col8  
Normal Distribution. Mean = 0.369 StdDev = 0.0096  
Attribute: Col9  
Normal Distribution. Mean = 0.3676 StdDev = 0.0096  
Attribute: Col10  
Normal Distribution. Mean = 0.366 StdDev = 0.01  
Attribute: Col11  
Normal Distribution. Mean = 0.3643 StdDev = 0.0102  
Attribute: Col12  
Normal Distribution. Mean = 0.3624 StdDev = 0.0106  
Attribute: Col13  
Normal Distribution. Mean = 0.3605 StdDev = 0.0109  
Attribute: Col14  
Normal Distribution. Mean = 0.3584 StdDev = 0.0111  
Attribute: Col15  
Normal Distribution. Mean = 0.3558 StdDev = 0.0108  
Attribute: Class  
Discrete Estimator. Counts = 13668 1 1 (Total = 13670)

*Imagen 52 (predicciones del cluster 0)*

A continuación se obtiene el porcentaje de predicción para cada cluster, que a su vez cada cluster predice B, I o S.

- (B) Cluster 0:** Total = 13670; Aciertos = 13668; No aciertos = 2; Porcentaje= **99.98%**
- (B) Cluster 1:** Total = 214; Aciertos = 212; No aciertos = 2; Porcentaje= **98.24%**
- (B) Cluster 2:** Total = 4726; Aciertos = 4724; No aciertos = 2; Porcentaje= **99.95%**
- (S) Cluster 3:** Total = 28888; Aciertos = 28886; No aciertos = 2; Porcentaje= **99.99%**
- (B) Cluster 4:** Total = 14313; Aciertos = 14311; No aciertos = 2; Porcentaje= **99.98%**
- (B) Cluster 5:** Total = 1513; Aciertos = 1511; No aciertos = 2; Porcentaje= **99.86%**
- (I) Cluster 6:** Total = 9930; Aciertos = 9928; No aciertos = 2; Porcentaje= **99.97%**
- (I) Cluster 7:** Total = 6405; Aciertos = 6403; No aciertos = 2; Porcentaje= **99.96%**
- (B) Cluster 8:** Total = 3054; Aciertos = 3052; No aciertos = 2; Porcentaje= **99.93%**
- (S) Cluster 9:** Total = 9570; Aciertos = 9568; No aciertos = 2; Porcentaje= **99.97%**

Resultados

$(99.98 + 98.24 + 99.95 + 99.99 + 99.98 + 99.86 + 99.97 + 99.96 + 99.93 + 99.97)/10 =$   
**99.78 (Promedio general de aciertos)**

**Predicción de bajas:**

$(99.98(\text{Cluster0}) + 98.24(\text{Cluster1}) + 99.95(\text{Cluster2}) + 99.98(\text{Cluster4}) + 99.86(\text{Cluster5}) + 99.93(\text{Cluster8})) / 6 (\text{Cantidad clusters}) =$  **99.65 (Promedio de aciertos de bajas)**

**Predicción de iguales:**

$(99.97(\text{Cluster6}) + 99.96(\text{Cluster7})) / 2 (\text{Cantidad de clusters}) =$  **99.965 (Promedio de aciertos de iguales)**

**Predicción de subas:**

$(99.99(\text{Cluster3}) + 99.97(\text{Cluster9})) / 2 =$  **99.98 (Promedio de aciertos de subas)**



## Conclusión final

Comparación entre los resultados de los modelos obtenidos de los dataset iniciales.

Antes, un resumen de los procesos que se hicieron con los dataset iniciales hasta obtener los modelos finales.

Se inició con el dataset inicial llamado DataSet\_Inicial.csv, este dataset contiene los datos crudos. A partir de este dataset se generó dos dataset, el primero DataSet\_1.csv obtenido a partir de “Algoritmo 1 para modelo de datos: Agrupando precios de a 15” y el segundo dataset DataSet\_2.csv obtenido a partir de “Algoritmo 2 para modelo de datos: Restando el siguiente elemento”.

Luego se hizo un proceso de estudio y minería de datos utilizando diferentes herramientas. Para el DataSet\_1.csv se utilizó un selector de atributos para analizar las columnas, en el proceso de datos se probó con árbol de inducción J48 y con redes neuronales MultilayerPerceptron. Al finalizar este proceso se logró obtener mejores resultados con la red neuronal para generar el modelo. Después de obtener el modelo, se hizo un proceso de testeo utilizando datos recientes para ver la cantidad de aciertos que tuvo dicho modelo.

Luego para el DataSet\_2.csv también se ha utilizado un selector de atributos para analizar las columnas, en el proceso de datos se probó con árbol de inducción J48 y con redes neuronales MultilayerPerceptron no obteniendo buenos resultados debido a datos demasiados simples por lo que la red neuronal no pudo desarrollar un modelo confiable que pueda predecir. Para esto se utilizó un proceso de normalización de datos del DataSet\_2.csv obteniendo como resultado el DataSet\_3.csv donde a este se le aplica el MakeDensityBasedClusterer obteniendo buenos resultados. Después de obtener el modelo, se testeo utilizando datos recientes para ver la cantidad de aciertos que tuvo el modelo.

Ya teniendo un resumen de lo que fue el proceso de minería de datos hasta obtener los modelos predictivos se puede hacer una comparación de resultados entre el modelo predictivo DataSet\_1 obtenido mediante el MultilayerPerceptron y el modelo predictivo DataSet\_3 obtenido mediante MakeDensityBasedClusterer.

En resultados generales el promedio de aciertos es similar 99.87 del MultilayerPerceptron y 99.78 para el MakeDensityBasedClusterer.

El que mejores aciertos tiene cuando el precio baja es el MultilayerPerceptron con 99.94 de promedio a diferencia de MakeDensityBasedClusterer con 99.65 de promedio.

Para cuando el precio se mantiene el MakeDensityBasedClusterer es mejor con 99.96 a diferencia de 99.61 del MultilayerPerceptron.

Y para las subas el MakeDensityBasedClusterer es un poco mejor con 99.98 y el MultilayerPerceptron 99.89.

Ambos modelos generan buenos resultados de predicción de un mismo conjunto de datos,

por lo tanto cualquiera de estos dos se podría utilizar tranquilamente más allá de sus diferencias técnicas.

```
-----Resultados de modelo MultilayerPerceptron-----  
Cantidad de 1: 18582  
Cantidad: 29190  
Promedio: 99.87  
-----Prediccion de Bajas-----  
Cantidad de B: 12164  
Promedio: 99.94  
-----Prediccion de Iguales-----  
Cantidad de I: 4847  
Promedio: 99.614  
-----Prediccion de Subas-----  
Cantidad de S: 12179  
Promedio: 99.895  
  
Process finished with exit code 0
```

Imagen 53

Resultados de modelo MakeDensityBasedClusterer

Promedio general: **99.78 (Promedio general de aciertos)**

Predicción de bajas: **99.65 (Promedio de aciertos de bajas)**

Predicción de iguales: **99.965 (Promedio de aciertos de iguales)**

Predicción de subas: **99.98 (Promedio de aciertos de subas)**

## Trabajos a futuros

En esta sesión se describen algunas implementaciones que no se han podido desarrollar en este trabajo pero que quedan a modo de implementar a futuro.

**API:** Desarrollar una api que pueda tomar el precio actual de la divisa EUR-USD del mercado de forex y a travez de algun modelo generado en este trabajo como MultilayerPerceptron o MakeDensityBasedClusterer, procesar el precio actualizado y generar una salida indicando si el precio dentro de los próximos 15 min bajará, se mantendrá o subirá.

**Front-End:** Desarrollar un front-end donde la experiencia de usuario sea visualizar la tendencia del precio del EUR-USD, teniendo en cuenta cómo se prolongará en los próximos 15 minutos.

# Código fuente en Python

Link del repositorio del código fuente:

<https://github.com/matiasacevedo/TesisModulo1/tree/master/venv>

En este link también están todos los archivos utilizados en este trabajo.

## Anexos Materiales y métodos

### Evaluador de atributos

#### **InfoGainAttributeEval**

NOMBRE

`weka.attributeSelection.InfoGainAttributeEval`

SINOPSIS

`InfoGainAttributeEval`

Evalúa el valor de un atributo midiendo la ganancia de información con respecto a la clase. Esta herramienta utiliza la ganancia de información o también llamada entropía concepto que Claude Shannon introdujo (cuando describo esa entropía obtener información de acuerdo a la columna seleccionada). Según Wikipedia [2], en el ámbito de la teoría de la información la entropía, también llamada entropía de la información y entropía de Shannon (en honor a Claude E. Shannon), mide la incertidumbre de una fuente de información. La entropía también se puede considerar como la cantidad de información promedio contenidos. La entropía también se puede considerar como la cantidad de información promedio que contienen los símbolos usados.

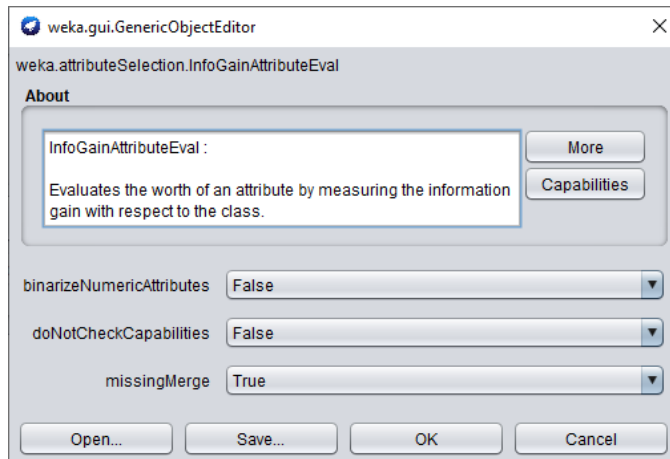
El `InfoGainAttributeEval` utiliza el Ranker para obtener las columnas llamadas atributos más relevantes.

#### **Opciones**

`missingMerge`: distribuye los recuentos de los valores perdidos. Los recuentos se distribuyen entre otros valores en proporción a su frecuencia. De lo contrario, la falta se trata como un valor separado.

`binarizeNumericAttributes`: simplemente binarice los atributos numéricos en lugar de discretizarlos correctamente.

`doNotCheckCapabilities`: si se establece, las capacidades del evaluador no se verifican antes de construir el evaluador (use con precaución para reducir el tiempo de ejecución).



## Selector de atributos

### Ranker

NAME

weka.attributeSelection.Ranker

SINOPSIS

Ranker

Ranker es un selector de atributos de Weka, este selector clasifica los atributos según sus evaluaciones individuales e informa un ranking de los atributos más relevantes.

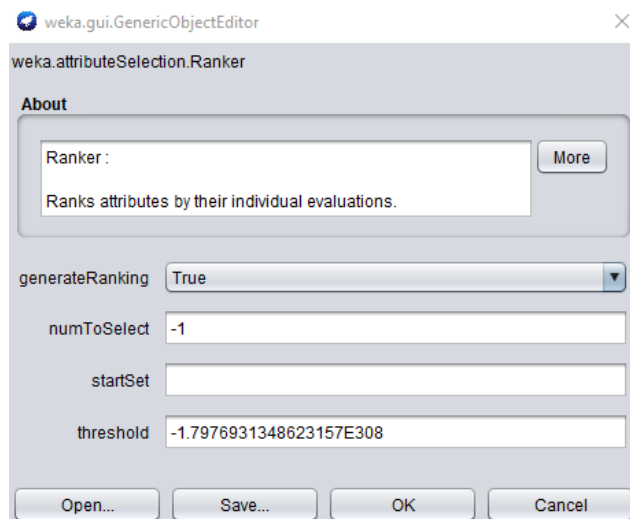
### Opciones

generateRanking: Una opción constante. Ranker solo es capaz de generar clasificaciones de atributos.

numToSelect: Especifique el número de atributos que se conservarán. El valor predeterminado (-1) indica que se conservarán todos los atributos. Utilice esta opción o un umbral para reducir el conjunto de atributos.

threshold: Establece el umbral por el cual se pueden descartar los atributos. El valor predeterminado hace que no se descarten atributos. Utilice esta opción o numToSelect para reducir el conjunto de atributos.

startSet: Especifica un conjunto de atributos para ignorar. Al generar la clasificación, Ranker no evaluará los atributos de esta lista. Esto se especifica como una lista separada por comas de índices de atributos que comienzan en 1. Puede incluir rangos. P.ej. 1,2,5-9,17.



## Clasificadores de predicción

### Árbol de inducción J48

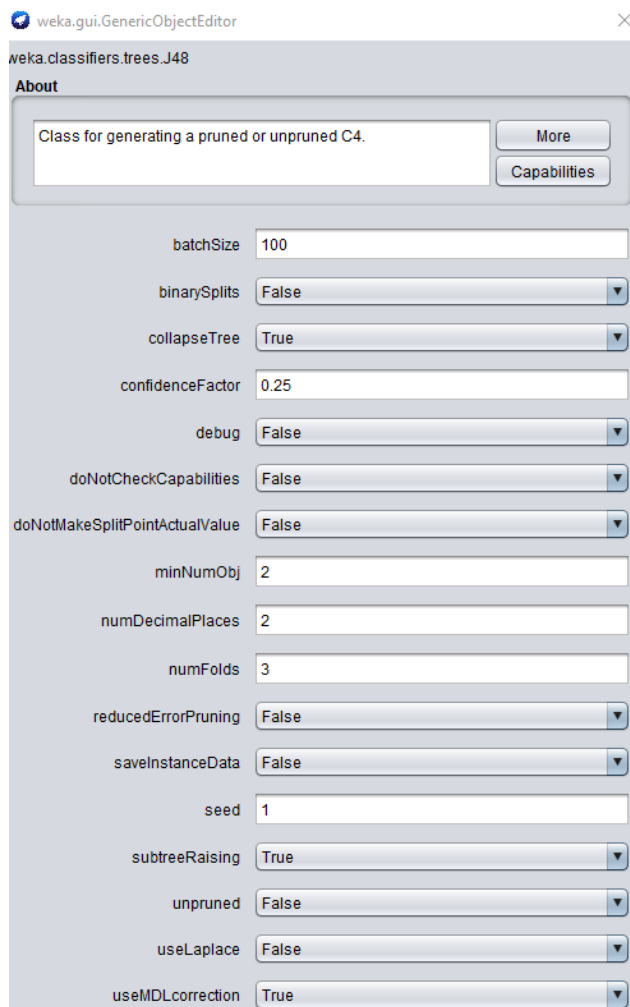
#### NOMBRE

weka.classifiers.trees.J48

#### SINOPSIS

Clase para generar un árbol de decisión C4.5 podado o no podado.

Según María García Jiménez (2007). Es una implementación del algoritmo C4.5, uno de los algoritmos de minería de datos más utilizados. Se trata de un refinamiento del modelo generado con OneR. Supone una mejora moderada en las prestaciones, y poder conseguir una probabilidad de acierto ligeramente superior al del anterior clasificador. El parámetro más importante que deberemos tener en cuenta es el factor de confianza para la poda “confidence level”, que influye en el tamaño y capacidad de predicción del árbol construido. Para cada operación de poda, define la probabilidad de error que se permite a la hipótesis de que el empeoramiento debido a esta operación es significativo. A menor probabilidad, se exige que la diferencia en los errores de predicción antes y después de podar sea más significativa para no podar. El valor por defecto es del 25%. Según baje este valor, se permitirán más operaciones de poda.



## Test Options

Definición según Diego García Morate (2008).

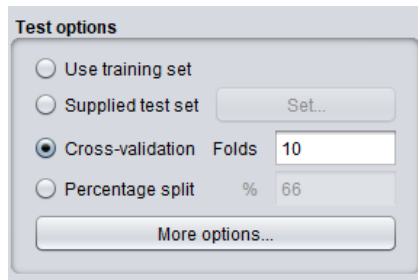
**Use training set:** Con esta opción Weka entrenará el método con todos los datos disponibles y luego lo aplicará otra vez sobre los mismos.

**Supplied test set:** Marcando esta opción tendremos la oportunidad de seleccionar, pulsando el botón Set . . . , un fichero de datos con el que se probará el clasificador obtenido con el método de clasificación usado y los datos iniciales.

**Cross-validation:** Pulsando el botón Cross-validation Weka realizará una validación cruzada estratificada del número de particiones dado (Folds). La validación cruzada consiste en: dado un número  $n$  se divide los datos en  $n$  partes y, por cada parte, se construye el clasificador con las  $n-1$  partes restantes y se prueba con esa. Así por cada una de las  $n$  particiones.

Una validación-cruzada es estratificada cuando cada una de las partes conserva las propiedades de la muestra original (porcentaje de elementos de cada clase).

**Percentage split:** Se define un porcentaje con el que se construirá el clasificador y con la parte restante se probará.



## MultilayerPerceptron (Red neuronal)

NAME

`weka.classifiers.functions.MultilayerPerceptron`

SYNOPSIS

Un clasificador que utiliza retropropagación para aprender un perceptrón multicapa para clasificar instancias.

La red puede construirse a mano o configurarse utilizando una simple heurística. Los parámetros de la red también se pueden monitorear y modificar durante el tiempo de entrenamiento. Los nodos de esta red son todos sigmoides (excepto cuando la clase es numérica, en cuyo caso los nodos de salida se convierten en unidades lineales sin umbral).

Concepto de algunas opciones:

GUI: cómo false, ya que llevaría más tiempo debido a que esta opción muestra con una interfaz gráfica la creación del modelo.

Muestra una interfaz gráfica de usuario. Esto permitirá pausar y alterar la red neuronal durante el entrenamiento.

batchSize: El número preferido de instancias para procesar si se está realizando la predicción por lotes. Se pueden proporcionar más o menos instancias, pero esto brinda a las implementaciones la oportunidad de especificar un tamaño de lote preferido.

hiddenLayers: cantidad de nodos ocultos, seteada con un valor comodín "a" que significa que sumará la cantidad de nodos de entrada (16 columnas) utilizadas para alimentar el modelo. Entonces sería 16 y se la divide a 2.

Seed: usada para inicializar el generador de números aleatorios. Los números aleatorios se usan para establecer los pesos iniciales de las conexiones entre nodos, y también para mezclar los datos de entrenamiento.

momentum: impulso aplicado a las actualizaciones de peso.

nominalToBinaryFilter: esto pre procesa las instancias con el filtro NominalToBinary. Esto podría ayudar a mejorar el rendimiento si hay atributos nominales en los datos.

trainingTime: el número de épocas para entrenar. Si el conjunto de validación no es cero, entonces puede terminar la red antes de tiempo.



debug: si se establece en verdadero, el clasificador puede generar información adicional en la consola.

resume: establece si el clasificador puede continuar entrenando después de realizar el número solicitado de iteraciones.

Tenga en cuenta que establecer esto en verdadero conservará ciertas estructuras de datos que pueden aumentar la tamaño del modelo.

autoBuild: agrega y conecta capas ocultas en la red.

normalizeNumericClass: esto normalizará la clase si es numérica. Esto puede ayudar a mejorar el rendimiento de la red. Normaliza la clase entre -1 y 1. Tenga en cuenta que esto es solo internamente, la salida se reducirá al rango original.

learningRate: la tasa de aprendizaje para las actualizaciones de peso.

doNotCheckCapabilities: si se establece, las capacidades del clasificador no se verifican antes de que se construya el clasificador (use con precaución para reducir el tiempo de ejecución).

reset: esto permitirá que la red se reinicie con una tasa de aprendizaje más baja. Si la red difiere de la respuesta, esto reiniciará automáticamente la red con una tasa de aprendizaje más baja y comenzará a entrenar nuevamente. Esta opción solo está disponible si la GUI no está configurada. Tenga en cuenta que si la red diverge pero no se le permite reiniciar, fallará el proceso de entrenamiento y devolverá un mensaje de error.

weka.classifiers.functions.MultilayerPerceptron

**About**

A classifier that uses backpropagation to learn a multi-layer perceptron to classify instances.

[More](#)

[Capabilities](#)

GUI ☐

autoBuild ☒

batchSize 100

debug ☐

decay ☐

doNotCheckCapabilities ☐

hiddenLayers a

learningRate 0.3

momentum 0.2

nominalToBinaryFilter ☒

normalizeAttributes ☒

normalizeNumericClass ☒

numDecimalPlaces 2

reset ☒

resume ☐

seed 0

trainingTime 500

validationSetSize 0

validationThreshold 20

## MakeDensityBasedClusterer

### MakeDensityBasedClusterer

Envuelve un algoritmo de agrupamiento para que devuelva una distribución de probabilidad y densidad. A cada conglomerado y atributo se le ajusta una distribución discreta o una distribución normal simétrica (de la cual la desviación estándar mínima es un parámetro).

La herramienta de agrupamiento basado en densidad se basa en la detección de en qué áreas existen concentraciones de puntos y dónde están separados por áreas vacías o con escasos puntos. Los puntos que no forman parte de un agrupador se etiquetan como ruido. Opcionalmente, el tiempo de los puntos se puede usar para buscar grupos de puntos que se agrupan en agrupadores de espacio y tiempo.

Esta herramienta utiliza algoritmos de agrupación de aprendizaje de máquina no supervisados que detectan automáticamente patrones basándose puramente en la ubicación espacial y en la distancia a un número de vecinos especificado. Estos algoritmos

se consideran no supervisados porque no requieren ninguna formación sobre en qué consiste un clúster.

Name: weka.clusterers.MakeDensityBasedClusterer

#### SINOPSIS

MakeDensityBasedClusterer es un metaclusterer que envuelve un algoritmo de agrupamiento para que devuelva una distribución de probabilidad y densidad. A cada conglomerado y atributo se le ajusta una distribución discreta o una distribución normal simétrica (de la cual la desviación estándar mínima es un parámetro).

#### OPCIONES

clusterer - el clusterer para envolver

debug: si se establece en true, clusterer puede generar información adicional en la consola.

doNotCheckCapabilities: si se establece, las capacidades del clusterer no se verifican antes de que se construya el clusterer (use con precaución para reducir el tiempo de ejecución).

minStdDev: establece la desviación estándar mínima permitida.

#### Expectation Maximization (EM)

Este algoritmo es útil ya que asigna una distribución de probabilidad a cada instancia que indica la probabilidad de que pertenezca a cada uno de los conglomerados. EM puede decidir cuántos clústeres crear mediante validación cruzada, o puede especificar a priori cuántos clusters generar.

La validación cruzada realizada para determinar el número de clústeres se realiza en los siguientes pasos:

1. El número de clústeres se establece en 1.
2. El conjunto de entrenamiento se divide aleatoriamente en 10 pliegues.
3. La EM se realiza 10 veces utilizando los 10 pliegues de la forma habitual de CV.
4. La log likelihood se promedia sobre los 10 resultados.
5. Si la log likelihood ha aumentado, el número de conglomerados se incrementa en 1 y el programa continúa en el paso 2.

#### K-means

La técnica clásica de agrupamiento k-medias. Primero, especifica de antemano cuántos conglomerados se buscan: este es el parámetro k. Entonces, k puntos se eligen al azar como centros de conglomerados. Todas las instancias se asignan a su centro de conglomerado más cercano de acuerdo con la métrica de distancia euclidiana ordinaria. A continuación, se calcula el centroide, o media, de las instancias de cada grupo; esta es la parte de "medias".

Estos centros se toman como nuevos valores centrales para sus respectivos grupos. Finalmente, todo el proceso se repite con los nuevos centros de clúster. La iteración continúa hasta que se asignan los mismos puntos a cada grupo en rondas consecutivas, en

cuya etapa los centros del grupo se han estabilizado y seguirán siendo los mismos para siempre.

Este método de agrupamiento es simple y efectivo. Es fácil demostrar que elegir el centro del grupo como centroide minimiza la distancia total al cuadrado desde cada uno de los puntos del grupo hasta su centro. Una vez que la iteración se ha estabilizado, cada punto se asigna a su centro de conglomerado más cercano, por lo que el efecto general es minimizar la distancia al cuadrado total desde todos los puntos a sus centros de conglomerado. Pero el mínimo es local; no hay garantía de que sea el mínimo global. Los grupos finales son bastante sensibles a los centros de los grupos iniciales.

### SubsetEval

Más que un método de agrupación Subset Eval permite tener un mecanismo de selección y validación de las variables seleccionadas con las cuales se utilizará para trabajar.

Los evaluadores de subconjuntos toman un subconjunto de atributos y devuelven una medida numérica que guía la búsqueda. Están configurados como cualquier otro objeto Weka. CfsSubsetEval evalúa la capacidad predictiva de cada atributo individualmente y el grado de redundancia entre ellos, prefiriendo conjuntos de atributos que están altamente correlacionados con la clase pero con baja intercorrelación.

Una opción agrega iterativamente atributos que tienen la mayor correlación con la clase, siempre que el conjunto no contenga ya un atributo cuya correlación con el atributo en cuestión sea aún mayor. Los que faltan pueden tratarse como un valor separado o sus recuentos se pueden distribuir entre otros valores en proporción a su frecuencia. ConsistencySubsetEval evalúa los conjuntos de atributos por el grado de coherencia en los valores de clase cuando las instancias de entrenamiento se proyectan en el conjunto. La consistencia de cualquier subconjunto de atributos nunca puede mejorar la del conjunto completo, por lo que este evaluador generalmente se usa junto con una búsqueda aleatoria o exhaustiva que busca el subconjunto más pequeño con una consistencia que es la misma que la del conjunto completo de atributos.

## Cronograma

Actividades	Meses									
	1 a 4	5	6	7	8	9	1 0	1 1	1 2	1 3
Formulación de modelo de tesis: Investigación, planteo del problema, objetivos, definición de conceptos claves y elección de tecnologías a utilizar.	X									

Compilación de datos: Desarrollo de módulo para obtener datos en csv, establecer las variables, Estudio preliminar de calidad de datos e informe de datos.		X								
Análisis de datos y desarrollo de módulos de adaptación a series temporales. Estudio de modelos de agrupación			X	X						
Estudio de modelos de predicción: Adaptación de los datos para usar los modelos predictivos y Aplicación de modelo ID3 y Bayes.					X	X				
Desarrollo de módulos inteligentes: Diseño e implementación, integración del módulo de captura y pruebas y depuración del módulo inteligente							X	X		
Test y evaluación de casos: Diseños de casos, obtener resultados y generar informe.									X	
Redacción final y entrega: Integración de todos los informes parciales y conclusión final.										X

## Referencias

- [1] Forex Tester. (2021). Fuentes de datos.  
<https://forextester.com/es/data/datasources>
- [2] Wikipedia. Entropía (información).  
[https://es.wikipedia.org/wiki/Entrop%C3%ADa\\_\(informaci%C3%B3n\)](https://es.wikipedia.org/wiki/Entrop%C3%ADa_(informaci%C3%B3n))
- [3] Ian H. Witten; Eibe Frank; Mark A. Hall. (2011). Data Mining. Morgan Kaufmann Publishers.  
<https://www.wi.hs-wismar.de/~cleve/vorl/projects/dm/ss13/HierarClustern/Literatur/WittenFrank-DM-3rd.pdf>
- [4] María García Jiménez. (2007). Análisis de Datos en WEKA – Pruebas de Selectividad  
<http://www.it.uc3m.es/~jvillena/irc/practicas/06-07/28.pdf>
- [5] Diego García Morate. (2008). Manual de Weka.  
<https://knowledgesociety.usal.es/sites/default/files/MANUAL%20WEKA.pdf>
- [6] Santiago Morante. (2018). Precauciones a la hora de normalizar datos en Data Science.  
<https://empresas.blogthinkbig.com/precauciones-la-hora-de-normalizar/>
- [7] TUTORESFX. (2020). Introducción al Mercado de Divisas.  
<http://eventosfinancieros.com/pdf/et2020/Ebook-TutoresFx-IntroduccionAlMercadoDeDivisas.pdf>
- [8] Francisca Serrano. (2015). Day trading y operativa bursátil. Centro Libros PAF.  
[https://www.planetadelibros.com/libros\\_contenido\\_extra/35/34022\\_Daytrading\\_para\\_dummies.pdf](https://www.planetadelibros.com/libros_contenido_extra/35/34022_Daytrading_para_dummies.pdf)
- [9] Efrain Cazar M. (2021). El mercado internacional de divisas. Docutech-UPS.  
[https://digitalrepository.unm.edu/cgi/viewcontent.cgi?article=1315&context=abya\\_yala](https://digitalrepository.unm.edu/cgi/viewcontent.cgi?article=1315&context=abya_yala)
- [10] MSc. Mabel Gonzáles Castellanos Lic. César Soto Valero. (2013). Minería de datos para series temporales.  
[https://www.researchgate.net/publication/273447766\\_Mineria\\_de\\_datos\\_para\\_series\\_temporales](https://www.researchgate.net/publication/273447766_Mineria_de_datos_para_series_temporales)
- [11] Mgtr. Estelina Ortega de Gómez. (2014). Aplicación de la metodología box-jenkins en la formulación de un modelo predictivo de las exportaciones de la zona libre de Colón hacia Colombia.  
[http://gfnun.unal.edu.co/fileadmin/content/eventos/simposioestadistica/documentos/memorias/Memorias\\_2016/Posters/6.\\_Metodologia\\_Box-Jenkins\\_Ortega.pdf](http://gfnun.unal.edu.co/fileadmin/content/eventos/simposioestadistica/documentos/memorias/Memorias_2016/Posters/6._Metodologia_Box-Jenkins_Ortega.pdf)
- [12] Miguel Angel Alvarez. 2003. ¿Qué es Python?.  
<https://desarrolloweb.com/articulos/1325.php>
- [13] Edwin Melo Mayta de la Facultad de Ingeniería estadística e informática. (2016). Modelo de predicción mensual de mortalidad general intrahospitalaria en el hospital regional Manuel Nuñez Butron-Puno.  
[http://repositorio.unap.edu.pe/bitstream/handle/UNAP/2092/Melo\\_Mayta\\_Edwin.pdf?sequence=1&isAllowed=y](http://repositorio.unap.edu.pe/bitstream/handle/UNAP/2092/Melo_Mayta_Edwin.pdf?sequence=1&isAllowed=y)

[14] Andrés Pablo Mogni de la Universidad de Buenos Aires. (2013). Modelos de Series de Tiempo con aplicaciones en la industria aerocomercial.  
[http://cms.dm.uba.ar/academico/carreras/licenciatura/tesis/2013/Andres\\_Mogni.pdf](http://cms.dm.uba.ar/academico/carreras/licenciatura/tesis/2013/Andres_Mogni.pdf)