IMPLEMENTACIÓN WEB DE REDES NEURONALES ARTIFICIALES APLICADAS A LA PREDICCIÓN DE SERIES DE TIEMPO.

EDINSON JABID MARTINEZ GOMEZ

LUIS EDUARDO PEDRAZA CABALLERO

INGENIERIA DE SISTEMAS

FACULTAD DE INGENIERIA

UNIVERSIDAD DE LA COSTA

BARRANQUILLA - COLOMBIA 2012

IMPLEMENTACIÓN WEB DE REDES NEURONALES ARTIFICIALES APLICADAS A LA PREDICCIÓN DE SERIES DE TIEMPO.

EDINSON JABID MARTINEZ GOMEZ

LUIS EDUARDO PEDRAZA CABALLERO

Trabajo de investigación presentado como requisito de grado

Tutor:

DARWIN RAMIRO MERCADO POLO

PhD(C) en Sistemas Software Inteligentes y Adaptables

INGENIERIA DE SISTEMAS

FACULTAD DE INGENIERIA

UNIVERSIDAD DE LA COSTA

BARRANQUILLA - COLOMBIA 2012

NOTA DE ACEPTACIÓN:

_____

_____

_____

_____

_____

PRESIDENTE DEL JURADO

_____

JURADO

_____

JURADO

Barranquilla, Agosto del 2012

RESUMEN

En el presente proyecto se presenta el desarrollo e implementación de una plataforma web de dos tipos de redes neuronales artificiales aplicadas a la predicción de series de tiempo. Está desarrollada bajo el lenguaje de programación Python y utiliza ExtJs como framework javascript para la construcción de las interfaces gráficas de usuario.

Permite realizar la simulación del perceptrón multicapa y las redes neuronales basadas en funciones de base de radial. Para la primera se utiliza como algoritmo de aprendizaje el resilient backpropagation, el cual busca minimizar la función del error cuadrático medio para ajustar los pesos de la red.

El proceso de entrenamiento de la red RBF se realiza en dos fases, utilizando inicialmente aprendizaje no supervisado, a través del algoritmo de los k-means, para obtener los centros de las funciones de base radial, posteriormente, se hallan las desviaciones estándar mediante el algoritmo LMS y el ajuste de los pesos se obtiene con la regla de la pseudo-inversa.

Las entradas a la plataforma para realizar la simulación deben importarse a través de archivos en formato (.csv), una vez se han obtenido los resultados se permite representar gráficamente cada uno ellos.

Las simulaciones realizadas en base a series de tiempo permitieron obtener una buena aproximación en la que se conoce si el valor de la variable crecerá o decrecerá. Sin embargo debido a que los algoritmos implementados requieren un hardware de alto costo, se deben buscar otras alternativas como la computación en paralelo o la optimización de los algoritmos implementados.

ABSTRACT

In this Project we show the development and deployment of a web platform with two kinds of artificial Neural Networks applied to forecast time series. It has been developed with the language of programming Python and use Extjs 4 for the client side.

Allows the simulation of multilayer perceptron and neural networks based on radial basis functions. For the first algorithm it is used as the resilient backpropagation learning, which attempts to minimize the mean square error function to adjust the network weights.

The training process of the RBF network is performed in two phases, using initially unsupervised learning, through the algorithm of k-means, for the centers of the radial basis function, subsequently standard deviations are found by the LMS algorithm and adjustment of the weights is obtained with the rule of the pseudo-inverse.

The inputs to the simulation platform have to be imported through files (.csv) format, once we have obtained the results are graphically represented each of them. Simulations performed on a time series basis allow to obtain a good approximation in that is known if the variable value will increase or decrease.

But because the algorithms implemented require expensive hardware, we need to look for alternatives such as parallel computing and optimization algorithms implemented.

DEDICATORIA

A mi hijo Sebastián y a mi hija Sofía, por ser mi fuente de inspiración. A mi esposa Diana por tenerme la paciencia para poder obtener este logro.

EDINSON MARTÍNEZ

A mi familia, en especial a mi madre por brindarme siempre su apoyo, a los profesores Sonia Valbuena y Darwin Mercado por ser mis guías en mi proceso de formación académico e investigativo.

LUIS PEDRAZA

AGRADECIMIENTOS

Los autores expresan sus agradecimientos a todas las personas e instituciones que con su apoyo hicieron posible la realización de este proyecto, como son:

Principalmente a Dios que nos ha dado la inteligencia, la fortaleza en situaciones difíciles y la salud para afrontar los diferentes retos.

A nuestro tutor Darwin Mercado, por su confianza y apoyo incondicional en la realización del proyecto y en nuestro proceso de formación profesional.

Al doctor Carlos Fajardo, por indicarnos el camino en el desarrollo del presente trabajo.

A los profesores Hernán Pájaro, Alexis De La Hoz, Adriana Granados, Paola Ariza, Jorge Hernández, Emiro De La Hoz, Jazmín Cabarcas, Francisco Racedo, Sonia Valbuena, Darwin Mercado por brindarnos su conocimiento.

A nuestros amigos y compañeros por compartir con nosotros los logros obtenidos, alegrías, tristezas, trasnochos y experiencias.

A la Facultad de Ingeniería de Sistemas de la Universidad de la Costa, por brindarnos las herramientas necesarias para nuestra formación.

TABLA DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

Una serie de tiempo es un conjunto de datos obtenidos a partir de la observación de un fenómeno determinado durante periodos de tiempos iguales, representando el cambio de una variable especifica de tipo económico, físico, químico, financiero, biológico, entre otros.

Cuando se trabaja con series temporales, una de las tareas más importantes es la de predecir los datos futuros de la serie, es decir, a partir de los datos del pasado proyectar los valores que tomara la variable determinada. Para llevar a cabo esta actividad se construye un modelo matemático que capture, total o parcialmente las características de esta, por ejemplo los modelos ARIMA. Estos, son modelos autorregresivos integrados de medias móviles y han demostrado gran utilidad en la predicción a corto plazo de series de alta frecuencia. En contraste a los modelos ARIMA y métodos estadísticos, las redes neuronales artificiales son consideradas más robustas, especialmente en la representación de relaciones complejas que exhiben comportamientos no lineales [1].

Las Redes Neuronales Artificiales (RNA) son sistemas de procesamiento de la información cuya estructura y funcionamiento están inspirados en las redes neuronales biológicas. En los últimos años se ha masificado el uso de las RNA para la predicción de series temporales gracias a su capacidad generalizadora por el hecho de aprender a partir de ejemplos, tal y como lo hace el cerebro humano [2].

En el presente proyecto se realiza una implementación web de dos tipos de redes neuronales artificiales aplicadas al dominio de la predicción de series de tiempo, convirtiéndose en una plataforma integrada accesible desde cualquier lugar del mundo y en cualquier instante de tiempo.

Inicialmente se presentan los objetivos del proyecto y la importancia de desarrollarlo. Posteriormente el marco teórico que abarca las redes neuronales artificiales y su aplicación en la predicción de series de tiempo. Luego se detalla el diseño de la plataforma y las herramientas informáticas utilizadas. Finalmente se presentan los resultados obtenidos, las conclusiones y el trabajo futuro.

2. OBJETIVOS:

2.1 General:

- Desarrollar una plataforma web para diferentes tipos de redes neuronales artificiales aplicadas a la predicción de series de tiempo.

2.2 Específicos

- Seleccionar los tipos de redes neuronales y algoritmos más ideales para la construcción de sistemas ANS a partir de la exploración de diferentes tipos de redes neuronales aplicados a la predicción de series de tiempo.
- Diseñar los componentes de la plataforma web de acuerdo a los tipos de redes neuronales artificiales seleccionados.
- Desarrollar e implementar los componentes diseñados.
- Socializar el proyecto ante la comunidad académica y científica de las ciencias computacionales.

3. PLANTEAMIENTO DEL PROBLEMA.

Una de las tareas más importantes en el dominio de las series temporales es la predicción de esta, es decir obtener los valores futuros tomando como base los datos del pasado. Para esto son utilizados modelos matemáticos que permiten capturar las características de la serie, total o parcialmente, y obtener buenos resultados de predicción [1]. Sin embargo, hallar estos modelos se convierte en un trabajo muy complejo y tedioso para cualquier analista de series de tiempo, sobre todo cuando se trabaja con relaciones complejas que exhiben comportamientos no lineales.

Ante esto, las redes neuronales artificiales se han convertido en una solución eficiente y eficaz para el dominio planteado. En este contexto, se han desarrollado ANS (sistemas de redes neuronales artificiales) independientes y de diferentes tipos aplicados a la predicción de series temporales, sin contar con una plataforma integrada.

Los sistemas que se han desarrollado para el análisis de series de tiempo están limitados a un tipo de red neuronal artificial y son desarrolladas para ambiente desktop, generando como consecuencia problemas para la distribución de la aplicación, como por ejemplo, la generación de ejecutables dependiendo del sistema operativo, características de hardware necesarias para su funcionamiento, etc. Además de esto, el deseo de trabajar en un lugar y luego trasladarse a otro y continuar con el trabajo genera muchas complicaciones.

Por otro lado para el desarrollo de investigaciones en los cuales se relacionan las redes neuronales artificiales y las series temporales se torna muy complejo el contraste de los resultados que se obtienen con cada tipo de red neuronal.

De acuerdo a lo anterior, se plantea la siguiente pregunta problema:

¿Es posible el desarrollo de una plataforma web que integre diferentes tipos de redes neuronales artificiales aplicadas a la predicción de series de tiempo y facilite las tareas al análisis de datos en este dominio?

A esta pregunta se le encontrará de dar respuesta en el presente trabajo.

4. JUSTIFICACIÓN

Hoy en día para los líderes de las organizaciones de cualquier índole es muy relevante conocer el valor por anticipado de ciertas variables que afectan su funcionamiento, con el objetivo de equivocarse lo menos posible en la toma de decisiones.

Por ejemplo, el rector de una universidad desearía conocer el número de estudiantes que recibirá el siguiente semestre para un periodo determinado para así decidir si aumentar o disminuir el plantel de profesores.

Además de esto, el desarrollo de investigaciones que relacionan las redes neuronales artificiales y series temporales requiere de la comparación de resultados entre diferentes tipos de RNA. Para esto, no se cuenta con una plataforma web integrada de redes neuronales aplicadas a este dominio.

Otro factor importante, es el acceso a estas aplicaciones, el cual se debe permitirse desde cualquier lugar del mundo y en cualquier instante de tiempo, utilizando una arquitectura cliente/servidor.

Teniendo en cuenta lo mencionado anteriormente, se hace necesario desarrollar una plataforma web donde se integren diferentes tipos de redes neuronales artificiales aplicadas al dominio de la predicción de series de tiempo, que permitan una interacción hombre-máquina bastante cómoda y que sea accesible por cualquier persona, en cualquier lugar del mundo y en cualquier instante de tiempo a través de un navegador web.

5. ALCANCE

El proyecto será desarrollado en el departamento de sistemas de la Universidad de la Costa, ubicada en Barranquilla, Colombia.

En la plataforma web serán implementados dos tipos de redes neuronales artificiales, los cuales serán seleccionados en la primera fase del proyecto. Permitirá configurar los parámetros libres de los tipos de red neuronal implementados. Estará compuesto por una interfaz fácil de utilizar, en la cual se proporcionarán las entradas a través de archivos delimitados por comas y se exportarán los resultados a Excel. La forma de mostrar los resultados obtenidos será en grillas dinámicas, y se realizará una representación graficas de los mismos.

6. REDES NEURONALES ARTIFICIALES.

Las Redes Neuronales Artificiales (RNA) o sistemas conexionistas son sistemas de procesamiento de la información cuya estructura y funcionamiento están inspirados en las redes neuronales biológicas. Consisten en un conjunto de elementos simples de procesamiento llamados nodos o neuronas conectadas entre sí por conexiones que tienen un valor numérico modificable llamado peso [3].

La actividad que una unidad de procesamiento o neurona artificial realiza en un sistema de este tipo es simple. Normalmente, consiste en sumar los valores de las entradas (inputs) que recibe de otras unidades conectadas a ella, comparar esta cantidad con el valor umbral y, si lo iguala o supera, enviar activación o salida (output) a las unidades a las que esté conectada. Tanto las entradas que la unidad recibe como las salidas que envía dependen a su vez del peso o fuerza de las conexiones por las cuales se realizan dichas operaciones [4].

La arquitectura de procesamiento de la información de los sistemas de RNA se distingue de la arquitectura convencional Von Neumann (fundamento de la mayor parte de los ordenadores existentes) en una serie de aspectos fundamentales.

En primer lugar, el procesamiento en un sistema conexionista no es secuencial sino paralelo, esto es, muchas unidades de procesamiento pueden estar funcionando simultáneamente [3].

En segundo lugar, la información que posee un sistema no está localizada o almacenada en compartimentos discretos, sino que está distribuida a lo largo de los parámetros del sistema. Los parámetros que definen el conocimiento que una red neuronal posee en un momento dado son sus conexiones y el estado de activación de sus unidades de procesamiento [3].

Las neuronas artificiales son modelos que tratan de simular el comportamiento de las neuronas biológicas, cada neurona se representa como una unidad de proceso que forma parte de una entidad mayor, esto se conoce como red neuronal. El proceso consta de una

serie de entradas x_i, que equivalen a las dendritas de donde reciben la estimulación, ponderadas por unos pesos w_i que representan los impulsos entrantes, son evaluados y se combinan con la función de red que dará el nivel de potencial de la neurona; la salida de la función de red es evaluada en la función de activación que da lugar a la salida de la unidad de proceso, ver figura 1. Una neurona artificial se comporta como la neurona biológica pero de una forma muy simplificada.

Neurona Biológica	Neurona Artificial
Soma	Neurona
Dendrita	Entrada
Axón	Salida
Sinapsis	Peso

Tabla 1. Analogía neurona biológica y neurona artificial

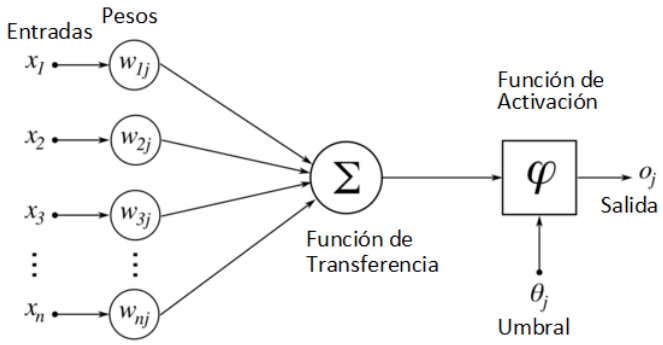

Figura 1. Modelo de una neurona artificial.

La salida de la red neuronal viene dada de la siguiente manera:

$$o_j = f(\sum w_{ij}\, x_j - \theta_i)$$

Donde $f(.)$, es la función de activación de la neurona. Dentro de las más utilizadas se encuentran la función identidad, escalón, lineal a tramos, sigmoidea y sinusoidal.

Función	Ecuación	Rango
Identidad	$y = x$	$[-\infty, \infty]$

Escalón	$y = \begin{Bmatrix} 1, si\ x \geq 0 \\ 0, si\ x < 0 \end{Bmatrix}$ $y = \begin{Bmatrix} 1, si\ x \geq 0 \\ -1, si\ x < 0 \end{Bmatrix}$	$[-1,1]$
Lineal a tramos	$y = \begin{Bmatrix} x, si -1 \leq x \leq 1 \\ 1, si\ x > 1 \\ -1, si\ x < -1 \end{Bmatrix}$	$[-1,1]$
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = \tanh(x)$	$[0,1]$ $[-1,1]$
Sinusoidal	$y = sen(wx + \theta)$	$[-1,1]$

Tabla 2. Funciones de activación de una neurona artificial.

El aprendizaje en una RNA es un proceso de ajuste o modificación de los valores o pesos de las conexiones, en el que se logra que las salidas del sistema sean lo más parecidas a las salidas deseadas proporcionadas por el usuario. Las redes neuronales artificiales pueden clasificarse de acuerdo con el tipo de aprendizaje que utilizan.

6.1 Clasificación de las Redes Neuronales Artificiales.

Las redes neuronales requieren un proceso de entrenamiento previo a su utilización. Este comienza con la asignación de valores a los pesos asociados a cada conexión, y la definición de los parámetros de aprendizaje utilizados. En general, los valores de los pesos se eligen de forma aleatoria. Tras realizarse esta operación se lleva a cabo el entrenamiento de la red, lo cual conlleva la adaptación de estos pesos. Existen dos formas de aprendizaje claramente diferenciadas [5]:

- Aprendizaje supervisado (supervised): requiere la presencia de pares de vectores, cada uno de ellos compuesto por los valores de salida de la red correspondientes a los valores de entrada. En este caso, es necesario que el entrenamiento de la red sea controlado por un supervisor o un conjunto de reglas.

- Aprendizaje no supervisado (unsupervised): en el cual se utilizan sólo los vectores de entrada para entrenar la red. Los valores de las salidas son determinados por la red durante el curso del aprendizaje. Se utiliza para construir modelos internos que capturan similitudes en los valores de los vectores de entrada, sin recibir información adicional.

6.2 Perceptrón Multicapa – MLP

El perceptrón multicapa (Multilayer Perceptron MLP) propuesto por Rumelhart es el componente principal de una red neuronal, proporciona la base para la mayoría de las aplicaciones de las redes neuronales [6].

El MLP es una red formada por una capa de entrada, al menos una capa oculta y una capa de salida, tal como se muestra en la siguiente figura.

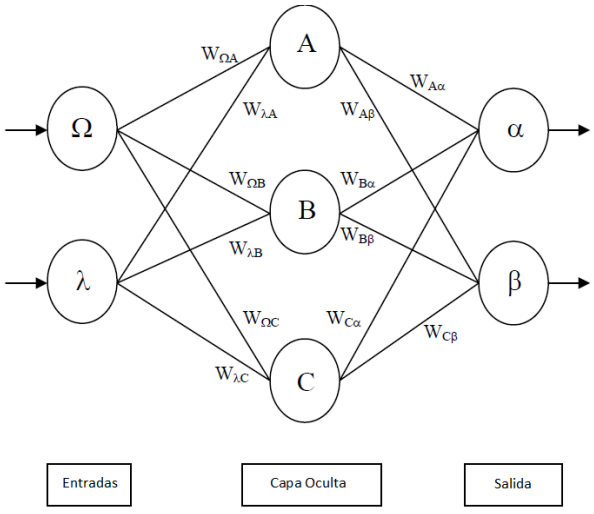

Figura 2. Red Neuronal Perceptrón Multicapa.

Dentro de las características más importantes del perceptrón multicapa se encuentran las siguientes:

- Se trata de una estructura altamente no lineal.

- Presenta tolerancia a fallos.
- El sistema es capaz de establecer una relación entre dos conjuntos de datos.
- Existe la posibilidad de realizar una implementación hardware.

El algoritmo más popular de aprendizaje para el perceptrón multicapa es el backpropagation, el cual consiste en utilizar el error generado por la red y propagarlo hacia atrás, es decir, reproducirlo hacia las neuronas de las capas anteriores.

El algoritmo se describe de la siguiente manera:

- Inicializar los pesos de la red neuronal con números aleatorios y pequeños.
- Aplicar las entradas a la red neuronal y obtener la salida.
- Calcular el error producido por la red neuronal, dado por la siguiente expresión:

$$e_i = (d_i - y_i)y'(x_i) \qquad (1)$$

Donde x_i representa el patrón de entrada i, y_i la salida de la red, d_i la salida deseada, $y'(x_i)$ representa la derivada de la función de salida de la red y e_i el error producido por la red.

- Actualizar los pesos entre las neuronas de la capa oculta y las neuronas de salida, mediante la siguiente expresión:

$$w_{ij} = w_{ij} + ne_i * x_i \qquad (2)$$

Donde n representa la constante de aprendizaje de la re neuronal. Generalmente se utiliza un valor entre -1 y 1.

1- Calcular el error de las neuronas de las capas ocultas, utilizando el error producido por la capa de salida.

$$e_k = g'(x_i)(e_i * w_{ij}) \qquad (3)$$

Donde $g'(x_i)$ representa la derivada de la función de salida de la neurona de la capa oculta y e_k el error generado en la neurona de capa oculta.

2- Actualizar los pesos de la capa oculta utilizando la expresión (2).

Los algoritmos de aprendizaje de las redes neuronales tratan de determinar los pesos sinápticos de manera que se minimice el error cuadrático. Para una iteración dada, el error cuadrático se calcula de la siguiente manera:

$$E(k) = \frac{1}{2}\sum_{i=1}^{M}(d_i - y_i)^2 \qquad (4)$$

Donde M es el numero de datos de entrenamiento, di es la salida deseada y yi es la salida de la red.

El algoritmo backpropagation para el MLP presenta ciertas desventajas, como son: lentitud de convergencia, precio a pagar por disponer de un método general de ajuste funcional, puede incurrir en sobreaprendizaje, fenómeno directamente relacionado con la capacidad de generalización de la red. Y no garantiza el mínimo global de la función de error, tan solo un mínimo local.

Ante las desventajas presentadas se han desarrollado variantes que buscan mitigar estos inconvenientes, tal es el caso del resilient backpropagation, considerado como uno de los algoritmos basados en gradiente más adecuados para entrenar redes neuronales artificiales.

El algoritmo resilient backpropagation es considerado como uno de los algoritmos más robustos para la estimación de los parámetros (o pesos) de una red neuronal. En este proceso se busca encontrar los valores de los parámetros la red neuronal tal que se minimice la diferencia entre los valores deseados y los valores calculados por la red.

El algoritmo RPROP, y sus variantes, difiere de la técnica clásica de propagación hacia atrás del error (o algoritmo backpropagation) en que las derivadas parciales de la función de error sólo son usadas para determinar el sentido en que deben ser corregidos los pesos de la red pero no las magnitudes de los ajustes. Los algoritmos basados en backpropagation modifican los valores de los parámetros proporcionalmente al gradiente de la función de error, de tal forma que en regiones donde el gradiente tiende a ser plano, el algoritmo avanza lentamente. RPROP tampoco se ve afectado por la saturación de las neuronas de la red neuronal, ya que solamente se usa la derivada para determinar la dirección en la

actualización de pesos. Consecuentemente, converge más rápidamente que los algoritmos basados en backpropagation [1]

6.3 Redes Neuronales de Funciones de Base Radial – RBF

Una red de funciones de base radial es una red neuronal que utiliza funciones de base radial como funciones de activación. La arquitectura utilizada por las RBF es muy similar a la del perceptrón multicapa, con la característica de que las RBF utilizan siempre tres capas; una capa de entrada, una capa oculta y una de salida, mientas que los MLP pueden tener más.

En las RBF la capa oculta realiza una transformación no lineal del espacio de entrada. Las neuronas de esta capa son las funciones de base radial y cada neurona de la capa de salida es un combinador lineal [6].

De modo general, el valor generado por una red RBF con una única salida, viene definido por la Ecuación:

$$y_i = \sum_{j=1}^{N} w_j f_j\left(\frac{\|x_j - u_j\|}{\sigma_j}\right) \qquad (5)$$

Donde $f_j(.)$ representa el centro *j* que implementa la función de base radial, u_j y σ_j son los vectores correspondientes al centro y la varianza de la función de base radial *j* y w_j es el peso de la conexión que une el centro *j* con la neurona de salida.

Las redes RBF son excelentes aproximadores universales y los parámetros que definen el proceso de aproximación son:

- Los pesos entre los centros y las neuronas del nivel de salida.
- La posición de los centros.
- Las funciones de Gauss de los centros.

Los pesos se ajustan utilizando el algoritmo de mínimos cuadrados ordinarios (LMS, Least Mean Square), pero se pueden utilizar también otros algoritmos como el método de la

pseudo-inversa. El algoritmo de los mínimos cuadrados ordinarios (LMS) se puede resumir como se muestra a continuación [5].

- Inicializar los pesos en valores aleatorios pequeños.
- Calcular la salida de la red para el patrón de entrada presentado.
- Calcular el error generado por la red. Con la siguiente ecuación:

$$e_i = d_i - y_i \qquad (6)$$

Donde *di* es la salida deseada y *yi* es la salida de la red.

- Actualizar los pesos de la red, mediante la siguiente expresión:

$$w_i = w_i + ne_i x_i \qquad (7)$$

Donde *wi* son los pesos entre las neuronas de la capa oculta y las neuronas de la capa de salida, *n* es la constante de aprendizaje y *xi* es la salida de los centros.

Para ajustar los pesos utilizando el método de la pseudo-inversa, es necesario representar matricialmente las salidas deseadas como el producto de las salidas de los centros por el vector de pesos, como se muestra a continuación [6]:

$$\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} = \begin{bmatrix} \exp\left[-\frac{\|x_1-u_1\|^2}{2\sigma_1^2}\right] & \cdots & \exp\left[-\frac{\|x_1-u_n\|^2}{2\sigma_n^2}\right] \\ \exp\left[-\frac{\|x_2-u_1\|^2}{2\sigma_1^2}\right] & \cdots & \exp\left[-\frac{\|x_2-u_n\|^2}{2\sigma_n^2}\right] \\ \vdots & \vdots & \vdots \\ \exp\left[-\frac{\|x_n-u_1\|^2}{2\sigma_1^2}\right] & \cdots & \exp\left[-\frac{\|x_n-u_n\|^2}{2\sigma_n^2}\right] \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

Figura 3. Representación matricial de las salidas deseadas en una red RBF.

De esta manera se obtiene la siguiente relación:

$$D = G * W \qquad (8)$$

Donde D es el vector de salidas deseadas, W es el vector de pesos y G es la matriz de salida de los centros de la RBF.

De la expresión anterior se obtiene el ajuste de pesos aplicando la pseudo inversa de la matriz G:

$$W = G^{+} * D \qquad (9)$$

Donde G^{+} es la pseudo-inversa de la matriz G:

$$G^{+} = (G^{T}G)^{-1}G^{T} \qquad (10)$$

Los centros de la capa intermedia se pueden seleccionar utilizando diferentes métodos:

- Selección aleatoria de centros: Algunos vectores del espacio de entrada se pueden elegir como centros. Si es posible, deben ser seleccionados de tal forma que estén repartidos de manera regular por todo el espacio de entrada. En este caso, la desviación estándar de la función de Gauss utilizada por los centros, se elige teniendo en consideración la separación entre ellos.
- Autoselección de centros: Los centros se pueden mover durante el entrenamiento de la red. Una posibilidad es la utilización de métodos de entrenamiento que no requieran supervisión (a la vez que se utiliza entrenamiento o aprendizaje supervisado para ajustar los pesos, por ejemplo utilizando el al-goritmo LMS). Los centros se sitúan en aquellas zonas del espacio de entrada, donde haya un número significativo de datos. La regla de los k-vecinos más próximos se puede utilizar para ajustar el valor de los centros. Esta regla clasifica un vector de entrada x, asignándole una etiqueta que lo asocia al centro más próximo a dicho vector de entrada. El análisis de las etiquetas define el movimiento de los centros.

  El algoritmo de los k-vecinos, se describe a continuación:

- Seleccionar k muestras como centros iniciales.
- Iterar mientras no haya cambios significativos
- Asociar cada muestra al centro más cercano.
- Recalcular los centros como el promedio de todos sus elementos.

El resultado de utilizar el algoritmo de los k-vecinos con k=4 puede apreciarse en la siguiente grafica:

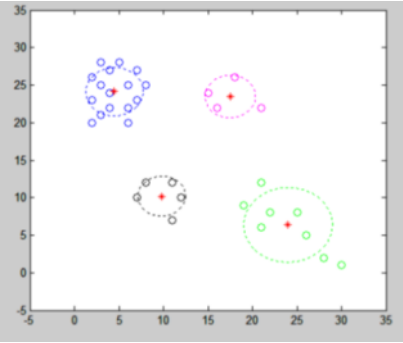

Figura 4. Grafica del algoritmo de los k-vecinos.

- Selección supervisada de centros: Los centros se pueden ajustar utilizando algoritmos supervisados. En este caso los cambios se realizan utilizando métodos como el LMS, descrito anteriormente.

Los citados algoritmos no garantizan la convergencia de la red, ya que E no está linealmente relacionada con la ubicación de todos los centros y, por tanto, se pueden alcanzar mínimos locales. La constante de aprendizaje utilizada para ajustar los centros, no necesita ser la misma que la constante para ajustar los pesos. De forma similar, es posible adaptar la varianza de las funciones de Gauss para maximizar los efectos de los cambios producidos en los centros. Las redes RBF en las que todos sus parámetros se ajustan mediante la utilización de algoritmos de aprendizaje supervisado, gozan de mayor capacidad de generalización [5].

7. REDES NEURONALES ARTIFICIALES Y SERIES DE TIEMPO.

Una serie de tiempo viene determinado por un conjunto de observaciones que están ordenadas en el tiempo, representando el cambio de una variable ya sea de tipo económico, físico, químico, biológico, etc. a lo largo de un período determinado. Por lo general, el conjunto de observaciones disponibles se encuentra almacenado a intervalos de tiempo iguales [5].

El objetivo del análisis de las serie de tiempo es el conocimiento de su patrón de comportamiento, para así poder prever su evolución en el futuro cercano, suponiendo que las condiciones no variarán significativamente. Si bien el comportamiento de cualquier serie de tiempo puede observarse gráficamente, no en todos los casos es posible distinguir las particularidades que cada una puede presentar. Existen ciertos movimientos o variaciones características que pueden medirse y observarse por separado. Estos movimientos son llamados a menudo componentes de una serie de tiempo, y se asume que son causados por fenómenos distintos.

Matemáticamente las series de tiempo están representadas mediante la relación tiempo observación, descrita a través de un conjunto de datos numéricos, como se muestra a continuación [7]:

$$\{y(t_1), y(t_2), \dots, y(t_n)\} = \{y(t): t \in T \subseteq R\} \qquad (11)$$

El objetivo entonces es encontrar un modelo que permita predecir los valores futuros a través de los ya obtenidos, de la siguiente manera:

$$y(t_{n+1}) = f(y(t_n), y(t_{n-1}), \dots) \qquad (12)$$

Este modelo generalmente abstrae las características más importantes de la serie, de tal manera que se puedan obtener las predicciones de la serie para intervalos de tiempo determinados.

En muchas áreas del conocimiento las observaciones de interés son obtenidas en instantes sucesivos del tiempo, por ejemplo, a cada hora, durante 24 horas, mensuales, trimestrales, semestrales o bien registradas por algún equipo en forma continua. De aquí la importancia de este tópico.

A través del tiempo se han desarrollado un gran número de técnicas para la predicción y el modelado de series de tiempo, siendo las redes neuronales artificiales consideradas las más robustas y eficaces para esta tarea.

Para lograr predecir una serie de tiempo mediante una red neuronal artificial, esta debe configurarse de manera que las entradas de la red sean valores pasados al que se desea predecir, tal y como se muestra en la siguiente figura.

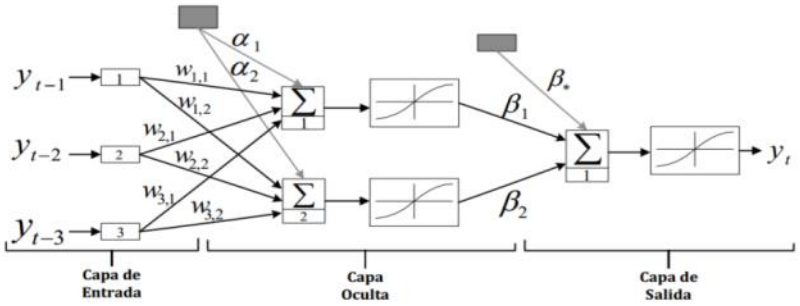

Figura 5. Configuración para una red neuronal aplicada a la predicción de series de tiempo.

El entrenamiento de la red consiste en ajustar los pesos para obtener los valores deseados de la serie en un instante de tiempo específico [1].

8. ARQUITECTURA CLIENTE/SERVIDOR.

La arquitectura cliente-servidor(C/S) es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, que le da respuesta. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora, aunque es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras [8].

En la arquitectura C/S el remitente de una solicitud es conocido como cliente. Sus características son:

- Es quien inicia solicitudes o peticiones, tienen por tanto un papel activo en la comunicación.
- Espera y recibe las respuestas del servidor.
- Por lo general, puede conectarse a varios servidores a la vez.
- Normalmente interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

Al receptor de la solicitud enviada por el cliente se conoce como servidor. Sus características son:

- Al iniciarse esperan a que lleguen las solicitudes de los clientes, desempeñan entonces un papel pasivo en la comunicación.
- Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente.
- Por lo general, aceptan conexiones desde un gran número de clientes (en ciertos casos el número máximo de peticiones puede estar limitado).
- No es frecuente que interactúen directamente con los usuarios finales.

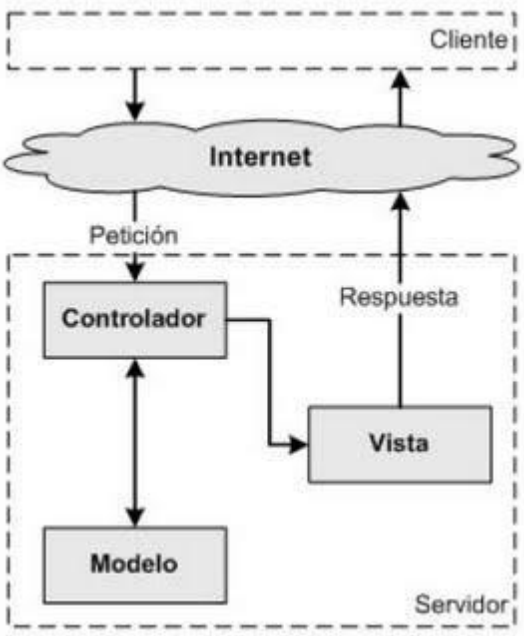

Figura 6. Arquitectura cliente/servidor.

Hoy en día el desarrollo de aplicaciones que utilizan la arquitectura cliente-servidor, generalmente trabajan implementando en este último el patrón de diseño MVC (Modelo, Vista, Controlador), ver figura x, ya que se ha convertido en una guía para el diseño de arquitecturas de aplicaciones y brinda una fuerte interactividad con los usuarios. La intención de implementar este patrón es separar los datos de la aplicación, la interfaz de usuario y la lógica de control.

9. PATRÓN DE DISEÑO MVC.

MVC es un patrón de diseño que fue inicialmente utilizado para construir interfaces de usuario en Smalltalk80 [9].

El patrón MVC (Model-View-Controller) separa el modelado del dominio, la presentación y las acciones basados en las entradas del usuario en tres modelos diferentes:

El modelo: Maneja el comportamiento y los datos del dominio de la aplicación, responde a los requerimientos de información acerca de su estado (usualmente desde la vista) y responde a las instrucciones para cambiar de estado (usualmente desde el controlador).

La vista: son el conjunto de interfaces de usuario que representan los formularios de entrada y salida de información.

El Controlador: Interpreta las acciones del usuario de teclado y ratón, informando al modelo y/o a la vista para cambiar apropiadamente sus estados.

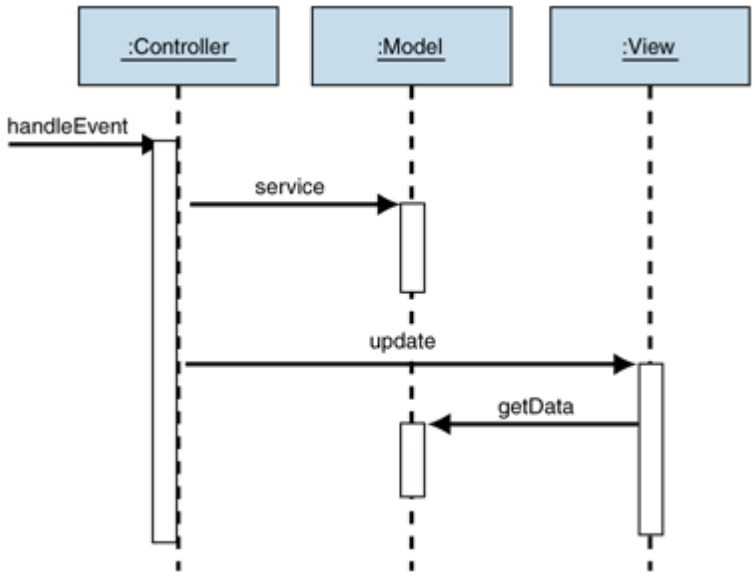

Figura 7. Patrón de diseño Modelo, Vista, Controlador - MVC.

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo de control generalmente es el siguiente:

- El usuario interactúa con la interfaz de alguna manera (ej. presionando un botón, un enlace).
- El controlador recibe (por parte de los objetos de la interfaz vista) la notificación de la acción solicitada por el usuario.
- El controlador accede al modelo, posiblemente actualizando los datos enviados por el usuario.
- El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario.
- La vista usa el modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo.
- En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los da tos del modelo a la vista.
- La interfaz espera por nuevas interacciones de usuario para iniciar nuevamente el ciclo.

10. HERRAMIENTAS INFORMÁTICAS

10.1 Lenguaje de Programación Python.

Python es un lenguaje de programación de propósito general es decir que puede ser utilizado para varios propósitos ya sea web, bases de datos, cálculos, etc. Python se caracteriza por tener una sintaxis limpia y relativamente fácil de entender en comparación con otros lenguajes de programación, como por ejemplo C++.

Es distribuido bajo la licencia FLOSS (Free/Libre and Open Source Software), lo que indica que es posible distribuir copias del software, manipular el código fuente y realizar modificaciones a este y generar nuevas versiones que contribuyan a la comunidad.

Python es un lenguaje de programación de alto nivel, esto indica que cuando se escriben programas en python nunca se debe preocupar por detalles de bajo nivel, como manejar la memoria empleada por tu programa, etc. Además de esto gracias a su naturaleza de ser open source ha sido manipulado para hacerlo funcionar en diversos sistemas operativos, ventaja importante ya que todos los programas escritos en python trabajaran en cualquier plataforma sin requerir cambio alguno.

Otra característica importante de Python, es que es interpretado, esto indica que no existen compilaciones separadas y pasos de ejecución. Solo se ejecuta el programa desde el código fuente. Internamente, Python convierte el código fuente en una forma intermedia llamada bytecodes, después los traduce en el lenguaje nativo de tu computadora y ejecuta. Todo esto hace el uso de Python mucho más sencillo. Solo se deben ejecutar los scripts, sin la necesidad de enlazar y cargar librerías, etc. Esto lo convierte en portable, ya que solo se necesita copiar el código del programa Python en cualquier otro sistema y trabajará igualmente.

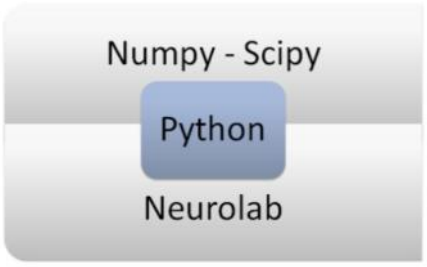

Figura 8.Módulos de python para desarrollo científico.

Hay que destacar que python es un lenguaje multiparadigma, ya que permite trabajar con diversos paradigmas de programación, como lo son: el estructural, orientado a objetos y funcional, convirtiéndolo en una herramienta bastante versátil para el desarrollo de aplicaciones en cualquier campo de las ciencias computacionales. Por ejemplo, numpy es módulo de alto nivel basado en C, bien optimizado, para trabajar con vectores y matrices, scipy es un conjunto de herramientas para computación científica, que permite trabajar con algebra linel, procesamiento de datos, entre otras. Para las redes neuronales artificiales se cuenta con un módulo llamado neurolab, con el que es posible trabajar con distintos tipos de redes neuronales como el mlp, hopfield, entre otros.

Esto lo convierte en una herramienta muy poderosa frente a otros lenguajes como java o c#.

10.2 Framework Javascript ExtJs.

ExtJs es un framework cross-browser RIA, facil de utilizar para la construcción de interfaces gráficas de usuario ricas, utilizado por millones de desarrolladores web en el mundo.

Fue desarrollado a principios de 2006 por Jack Slocum, quien comenzó a trabajar en un conjunto de empresas de servicios públicos de extensión de la interfaz de usuario de Yahoo! (YUI) de la biblioteca. A finales del año, había ganado tanto en popularidad que el nombre fue cambiado a Ext JS, un reflejo de su madurez y su independencia como un framework. La compañía se formó a principios de 2007, y Ext JS es ahora es doble licencia

bajo GPL y una licencia comercial. Sencha es la compañía que fabrica Ext JS (y otros frameworks que se incluyen ext GWT y Touch Sencha).

El nombre de la empresa solía ser el mismo que el framework popular, por lo tanto, el nombre de la compañía se cambió a Sencha en 2010.

Ext Js es Muy Potente en el diseño de paneles, grids, layouts, tooltips, pestañas, control de formularios, menús, check box, radio button, arboles, barra de herramientas, combobox, campos de fecha, y muchos widget más, lo cual está ampliamente documentado. Utilizada por varias entidades de las cuales podemos citar algunos ejemplos: Adobe, Amazon.com, Boeing, Borland, Cisco, CNN, Down Jones & Co, HP, IBM. Ext facilita mucho el trabajo a los desarrolladores, pues en unas cuantas líneas de código en javascript es posible hacer interfaces realmente elegantes.

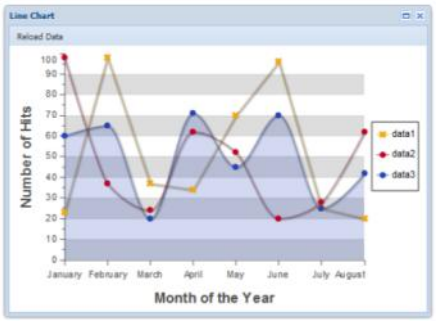

Figura 9. Grafica con ExtJs

Figura 10. Grilla con ExtJs

Una de las grandes ventajas de utilizar ExtJS es que permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de layouts similar al que provee Java Swing, gracias a esto provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno (Firefox, IE, Safari, etc.).

Existe un balance entre Cliente – Servidor. La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.

La fuerza base de Ext-JS es el diseño de componentes. La fácil ampliación de los componentes predeterminados para satisfacer cualquier necesidad, y las extensiones que se encapsularán en esos componentes. Como resultado, los equipos de desarrollo pueden desarrollar incluso las más grandes de las aplicaciones sin tener que pisar el código de otro.

Comunicación asíncrona. En este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se dé cuenta.

Eficiencia de la red. El tráfico de red puede disminuir al permitir que la aplicación elija que información desea transmitir al servidor y viceversa, sin embargo la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico.

En su última versión, la 4.1.1, Ext-Js, implementa el patrón de diseño MVC, donde el modelo es una colección de campos y sus datos. Los modelos se hacen persistentes a través del paquete de datos y pueden ser vinculados a otros modelos a través de asociaciones. Las vistas son cualquier tipo de componente, como por ejemplo, grillas, paneles, arboles. Los controladores ocupan un lugar especial en la aplicación, son los encargados de interactuar con el modelo y las vistas cuando estos son inicializados y renderizados respectivamente.

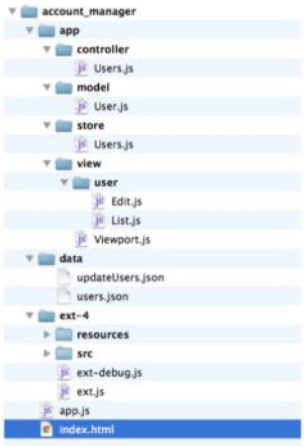

Figura 11. Estructura de una aplicación ExtJs utilizando MVC

10.3 Integración Python y ExtJs

La forma en que se comunican ExtJs y Python es a través del formato JSON (JavaScript Object Notation), permitiendo enviar datos en ambas direcciones entre las dos herramientas. JSON está constituido por dos estructuras: una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo. Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

Además del formato JSON se requiere establecer activo en el web server el CGI (Common Gateway Interface). CGI es un conjunto de normas que establecen como se intercambia la información entre el servidor web y un script personalizado.

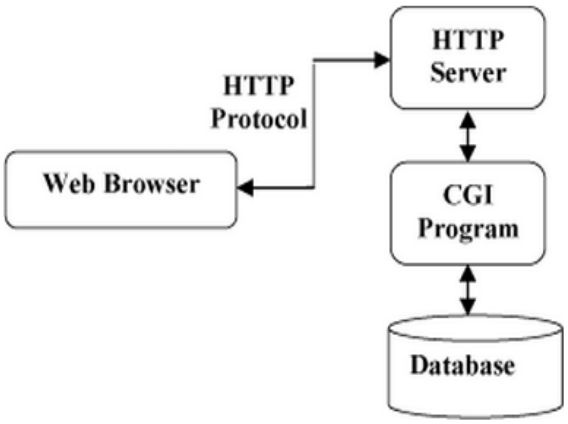

Figura 12. Arquitectura de CGI.

Cuando CGI está activo en el web server posibilita que los archivos que son solicitados a través del protocolo HTTP mediante un navegador web sean ejecutados como programas, y los resultados generados por este se envíen de vuelta al navegador para su visualización.

Los programas CGI generalmente son llamados scripts y pueden estar escritos en lenguajes como python, php, java, c, c++, etc.

11. ESTADO DEL ARTE.

En las últimas décadas, la aplicación de las redes neuronales artificiales en la predicción de series de tiempo ha ido creciendo por las características ideales que ofrecen las RNA para trabajar con modelos no lineales. Así mismo, el desarrollo de aplicaciones que faciliten el trabajo a la hora de realizar las simulaciones con rendes neuronales artificiales continua en aumento.

Vasconcelos, Ribeiro y Choren destacan las redes neuronales artificiales como uno de los métodos de predicción para series temporales gracias a su gran capacidad de adaptación y capacidad de representación de procesos no lineales [2].

Taib, Mohd , en sus tesis de maestría titulada Time series modelling and prediction using neural networks, se demuestra el uso de algoritmos combinados con un modelo hibrido de funciones de base radial para la modelación y pronóstico de series de tiempo no lineales, obteniendo resultados interesantes en cuanto a eficacia y eficiencia utilizando este tipo de red neuronal [10]. Koskela, Lehtokangas, Saarinen y Kaski, realizan un estudio teórico y experimental de diferentes tipos de RNA como el multilayer perceptron, Fir neural network y elman neural network. Destacan las ventajas y desventajas de cada uno de estos en cuanto al dominio del pronóstico y predicción de series de tiempo. De acuerdo a los resultados experimentales obtenidos es necesario realizar una buena configuración de la red para lograr una buena aproximación a los datos reales, dependiendo del tipo de RNA que se esté utilizando [11]. Saranli y Baykal presentan las redes tipo RBF para la predicción de series de tiempo caóticas utilizando como algoritmo de aprendizaje el algoritmo relocating-lms, resaltando la consecución de un error mínimo al comparar los datos reales de la serie de tiempo con los predichos por la red [12].

Fillipeto Maria en su tesis doctoral presenta una nueva técnica para la predicción no lineal de series de tiempo a través de las redes neuronales de funciones de base radial con

atribución de centros gaussianos de las funciones de base radial por descomposición de los espacios de datos en subespacios [13].

Fajardo, Fdez-Riverola, y Soto utilizan las RBF para la predicción del caudal de los ríos, en este se destaca el proceso de aprendizaje al que es sometido la red neuronal y los resultados con las distintas configuraciones utilizadas, demostrando las variaciones que esto puede generar en las simulaciones de este tipo de problemas. Además en Water flows modelling and forecasting using a RBF neural network, Fajardo, Gonzales, Fdez-Riverola, y Soto se demuestra la eficacia de las RBF en la predicción de series de tiempo [14].

Fajardo, en su tesis doctoral desarrolla un sistema hibrido inteligente para la predicción y el pronóstico de caudales en el cual utiliza las redes neuronales como una de las herramientas inteligentes que ayudan al sistema al desarrollo de la tarea objetivo [5].

Villa Garzon, en su tesis de maestria utiliza las redes casacada correlacion para el modelado y predicción del precio de la electricidad en mercados de corto plazo liberalizados, tomando como punto de partida y de comparativa el multilayer perceptron con algoritmo de aprendizaje resilient backpropagation [1].

Basados en la red tipo MLP, Velásquez y Franco, proponen una nueva versión no lineal del modelo airline; reemplazando la componente lineal de promedios móviles por un perceptrón multicapa con el objetivo de realizar pronósticos de series de tiempo con tendencia y ciclo estacional [15].

Velásquez, Dyner y Souza utilizan un modelo que combina un modelo lineal autorregresivo (AR) con un perceptrón multicapa (MLP) con una única capa oculta, permitiéndole unir las ventajas de los modelos autorregresivos y de las redes neuronales, de tal forma que es más fácil capturar dinámicas complejas, tal como es el caso de los precios de electricidad. Esto para realizar el modelado del precio spot de la electricidad en Brasil [16].

En cuanto al desarrollo de aplicaciones que involucren redes neuronales artificiales y las series temporales, Velásquez, Zambrano y Vélez, desarrollan ARNN: un paquete para la predicción de series de tiempo usando redes neuronales autorregresivas, basado en el

lenguaje de programación R. implementando como funciones principales la creación y estimación del modelo de RNA y el pronóstico de la serie temporal [17].

Para el caso de plataformas web que trabajen con redes neuronales se destaca el desarrollo realizado por Watta, Hassoun, y Dannug de applets que permiten simular varios tipos de redes neuronales. Además, Manic, Wilamowski y Malinowski desarrollan una herramienta web para la simulación de redes neuronales artificiales con objetivo educativo, resaltando características importantes de este tipo de proyectos como lo son: la accesibilidad con respecto a software y hardware y la facilidad en el despliegue de la plataforma, olvidándose de los obstáculos de la instalación [18]. Hay que resaltar que esta plataforma muestra una interfaz de usuario poco elegante.

Es evidente la importancia de las series de tiempo en las ramas del conocimiento humano en donde las RNA juegan un papel central al permitir realizar pronósticos a las series temporales. El desarrollo de aplicaciones que ayuden a optimizar el proceso de simulación de estas toma mucha relevancia en especial cuando los proyectos desarrollados son implementados en la internet.

El presente proyecto pretende realizar una implementación web para redes neuronales artificiales aplicadas a la predicción de series de tiempo, utilizando herramientas de última generación que permiten construir interfaces de usuario muy elegantes y excelentes resultados en procesamiento de datos. Por otra parte, se desea que el usuario final no deba preocuparse por los detalles de los elementos internos de la implementación, sino ocuparse de la definición de la arquitectura de las RNA implementadas.

12. DISEÑO METODOLÓGICO.

Para lograr el objetivo del proyecto se plantea una metodología basada en fases que contempla las siguientes etapas:

Fase 1. Exploración y selección de los diferentes tipos de redes neuronales artificiales a ser implementadas.

En esta fase se realiza un estudio bibliográfico de los diferentes tipos de redes neuronales artificiales aplicadas al tratamiento de las series de tiempo, para lo cual se consulta con expertos, revistas científicas, artículos web, entre otras fuentes. Posteriormente se establece un cuadro comparativo de cada una y se seleccionan los modelos de RNA más adecuados al dominio planteado.

Fase 2. Diseño de la plataforma.

En esta fase se toma la ingeniería de software como referente para el establecimiento de diseños y diagramas que representen y abstraigan las características más importante de los tipos de redes neuronales artificiales seleccionados en la fase anterior. Inicialmente se establecen los requisitos que debe cumplir el sistema, para luego tomarlos como base y plasmar en diagramas UML la representación del sistema.

Fase 3. Desarrollo e implementación de la plataforma.

Luego de establecer las características del sistema, se procede con la transformación de los diagramas en código fuente escrito en python, para el procesamiento de la lógica del negocio en el servidor y javascript, utilizando el framework ExtJs para las interfaces gráficas de usuario que tendrá acceso el cliente, teniendo en cuenta la

arquitectura cliente/servidor en que estará basada la plataforma. Finalmente se lleva a cabo el plan de pruebas del sistema e inicia la puesta en marcha.

Fase 4. Socialización del proyecto.

Esta etapa pretende dar a conocer ante la comunidad académica y científica de la ingeniería de sistemas y computación los resultados obtenidos con el desarrollo del proyecto, participando en diversos programas de divulgación científica.

13. DISEÑO DE LA PLATAFORMA.

13.1 Requisitos funcionales:

La plataforma web para redes neuronales artificiales aplicadas a la predicción de series de tiempo debe permitir lo siguiente:

- Importar datos en formato de archivo delimitado por comas (.csv).
- Configurar las variables para la fase de entrenamiento de la red neuronal.
- Entrenar la red neuronal con la configuración establecida y mostrar los resultados en grillas.
- Probar un conjunto de datos de test y realizar la simulación.
- Graficar los resultados obtenidos en la fase de entrenamiento y de test de la red neuronal.
- Graficar el error generado por la red neuronal respecto a los datos de test enviados.
- Exportar a Excel o csv los resultados obtenidos en la simulación de la red neuronal.
- Acceso a diferente usuario desde un navegador web.

13.2 Requisitos no funcionales:

- El diseño de la plataforma debe ser orientado a objetos.
- Debe utilizar el patrón MVC.
- Utilizar python como lenguaje de servidor y ExtJs como framework javascript para el desarrollo de las interfaces gráficas.

- Utilizar la arquitectura cliente/servidor.

13.3 Casos de Uso

Actor	Código	Casos de Uso
Usuario	CU01	Importar datos
	CU02	Entrenar Red Neuronal Artificial
	CU03	Testear Red Neuronal Artificial
	CU04	Graficar resultados
	CU05	Exportar Resultados

Tabla 3. Descripción casos de uso.

Nombre	**Importar datos**	**Código**	**CU01**
Versión	1.3	**Frecuencia**	Alta
Fecha Ult.Act.	15-06-2012	**Prioridad**	Alta
Actor(es) Primario(s)	Usuario		
Nivel	Sistema		
Objetivos	• Importar los datos que utilizará al red neuronal artificial para la fase de entrenamiento.		
Descripción	Importa los datos en formato csv que utiliza la red neuronal artificial seleccionada para su proceso de entrenamiento.		
Trigger	Usuario		

Importar datos			
Flujo normal o básico			
#	**Actor**	**#**	**Sistema**
1.1	Ingresa en la plataforma web.	1.2	Muestra la interfaz inicial de usuario
2.1	Selecciona el tipo de red neuronal con la que trabajará.	2.2	Despliega los formularios para la carga de datos en el formato csv.
3.1	Ingresa los datos requeridos.	3.2	Valida los datos ingresados
		3.3	Muestra los datos que fueron cargados.
Post condiciones	• Genera una grilla con los datos que fueron cargados en formato csv.		

Tabla 4. Caso de uso CU01.

Nombre	**Entrenar Red Neuronal Artificial**	**Código**	**CU02**
Versión	1.1	**Frecuencia**	Alta
Fecha Ult.Act.	16-06-2012	**Prioridad**	Alta
Actor(es) Primario(s)	Usuario		
Nivel	Sistema		
Objetivos	• Entrenar la red neuronal artificial a partir de los datos que fueron cargados.		
Descripción	Realiza el proceso de entrenamiento de la red neuronal a partir de los datos que fueron cargados y los parámetros establecidos.		
Trigger	Usuario		

Entrenar Red Neuronal Artificial			
Flujo normal o básico			
#	**Actor**	**#**	**Sistema**
1.1	Selecciona el tipo de red neuronal que entrenará.	1.2	Despliega un formulario para recopilar los datos de configuración del proceso de entrenamiento de la

			red.
2.1	Ingresa los datos requeridos.	2.2	Valida los datos que fueron ingresados.
		2.3	Ejecuta el proceso de entrenamiento de la red.
		2.4	Muestra los datos obtenidos en la fase de entrenamiento.
Post condiciones	• Muestra los datos organizados en grillas de la fase de entrenamiento de la red neuronal artificial seleccionada.		

Tabla 5. Caso de uso CU02.

Nombre	**Testear Red Neuronal Artificial**	**Código**	**CU03**
Versión	1.2	**Frecuencia**	Alta
Fecha Ult.Act.	16-06-2012	**Prioridad**	Alta
Actor(es) Primario(s)	Usuario		
Nivel	Sistema		
Objetivos	• Testear la red neuronal artificial a partir de los resultados obtenidos en la fase de entrenamiento de la red.		
Descripción	Realiza el proceso de simulación de la red neuronal a partir de los resultados obtenidos en la fase de entrenamiento de la red.		
Trigger	Usuario		

Testear Red Neuronal Artificial			
Flujo normal o básico			
#	**Actor**	**#**	**Sistema**
1.1	Selecciona el tipo de red neuronal que testeará.	1.2	Despliega un formulario para recopilar los datos con los que se realizará el proceso de test.
2.1	Ingresa los datos requeridos.	2.2	Valida los datos que fueron ingresados.
		2.3	Ejecuta el proceso de simulación de la red.
		2.4	Muestra los resultados obtenidos en formato de grilla.
Post condiciones	• Muestra los resultados obtenidos en la simulación en formato de grillas para una mejor visualización por parte del usuario.		

Tabla 6. Caso de uso CU03.

Nombre	**Graficar Resultados**	**Código**	**CU04**
Versión	1.2	**Frecuencia**	Alta
Fecha Ult.Act.	16-06-2012	**Prioridad**	Alta
Actor(es) Primario(s)	Usuario		
Nivel	Sistema		
Objetivos	• Representar en forma gráfica los resultados obtenidos en las fases de entrenamiento y/o test.		
Descripción	Realiza una representación gráfica de los resultados que se obtuvieron el fase de entrenamiento y/o test de la red neuronal.		
Trigger	Usuario		

Graficar Resultados			
Flujo normal o básico			
#	**Actor**	**#**	**Sistema**
1.1	Selecciona la opción graficar resultados	1.2	Despliega las opciones disponibles para gráficas
2.1	Selecciona la opción que desea graficar	2.2	Organiza los datos en vectores x y y para representarlos en el plano cartesiano.
		2.3	Muestra la gráfica en una nueva ventana.

Post condiciones	• Muestra los resultados obtenidos en la simulación en formato de grillas para una mejor visualización por parte del usuario.

Tabla 7. Caso de uso CU04.

Nombre	**Exportar Resultados**	**Código**	**CU05**
Versión	1.2	**Frecuencia**	Alta
Fecha Ult.Act.	16-06-2012	**Prioridad**	Alta
Actor(es) Primario(s)	Usuario		
Nivel	Sistema		
Objetivos	• Exportar los resultados obtenidos en la simulación de la red neuronal.		
Descripción	Exporta los resultados obtenidos en formato Excel o csv de las fases de entrenamiento y test de la red neuronal seleccionada.		
Trigger	Usuario		
Exportar Resultados			
Flujo normal o básico			
#	**Actor**	**#**	**Sistema**
1.1	Selecciona la opción exportar resultados	1.2	Muestra los datos a exportar
		1.3	Genera un archivo en Excel con los resultados de las fases de entrenamiento y test de la red neuronal seleccionada.
Post condiciones	• Genera un archivo con los resultados obtenidos en las fases de entrenamiento y test de la red neuronal seleccionada para utilidad del usuario.		

Tabla 8. Caso de uso CU05.

13.4 Diagrama de Casos de Uso.

A continuación se presenta el diagrama de casos de uso basado en la descripción anterior.

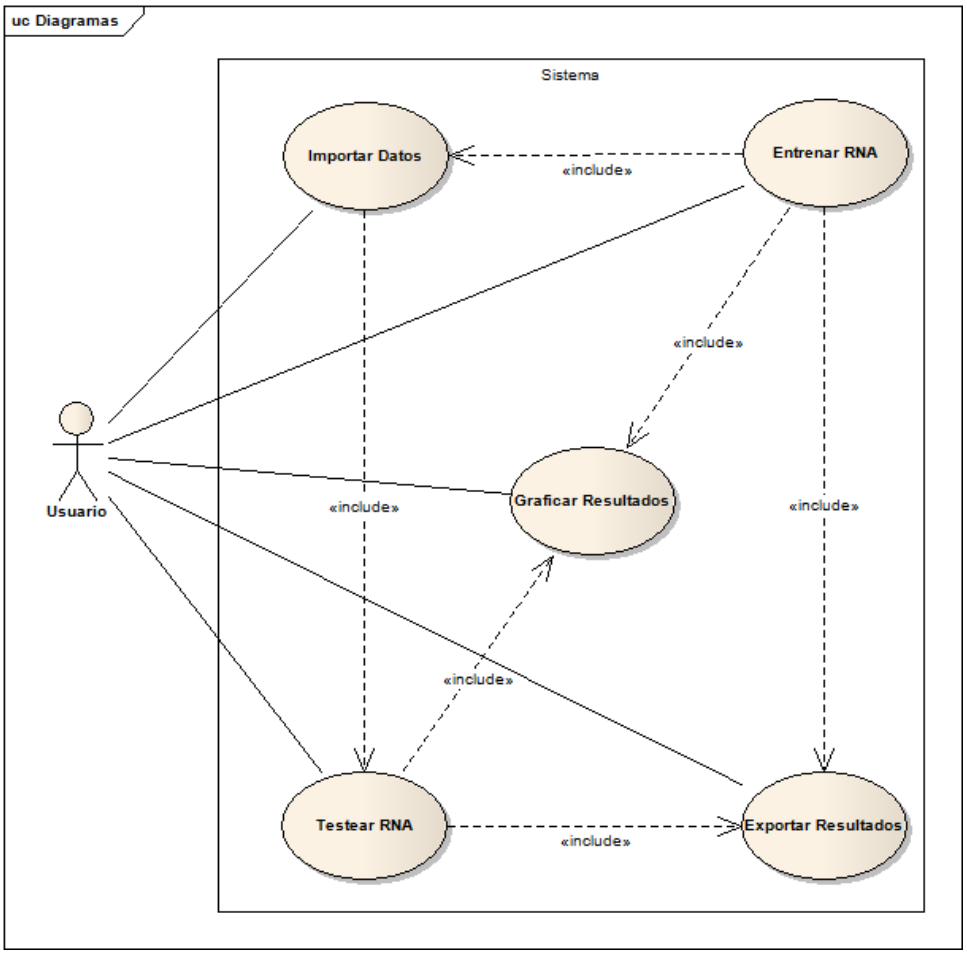

Figura 13. Diagrama de casos de uso.

13.5 Diagrama de Clases.

En la siguiente figura se puede observar el diagrama de clases utilizado en el lenguaje de programación Python, el cual se encarga de procesar los datos y aplicar los algoritmos de aprendizaje de las RNA. Para el caso del Perceptrón Multicapa se utiliza el paquete Neurolab, que facilita el trabajo con este tipo de red neuronal al tener funciones implementadas y probadas.

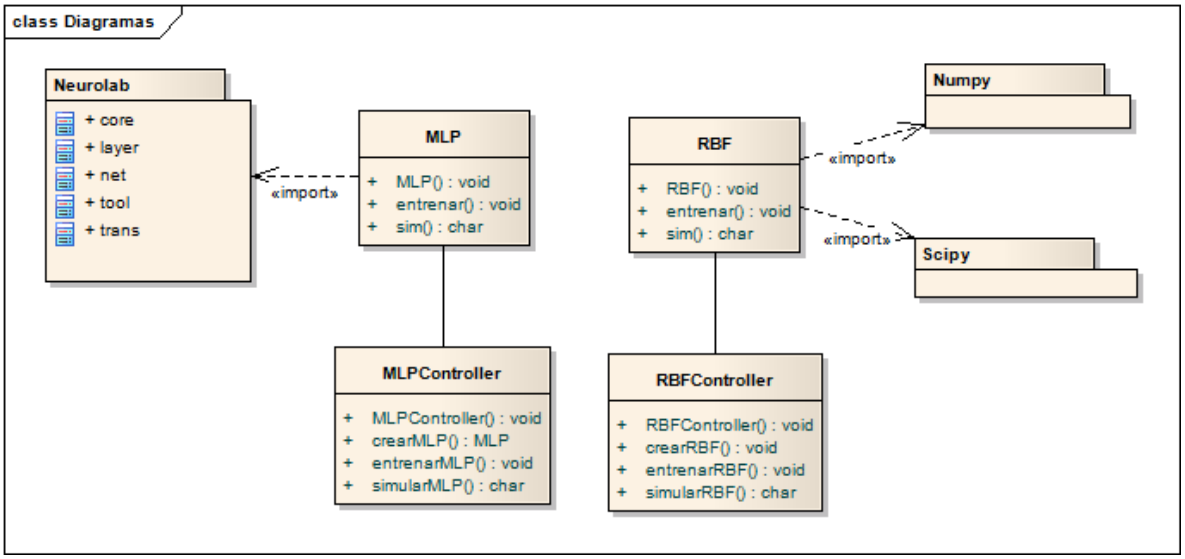

Figura 14. Diagrama de clases utilizadas en Python.

En la siguiente gráfica se muestra el diagrama de clases basado en el framework ExtJs.

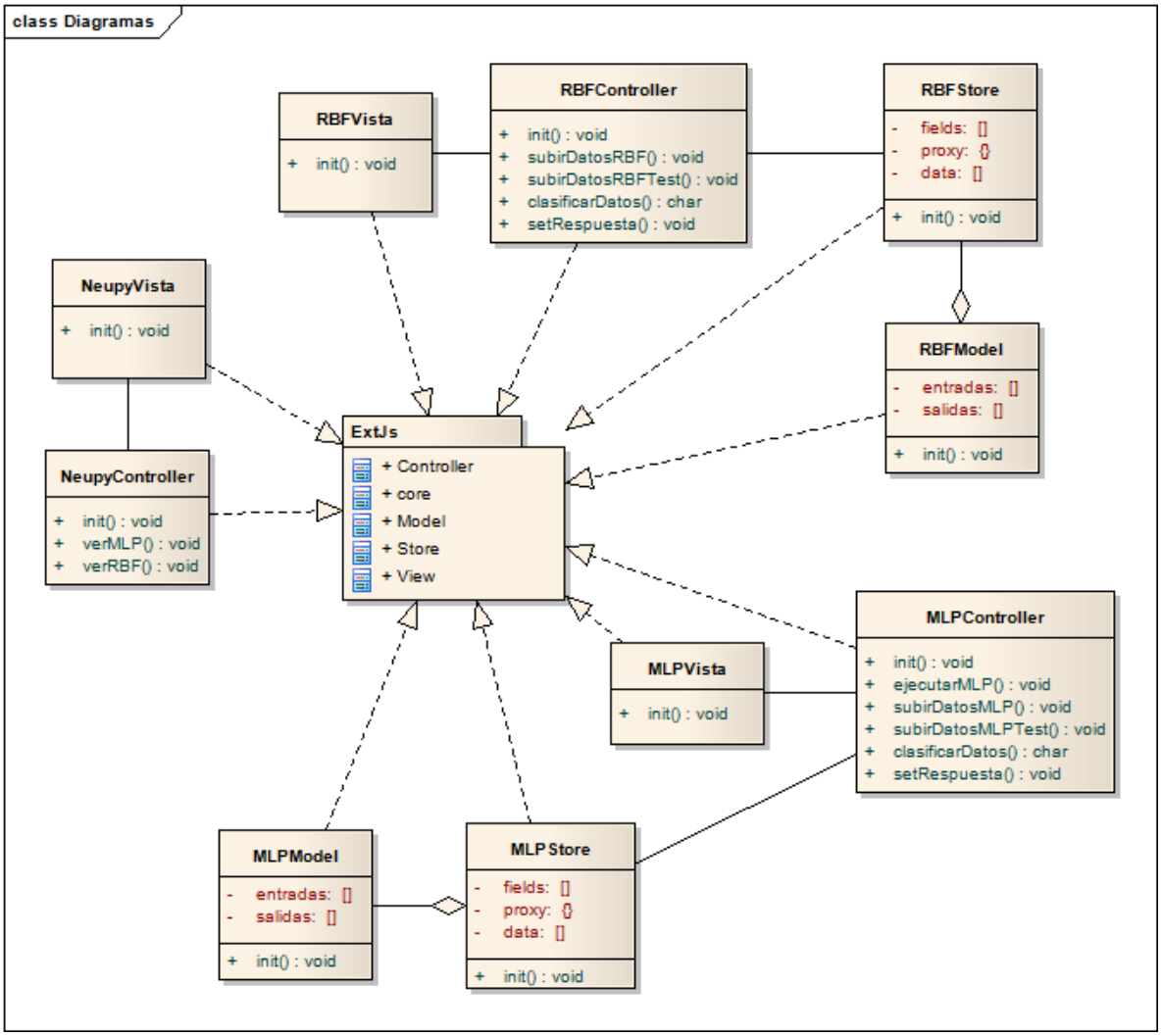

Figura 15. Diagrama de clases utilizado en el framework ExtJs.

13.6 Diagrama de Despliegue.

A continuación se muestra el diagrama de despliegue de la plataforma. Como se puede observar en el cliente es necesario el uso de un navegador con javascript habilitado. En el servidor se debe instalar un web server, por ejemplo: Apache, con cgi habilitado para ejecutar los scripts python.

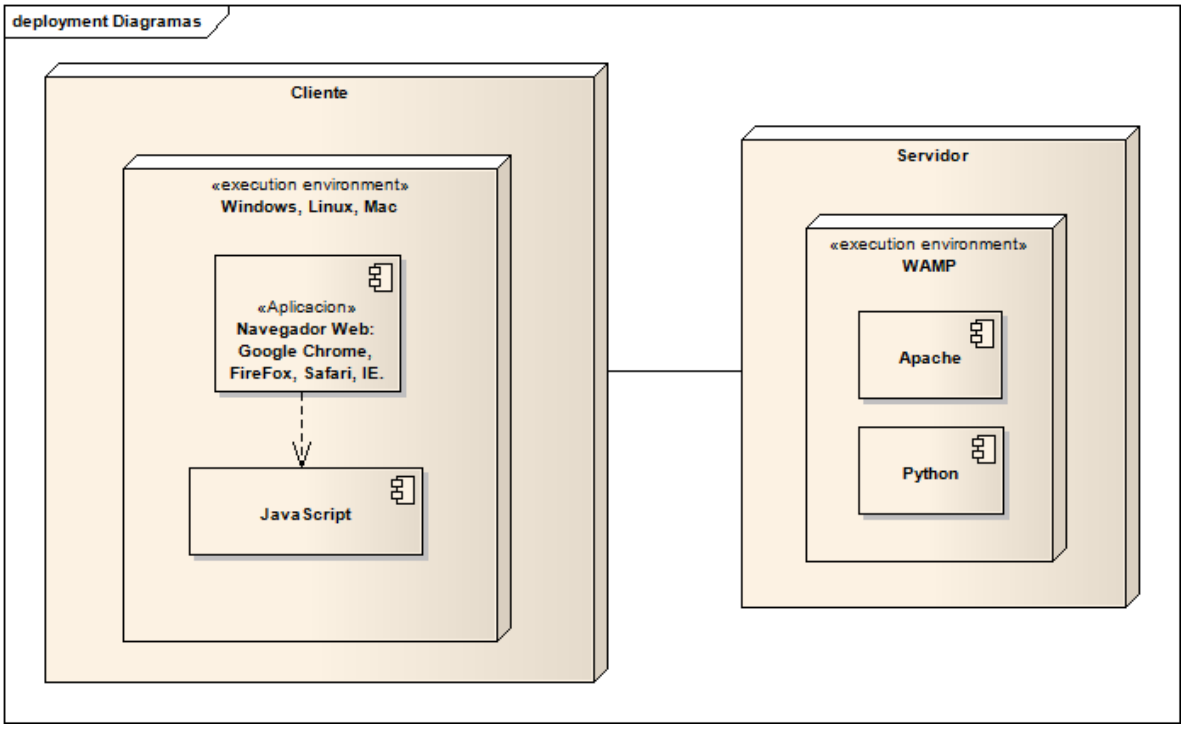

Figura 16. Diagrama de despliegue de la plataforma.

14. RESULTADOS.

Con el desarrollo de este proyecto se logra la generación de nuevo conocimiento y desarrollo tecnológico. Es un producto para ser utilizado en ámbitos investigativos que estén relacionados con las series de tiempo y las redes neuronales artificiales. Además de esto es fácil de escalar, innovador y con capacidad de adaptación simple.

La plataforma para redes neuronales artificiales aplicadas a la predicción de series de tiempo está escrita en el lenguaje de programación Python como base para procesamiento de los datos con los algoritmos de las RNA. Para las interfaces gráficas de usuario se utiliza ExtJs, el cual permite generar aplicaciones bastante elegantes en cuanto a la presentación de los datos al usuario.

Cuenta con una ventana principal en la que es posible seleccionar el tipo de red neuronal con la que se desea trabajar, otorgándose dos opciones: MLP y RBF. MLP corresponde al tipo de red neuronal perceptron multicapa, descrito en secciones anteriores. RBF permite trabajar con redes de funciones de base radial.

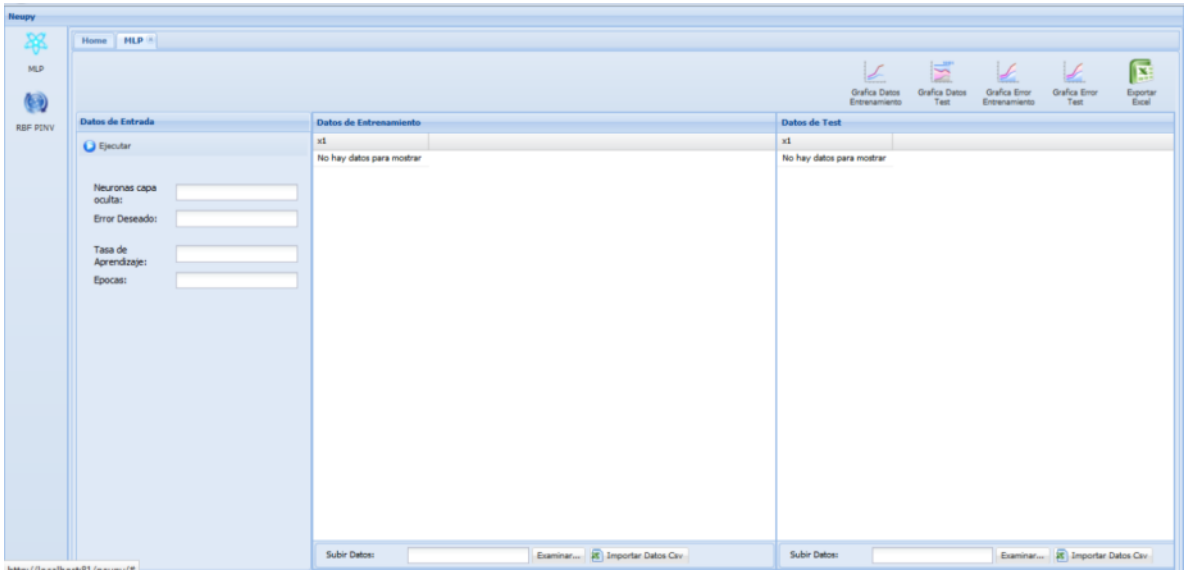

Figura 17. Ventana principal de la plataforma.

Una vez ha sido seleccionada el tipo de red neuronal que se simulará se cargan las opciones de configuracion disponibles para el usuario, hay que resaltar que se pretende que este se centre en la arquitectura de la red y no en ciertas configuraciones que demandan un tiempo considerable. Para facilitar esta tarea se dispone de archivos de configuración en los cuales están fijos los parámetros como por ejemplo, las funciones utilizadas en cada una de las capas del MLP, el algoritmo de aprendizaje utilizado, el cual está parametrizado para utilizar el resilient backpropagation.

Los datos de entrada de la red neuronal son importados a la aplicación a través de archivos en formato csv del cual se abstrae el número de entradas, los datos de aprendizaje y test de la red.

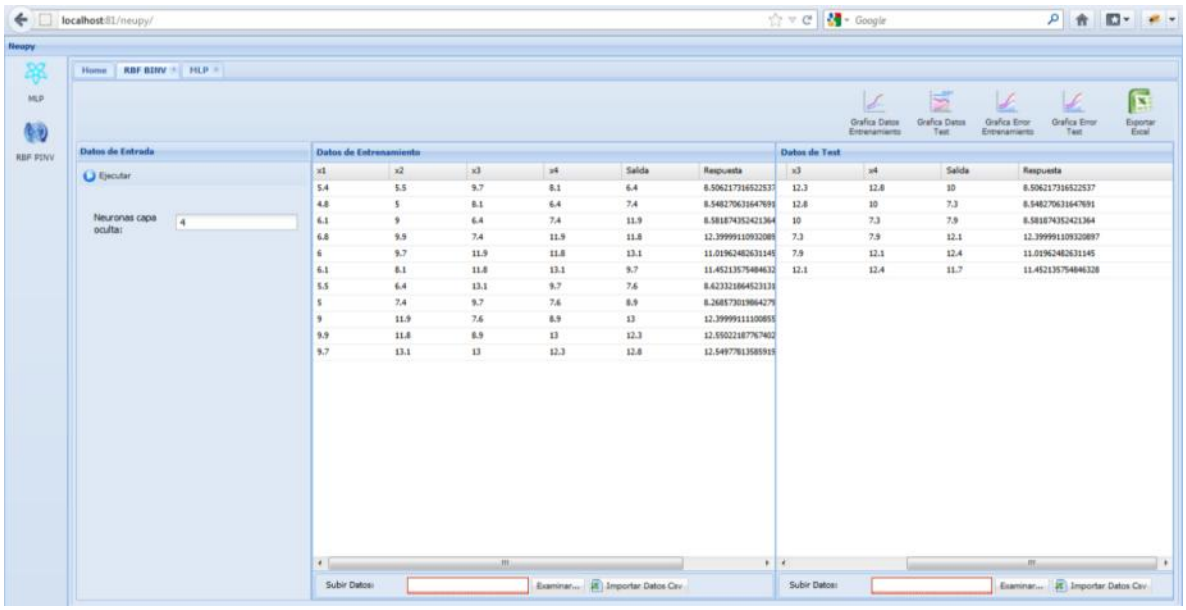

Figura 18. Importación de datos a la plataforma a través de archivos csv.

Posteriormente, es posible realizar la simulación de la red utilizando la opción ejecutar, la cual se encargará de realizar una petición asíncrona al servidor para obtener las respuestas de la red neuronal para los datos de entrenamiento y test. Estos datos son enviados codificados en formato JSON, los scipts escripts escritos en python se encargan de decodificarlos y realizar ejecutar las funciones correspondientes para el proceso de entrenamiento y test de la red.

Inmediatamente son obtenidos los resultados se tiene la opción de graficar los datos obtenidos por la red en las fases de entrenamiento y test. Además de estas, es posible representar el error obtenido en las mismas fases.

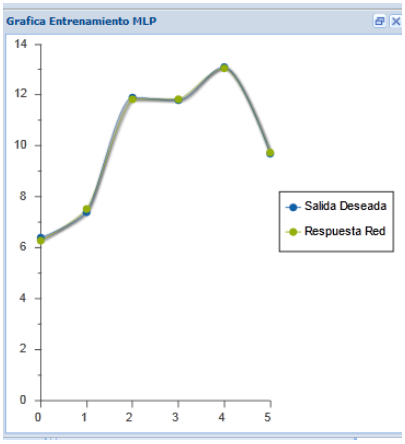

Figura 19. Grafica de la salida deseada y la salida de la red en la fase de entrenamiento.

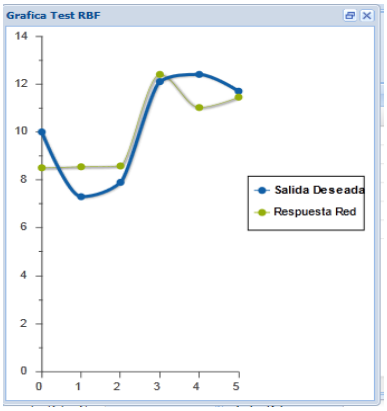

Figura 20. Grafica de la salida de la red y la salida deseada en la fase de test.

Para el caso de las redes tipo RBF, el proceso es similar, importar los datos a través de los archivos csv configurar los parámetros de la red y ejecutar para obtener los resultados. Para las rbf se utiliza el algoritmo de los k-vecinos para obtener los centros de las funciones de base radial y el algoritmo LMS para calcular las varianzas, posterior a esto con la regla de la pseudo-inversa se ajustan los pesos de la red.

En cuanto a las aproximaciones obtenidas, se probaron conjuntos de datos pequeños debido a las características hardware con las que se cuenta para el desarrollo del proyecto,

obteniéndose buenos resultados que permiten determinar si la variable que se esta prediciendo aumentará y disminuirá su valor en el futuro. Hay que tener en cuenta que los pronósticos son aproximaciones, es decir, no se garantiza que el error sea del 0%.

15. CONCLUSIONES.

En el presente trabajo se ha presentado el desarrollo de una plataforma web para redes neuronales artificiales aplicadas a la predicción de series temporales. Basada en tecnologías de última generación que permiten generar aplicaciones web con interfaces gráficas de usuario bastante elegantes, como es el caso del framework ExtJs, y por otra parte un lenguaje de programación de alto nivel en crecimiento y utilizado por muchos expertos en el área de la computación científica tanto en aulas de clase como para el desarrollo de proyectos experimentales, como lo es python. A través de la combinación de estas dos herramientas tecnológicas se ha logrado el objetivo del presente trabajo, gracias a que se logra pronosticar una serie de tiempo con buenos resultados de aproximación. Sin embargo hay que destacar que el hardware requerido para la simulación de este tipo de procesos suele ser bastante potentes. En este proyecto se utilizó una máquina con características de un usuario general, lo cual repercute en los tiempos de respuesta y la expiración de procesos cuando se realizan peticiones al servidor. Por tal motivo lo recomendable es utilizar un equipo con características avanzadas de procesamiento, otra alternativa es el uso de la computación en paralelo, funcionalidad que adapta perfectamente a las redes neuronales artificiales. Por otra parte se apoya el proceso de una tesis doctoral facilitando el trabajo del estudiante de doctorado en cuanto a la realización de pruebas y comparaciones entre distintos modelos de redes neuronales aplicados al dominio de la predicción de series temporales.

16. TRABAJO FUTURO.

Como punto final, es necesario indicar el trabajo que se continuará realizando con la plataforma desarrollada. Se pretende ampliar el número de tipos de redes neuronales artificiales para simular. Además de esto, incluir más opciones en cuanto a los algoritmos de aprendizaje para cada tipo de RNA implementada, en este aspecto se estudiará la posibilidad de incluir algoritmos genéticos.

17. BIBLIOGRAFÍA.

[1]. Modelado y Predicción del Precio de la Electricidad en Mercados de Corto Plazo Liberalizados Usando Redes Cascada Correlación. Fernán Alonso Villa Garzón. Tesis de Maestría (Msc. Ingeniería de Sistemas y Computación). 2011. Universidad Nacional de Colombia. Facultad de Minas. Maestría en Ingeniería de Sistemas.

[2]. Métodos para Previsão de Séries Temporais e suas Tendências de Desenvolvimento. Claudio Vasconcelos Ribeiro, Ronaldo Ribeiro Goldschmidt, Ricardo Choren. 2009. Institulo Militar de Engenharia. ISSN 1982-9035. Seção de Engenharia de Computação.

[3]. Integración Numérica con Redes Neuronales. Iván Christhofer Chaman García. Tesis de Grado (Ingeniero en Ciencias de la Computación). BeneméritaUniversidad Autónoma de Puebla. Facultad de Ciencias de la Computación. 2010.

[4]. Redes Neuronales Artificiales aplicadas al Análisis de Datos. Juan José Montaño Moreno. 2002. Universitat de les Illes Balears. Facultad de Psicología

[5]. Sistema Inteligente para la Estimación y Pronóstico de Caudales. Carlos Hernán Fajardo Toro. 2008. Tesis Doctoral. Universidad de Vigo. Departamento de Informática.

[6]. Desarrollo y Optimización de Nuevos Modelos de Redes Neuronales Basadas en Funciones de Base Radial. 2005. Tesis Doctoral. Universidad de Granada. Departamento de Arquitectura y Tecnología de Computadores.

[7]. Introducción al Análisis Clásico de Series de Tiempo. Arellano M. 2001. [En Línea]. http://ciberconta.unizar.es/LECCION/seriest/100.HTM.

[8]. Aplicación web para la monitorización en el espacio tiempo de enfermedades. Ivan Marques Marques. Tesis de grado (Trabajo Final de Carrera). 2012. Universitat Oberta de Catalunya.

[9]. Patrón de Diseño MVC. José Luis Jurado. [En Línea]. http://pis.unicauca.edu.co/moodle/file.php/291/Patron_Diseno_MVC.pdf

[10]. Time Series Modelling and Prediction Using Neural Networks. Mohd Nasir Taib.1993. Masters thesis. Universiti Teknologi Mara.

[11]. Time Series Prediction with Multilayer Perceptron, FIR and Elman Neural Networks. - Timo Koskela, Mikko Lehtokangas, Jukka Saarinen, and Kimmo Kaski. [En Línea]. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.1631.>

[12]. Chaotic Time-Series Prediction and the Relocating-LMS (RLMS) Algorithm for Radial Basis Function Networks. [En Línea]. <http://www.eurasip.org/Proceedings/Eusipco/1996/paper/pas_8.pdf.>

[13]. Predição Não-Linear de Séries Temporais Usando Redes Neurais RBF por Descomposição em Componentes Principais. Maria Cristina Filipetto de Castro. 2001. Tese de Doutorado. Universidad Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação.

[14]. Water flows modelling and forecasting using a RBF neural network. - Carlos H. Fajardo Toro, Daniel González Peña, Benedicto Soto González, Florentino Fernández-Riverola. Sistemas y Telemática. Universidad ICESI, Vol. 6 No 12. 2008.

[15]. Pronóstico de series de tiempo con tendencia y ciclo estacional usando el modelo airline y redes neuronales artificiales. J. D. Velásquez, C. J. Franco2. Ingeniería y Ciencia, ing. Cienc. ISSN 1794–9165 Volumen 8, número 15, enero-junio de 2012.

[16]. Spot Price Modelling in Brasil Using an Autoregressive Neural Network. Velásquez Juan, Dyner Isaac, Souza Reinaldo, C. Ingeniare. 2008. Revista Chilena de Ingeniería.

[17]. ARNN: A packages for time series forecasting using autoregressive neural netowork. Juan D. Velazquez, Cristian Zambrano, Laura Velez. [En Línea]. www.revistas.unal.edu.co/index.php/avances/article/download/26744/27047. 2011.

[18]. Internet Based Neural Network Online Simulation Tool. Milos Manic, Bogdan Wilamowki, Aleksander Malinowski. [En Línea]. <http://husky.if.uidaho.edu/pubs/nnott_iecon02_sf-007287.pdf>

[19]. Toward Automatic Time-Series Forecasting Using Neural Networks 3. Weizhong Yan. 2012. Neural Networks and Learning Systems, IEEE.

[20]. Software para la Gestión de la Autoevaluación para el Proceso de Acreditación de los Programas Académicos de la Corporación Universitaria de la Costa. Harold Arturo Combita Niño, Luis David Gutiérrez Molina. 2012. Universidad de la Costa. Facultad de Ingeniería.

[21]. Sencha, website oficial del framework Ext-JS . [En línea]. <http://www.sencha.com>.

[22]. Sencha – Ext JS, documentación. [En línea]. <http://docs.sencha.com/ext-js/4.1>

[23]. ExtJs 4 First Look. Loaine Groner. 2012. Birmingham – Mumbai.

[24]. Learning Python, Fourth Edition. Mark Lutz. 2009. ISBN: 978-0-596-15806-4.

18. ANEXOS

18.1 Tutorial para simular una red neuronal.

- Ingresar a la aplicación a través de la siguiente url**:** **http://proyectodegrado.no-ip.org/ .**
- Seleccionar el tipo de red neuronal para simular, como se muestra en la figura:

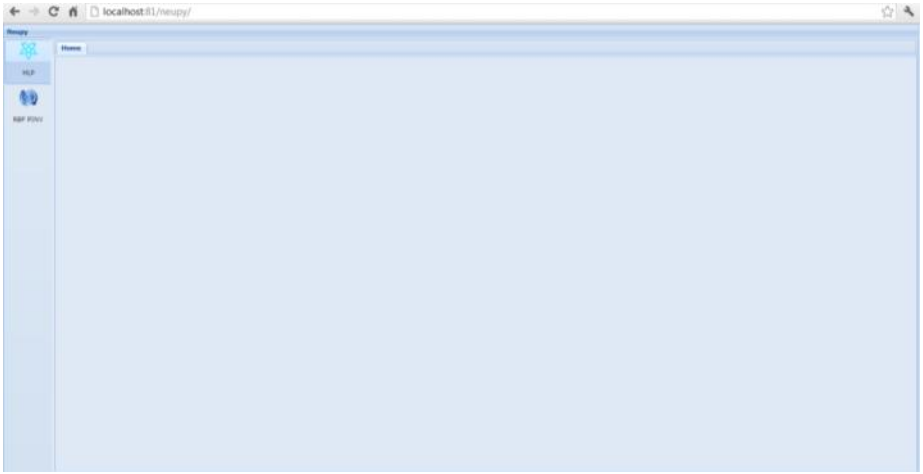

Figura 21. Ventana para seleccionar el tipo de red neuronal a simular.

- Luego de seleccionar la RNA, se procede con la importación de los datos, realizada con los botones (importar datos) ubicados en la parte inferior de la ventana.

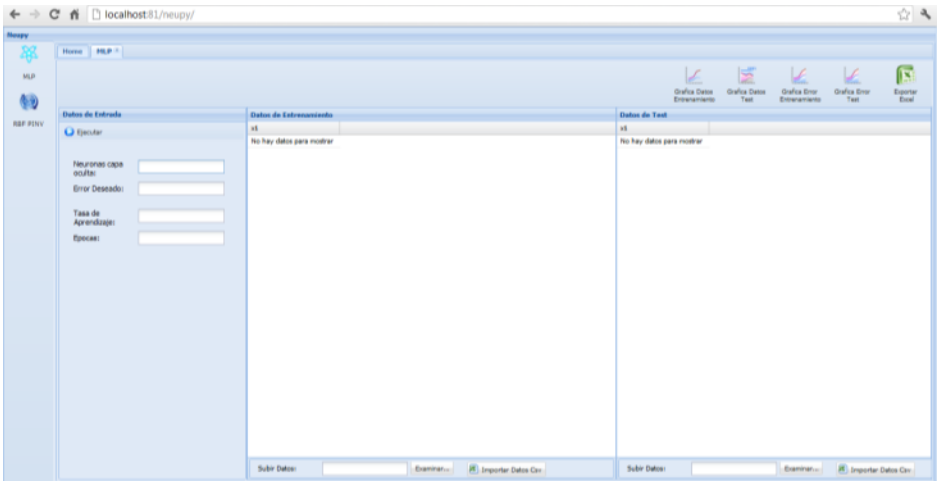

Figura 22. Ventana para configurar la red neuronal.

Hay que resaltar que el archivo .csv debe estar configurado de la siguiente manera:

	A	B	C	D	E
1	y1	y2	y3	Salida	Respuesta
2	9,9	9,7	8,1	6,4	
3	9,7	8,1	6,4	7,4	
4	8,1	6,4	7,4	11,9	
5	6,4	7,4	11,9	11,8	
6	7,4	11,9	11,8	13,1	
7	11,9	11,8	13,1	9,7	
8	11,8	13,1	9,7	7,6	
9	13,1	9,7	7,6	8,9	
10	9,7	7,6	8,9	13	
11	7,6	8,9	13	12,3	

Figura 23. Estructura del archivo de configuración de los datos de entrenamiento y/o test.

En la primera fila se especifican los nombres de las columnas que serán importados, teniendo en cuenta que se debe utilizar la palabra clave Salida para denotar la salida deseada y Respuesta para la salida de la red.

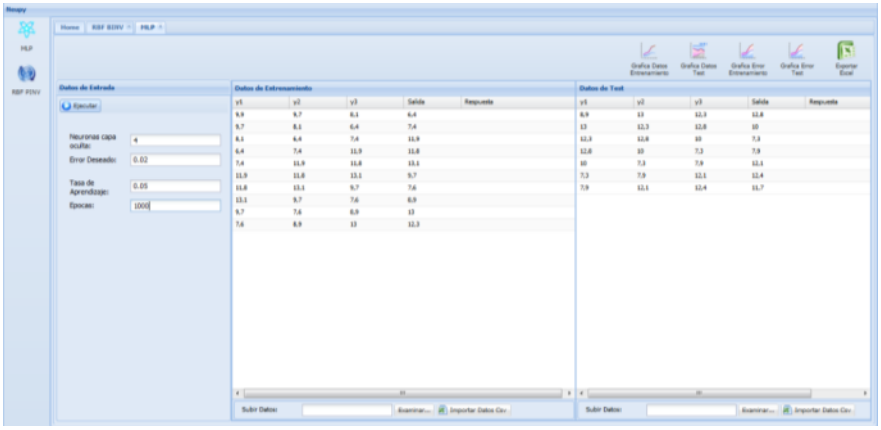

Figura 24. Configuración de la red neuronal a simular.

Luego de haber importado los datos de entrenamiento y test, se debe realizar la configuración de la red, llenando el formulario ubicado en la parte izquierda de la ventana. Finalmente se ejecuta el proceso de simulación pulsando el botón Ejecutar.

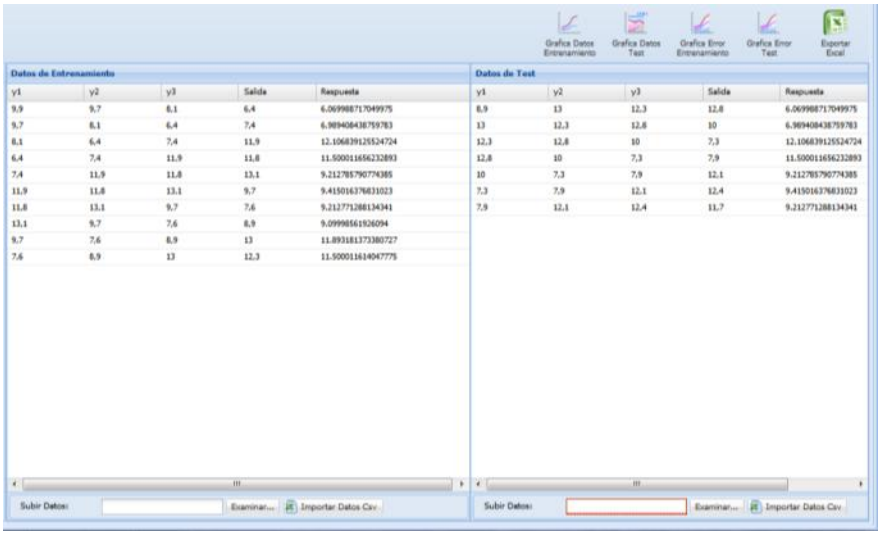

Figura 25. Resultados obtenidos de la simulación de la red neuronal.

Luego de haber obtenido los resultados, es posible graficar los datos obtenidos en la fase de entrenamiento y test, así como la grafica del error. Además de esto se encuentra habilitada la opción Exportar Excel, que se encarga de generar un archivo en Excel con los resultados obtenidos.

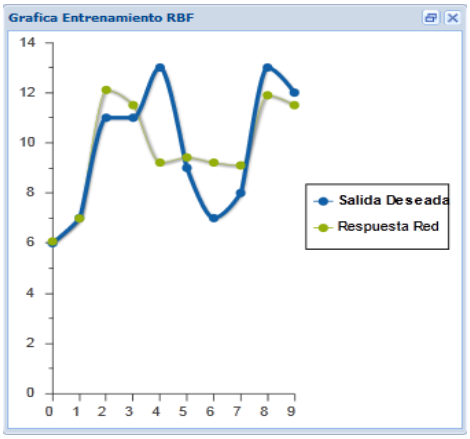

Figura 26. Grafica de los resultados obtenidos de la simulación.

18.2 Data Sets para Series de Tiempo.

A continuación se presentan un conjunto de datasets para series de tiempo.

Research Pipeline's:

Es un conjunto de sitios web para descargar datasets de diferentes áreas. Puede ser accedido mediante la siguiente url: http://www.researchpipeline.com/mediawiki/index.php?title=Main_Page

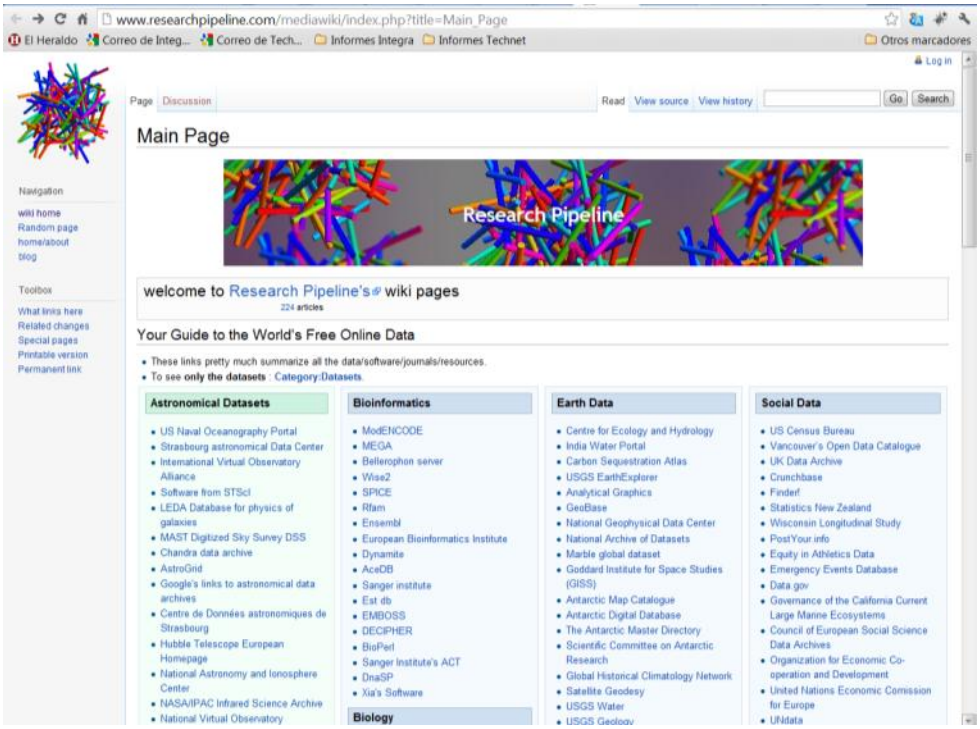

Figura 27. Datasets Research Pipeline's.

UCI Machine Learning Repository:

Es el repositorio del centro de aprendizaje máquina y sistemas inteligentes de la universidad de California. Es uno de los más importantes y con un gran conjunto de datos para realizar test. El acceso es a través de la siguiente url: http://archive.ics.uci.edu/ml/

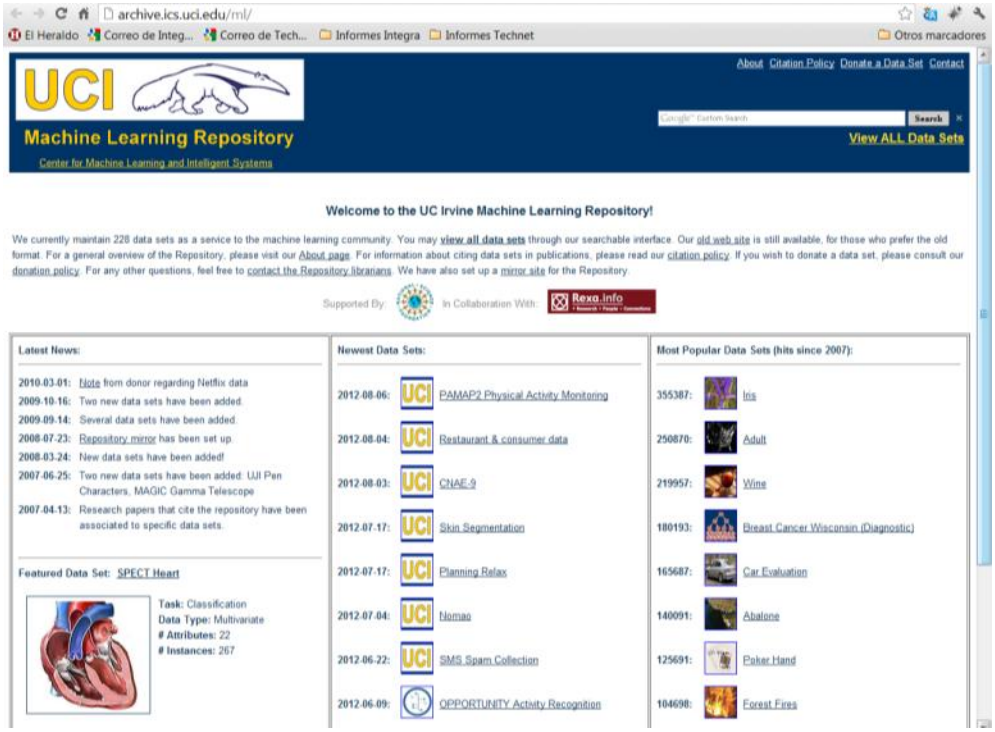

Figura 28. Repositorio UCI de la Universidad de California.

Time Series Data Library:

Es un gran repositorio de datasets de series de tiempo. Cuenta con distintos proveedores de datos de distintos países, convirtiéndolo en uno de los repositorios más grandes. Se pueden encontrar datasets de muchas áreas, como por ejemplo: físico, astronomía, química, computación, etc.

El acceso es mediante la siguiente url: http://datamarket.com/

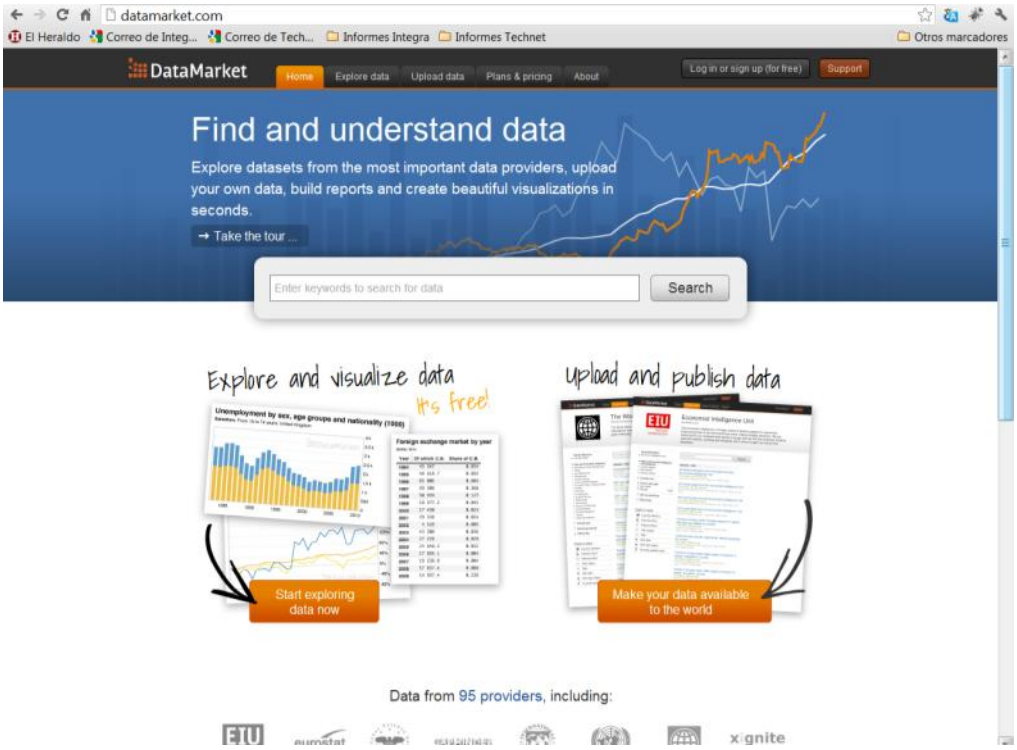

Figura 29. Time Seies Data Library Repository