Desarrollo de un modelo basado en Machine Learning para la predicción de la demanda de habitaciones y ocupación en el sector hotelero.

Ing. Fabián Pallares Cabrera

Ingeniero de Sistemas

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

MAESTRÍA EN INGENIERÍA

CARTAGENA

Desarrollo de un modelo basado en Machine Learning para la predicción de la demanda de habitaciones y ocupación en el sector hotelero.

Fabián Pallares Cabrera

Ing. De Sistemas

TESIS DE GRADO PARA OPTAR EL TÍTULO DE MAGISTER EN INGENIERÍA
Con énfasis en SISTEMAS Y COMPUTACIÓN

Director: WILLIAM CAICEDO, Ing de Sistemas M.S.c

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

MAESTRÍA EN INGENIERÍA

CARTAGENA noviembre 2014

Agradecimientos

Primero y antes que nada, dar gracias a Dios, por estar conmigo, por iluminar mi mente, darme fuerza de voluntad para no desfallecer después de las agotadoras jornadas de estudio y trabajo, proteger mi familia y poner en mi camino todos estas personas con las que he podido compartir y soportarme en este periodo de estudio, a mis padres por darme la oportunidad de estudiar una carrera profesional y estar siempre presto en el cuidado de los niños.

Quiero agradecer a mis hijos Gabriela y Fabián Andrés, quienes a pesar de su poca edad me han dado parte de su tiempo para que pueda estudiar y trabajar en esta investigación, ellos han sido mi motor para seguir avanzando.

Agradecer también a mi esposa quien siempre ha creído en mí, quien es mi amiga y compañera de estudio, quien siempre me ha apoyado y acompañado en todas estas travesías, con ella todo siempre ha sido mejor.

Por último, quiero agradecer de manera muy especial, a mi director de tesis William Caicedo, quien con sus aportes ha sido de mucha ayuda para el avance de este trabajo, su conocimiento, dedicación y esfuerzo ha sido fundamental.

CONTENIDO

Índice Tablas

Índice Figuras

RESUMEN

En esta investigación se entrenaron y validaron diferentes modelos para predecir la ocupación diaria de un hotel, y el mejor modelo encontrado se evaluó sobre los datos de pruebas, todo esto con el propósito de facilitar a los administradores hoteleros la toma de decisiones enfocada a la optimización de los recursos e incrementos en los beneficios del hotel, aplicables en la estrategia de Revenue Management.

El entrenamiento y validación se realizó utilizando cuatro técnicas de Machine Learning (Ridge Regression, Kernel Ridge Regression, Redes Neuronales Artificiales perceptron multicapa y de Función Base Radial). Los datos se separaron en tres conjuntos: entrenamiento, validación y pruebas. Los datasets se construyeron utilizando tres esquemas diferentes, en el primer caso se aplicaron los fundamentos para el desarrollo de modelos de series temporales, en donde las entradas al modelo están basadas en las observaciones realizadas de la ocupación en días anteriores, en el segundo esquema los datasets se basaron en las observaciones de la ocupación en días anteriores y se adicionaron otras variables como el días de la semana, festivos, temporada; en el tercer y último esquema se toman como variables de entrada al modelo la información de las reservas con días de anticipación, adicionalmente se incluye los días de semana , meses del año e información de festivos.

En los tres esquemas la validación se realiza sobre los datos reservados para tal fin, la medida de exactitud utilizada para comparar las técnicas es el MAPE (Mean Absolute Percentage Error), y la validación se realiza utilizando el método de validación Rolling Forecasting Update, aplicando los procedimientos de pronóstico one-step-ahead y (h) multi-step-ahead con h=7 para series de tiempo. En el tercer esquema de construcción de datasets se utiliza información de reservas con 90, 60, 30, 20, 15, 10, 7 días de anticipación.

Se desarrolla un algoritmo utilizando el lenguaje de programación Python, de forma que este permite realizar experimentos utilizando las técnicas Ridge Regression, Kernel Ridge Regression, Redes Neuronales (Perceptron y de Función Base Radial), se utilizan por cada técnica diferentes combinaciones de parámetros y esquemas diferentes de construcción de los datasets, de forma que se puede comparar el error MAPE producido en los datos de validación y al final se presentan los parámetros de los mejores modelos.

Los resultados encontrados muestran la superioridad de las técnica ridge regresión en el pronóstico de la ocupación, utilizando el esquema de construcción de datasets basado en las reservas observadas con 90, 60, 30, 20, 15,10 y 7 días de anticipación, adicionando información de festivos, temporadas, días de semana y meses del año. El error MAPE encontrado es de 8.2012 sobre el conjunto de validación y de 8.656144 sobre el conjunto de test.

1. INTRODUCCIÓN

El pronóstico de ocupación en hotelería es de mucha importancia en la toma de decisiones, anticiparse a la demanda futura permitirá a los administradores una mejor planeación en los inventarios, producción, mano de obra, compras, presupuesto financiero, administración de tarifas, todo enfocado en maximizar los ingresos y minimizar los costos.

Actualmente la industria hotelera ha adoptado estrategias para la maximización de los ingresos por habitaciones, a esta disciplina se le conoce como Revenue Management, la cual es una herramienta que permite vender cada habitación al cliente que está dispuesto a pagar el precio más alto a fin de lograr el más alto ingreso (Neamat El Gayar & Abdeltawad M.A. Hendawi). En la práctica de Revenue Management el pronóstico es el principal elemento en las decisiones de asignación de precios por habitación (Lee, 1990) (Weatherford, L. R & Kimes, S. E., 2003).

Otra práctica utilizada según la literatura, en la industria hotelera para aumentar los ingresos, es permitir reservar más habitaciones de la cantidad máxima disponible (overbooking), partiendo de la probabilidad de que existirán huéspedes que cancelarán tiempo antes del día de estadía, o simplemente no se presentarán (no-shows). Para esta práctica el reto está en poder conocer cuál es la cantidad de habitaciones por encima de la capacidad física a reservar sin incumplir a los

clientes que se presenten para hacer uso de reserva. En este punto nuevamente el pronóstico sigue siendo el componente principal de la decisión.

Muchas técnicas se han utilizado en la literatura para el pronóstico de la ocupación, algunos investigadores plantean la solución al modelamiento del pronóstico de la ocupación utilizando series de tiempo, otros utilizan modelos más avanzados considerando la tasa de incremento de reservaciones (pickup). En todos los casos la ocupación está influenciada por fenómenos como la variación estacional, días de la semana, festivos, eventos especiales, presencia de nuevos competidores, y otros factores más complejos de estimar como por ejemplo, alargues y acortes de estadías sobre huéspedes en el hotel, cancelaciones, huéspedes que llegan sin reservas (WALK IN), huéspedes que no se presentan el día de estadía (NO SHOW), etc.

En este trabajo se estudian los aportes de las técnicas de Machine Learning en la predicción de la ocupación utilizando técnicas como Ridge Regression, Kernel Ridge Regression, Redes Neuronales Artificiales (Perceptron Multicapa, Redes Función Base Radial).

En esta investigación se consideraron dos enfoques para la predicción de la demanda de habitaciones, el primer enfoque se basa en modelos de series de tiempo, en donde las variables explicativas del modelo corresponden a las observaciones de la demanda de la ocupación en periodos de tiempo anterior al día de llegada. En el segundo enfoque se consideró el pronóstico de la ocupación basado en las reservas realizadas con varios días de anticipación, al igual se

incluyen otras variables explicativas como días de la semana, número de días festivos antes y después del día de análisis, meses del año y temporada.

La organización del documento es la siguiente: El segundo capítulo trata los conceptos generales de los métodos de series de tiempo y métodos tradicionales, se presentan los conceptos básicos y generalidades de las técnicas de Machine Learning, específicamente se tratan las técnicas Ridge Regression, Kernel Ridge Regression, Redes Neuronales (Perceptron Multicapa y Redes Neuronales Función Base Radial); además se explican los procedimientos de validación utilizados para series de tiempo y los métodos de cálculo del error de validación. El tercer capítulo presenta el estado del arte referente al pronóstico de ocupación, el cuarto capítulo presenta la metodología aplicada, composición de los datos y además se presentan los resultados del error de validación por cada técnica y método utilizado. En el quinto capítulo se presentan los resultados obtenidos sobre el conjunto reservado para pruebas y por último se expresan las conclusiones de la investigación y futuros trabajos.

1.1 Formulación del Problema

En la actualidad, la industria hotelera ha encaminado esfuerzos implantado estrategias para maximizar los ingresos generados por habitaciones. La estrategia más usada recientemente es conocida como Revenue Management. (Sevinç Gökşen, 2012),

Revenue Management persigue vender el producto correcto, al precio correcto y al cliente correcto en el momento correcto.

La estrategia se fundamenta en establecer segmentos de clientes, definir tarifas adecuadas para cada segmento, y a través de pronóstico de la demanda determinar a qué precios es conveniente vender, a quien y cuando; todo soportado en modelos matemáticos para optimizar los ingresos.

Revenue Management ofrece flexibilidad para rechazar las reservas si esta no ofrece los mejores beneficios, todo esto apoyado en los pronósticos. Sin embargo, frecuentemente la aplicación de estas técnicas produce impacto negativo en los ingresos debido a los cálculos errados de las predicciones, los cuales contribuyen a tomar decisiones erradas en la planeación y administración de tarifas, por ejemplo si el pronóstico indica que la ocupación va a ser alta muy probablemente la estrategia es subir el precio y si es baja seguramente toca bajar el precio para promover la venta. Si el pronóstico no es correcto se puede correr el riesgo de que se venda a un precio excesivamente bajo, o se deje de realizar una venta.

Otras de las dificultades que se presentan a diario se debe a que en ocasiones los huéspedes reservan y no se presentan al hotel (no-shows), los hoteleros hacen uso del overbooking para controlar los efectos negativos de lo anteriormente descrito.

1.2 Justificación

La industria del turismo es de altísima importancia para muchos países y ciudades en todo el mundo, esta se convierte en un proveedor directo e indirecto de empleos. Cartagena de Indias es una de las principales ciudades turísticas de Colombia, y el uso de herramientas y estrategias adecuadas para la toma de decisiones en el sector hotelero puede contribuir activamente al desarrollo social y económico de la ciudad.

Dentro del proceso de toma de decisiones, la necesidad de pronósticos con mayor exactitud en la ocupación de cada hotel permite establecer una estrategia adecuada de precios para impulsar las ventas lo cual es de vital importancia dada la naturaleza perecedera de los productos (habitaciones), debido a que si no se realiza la venta de la habitación no hay forma de recuperar los ingresos en el futuro.

En esta propuesta estamos interesados en desarrollar un software prototipo para pronóstico de ocupación que pueda ser utilizado por los administradores hoteleros como apoyo en la toma de decisiones, enfocado a la estrategia de maximización de ingresos y planeación de recursos, de forma que esto pueda ser una herramienta para mejorar la competitividad del sector en la ciudad.

1.3 Objetivos

1.3.1 General

Desarrollar y validar un modelo basado en Machine Learning para la predicción de la demanda de habitaciones y ocupación en el sector hotelero.

1.3.2 Específicos

1. Realizar una revisión literaria para escoger las técnicas de Machine Learning que pueden ser aplicadas a la predicción de series temporales.
2. Construir un conjunto de datos para entrenamiento, validación y pruebas que permita implementar las técnicas seleccionadas en el objetivo anterior.
3. Realizar una comparativa de las diferentes técnicas de Machine Learning escogidas, con miras a identificar cual ofrece mejores resultados en la predicción de la demanda de habitaciones y ocupación en el sector hotelero.
4. Desarrollar un software prototipo que implemente la técnica de Machine Learning seleccionada con los mejores resultados.

1.4 Aspectos Metodológicos

Para dar inicio a esta investigación se realizó un estudio de cada uno de las técnicas de Machine Learning utilizadas en temas de pronósticos en series temporales y para análisis de regresión, usando como fuente de información las bases de datos científicas y el conocimiento de profesionales que han trabajado con estas técnicas, esto permitió determinar las técnicas a utilizar en el transcurso de esta investigación.

Para la aplicación de las técnicas se construyeron tres conjuntos de datos, un conjunto para entrenamiento, uno para validación y el último para test, los datos son recolectados desde las operaciones de reservas y ocupación diaria de un hotel, de forma que los resultados de este trabajo son basados en información real.

Como parte de esta investigación se realizó la construcción de prototipos utilizando el lenguaje Python basados en las técnicas de Machine Learning escogidas, una vez finalizado estos desarrollos, se aplicó una metodología estándar en machine learning construyendo conjuntos de datos separados para validación y pruebas, el desempeño de los modelos entrenados se miden utilizando la medida de error MAPE (Mean Absolute Percentage Error) , se selecciona el modelo con menor error de pronóstico sobre los datos de validación, y solo se utiliza este modelo sobre el conjunto de datos reservados para pruebas, evitando con esto un sesgo optimista, los resultados son presentados más adelante en este documento.

2. MARCO TEÓRICO

Las técnicas de pronósticos tienen como objetivo general obtener un valor futuro bajo la incertidumbre, tomando como base la información histórica y tratando de encontrar un patrón de comportamiento que permita conocer cuál va a ser el compartimiento en el futuro.

Los métodos de pronósticos se dividen en Cualitativos y Cuantitativos. Los métodos cualitativos tienen como característica principal, depender del conocimiento y experiencia de los expertos, dentro de estos métodos encontramos las siguientes técnicas:

- Ajuste Curva subjetiva: Es creada por experto y se basa en el conocimiento del negocio, aplicada en la predicciones de nuevas productos, donde se analiza, para productos con características similares, cual va a hacer el comportamiento desde el lanzamiento de este, hasta la estabilización de las ventas, la complejidad está en cómo definir la forma que tendrá la curva.
- Delphi: Utiliza el juicio del experto. Existen varios expertos que establecen el pronóstico. No hay datos, todo se hace por intuición.

Por otra parte, las técnicas cuantitativas de predicción consisten en encontrar un patrón en los datos disponibles para poder proyectarlo al futuro, es decir, estos métodos requieren de un análisis de la información para poder efectuar la predicción de la variable que sea de interés, dentro de estos métodos encontramos:

2.1 Métodos de series temporales

Los métodos de series temporales utilizan datos históricos como base para estimar resultados futuros, se asume que la variable a predecir es función de las observaciones de estas variables en periodos de tiempos anteriores. En este tipo de análisis pueden estar involucrados los siguientes componentes:

- **Tendencia**

Es el componente de largo plazo que representa el crecimiento o declinación de la serie. En términos intuitivos, la tendencia caracteriza el patrón gradual y consistente de las variaciones de la serie, que es consecuencia de "fuerzas persistentes" que afectan el crecimiento o la reducción de la serie.

- **Ciclo**

Es la fluctuación en forma de onda alrededor de la tendencia. Una de las fluctuaciones cíclicas más comunes en series de tiempo son las llamadas ciclo económico, la cual está representada por fluctuaciones ocasionadas por periodos recurrentes de prosperidad alternando con recesión, sin embargo, dichas fluctuaciones no necesitan ser causadas por cambios en los

factores económicos, como por ejemplo; en la producción agrícola donde la fluctuación cíclica puede estar ocasionada por los cambios climáticos.

- **Variación Estacional**

Son patrones periódicos que se repiten año tras año, factores como el clima y las costumbres ocasionan estos tipos de patrones.

- **Fluctuaciones irregulares o irregularidad**

Son movimientos erráticos que siguen un patrón indefinido o irregular. Estos movimientos representan lo que queda de la serie después de haber restado las demás componentes.

Muchas de las fluctuaciones irregulares son causadas por hechos inusuales que no se pueden predecir, como son los sismos, huracanes, guerras, entre otros.

Algunas técnicas de series de tiempo encontradas son:

1. Método Ingenuo: este método asume que la variable a predecir tiene igual valor a su medición anterior.
2. Promedio Móvil: la predicción es el resultado del promedio de n observaciones anteriores.
3. Métodos de Descomposición. Como su nombre lo indica, descompone la serie de tiempo en sus cuatro componentes con el objetivo de describir cada una de ellas por separado logrando así describir y predecir en conjunto la serie de tiempo.

4. Suavizamiento Exponencial. El objetivo es filtrar o suavizar la serie para tener una mejor idea del comportamiento de la tendencia y por tanto tener un pronóstico más confiable. Dentro de los métodos de suavizamiento encontramos los métodos de Holt Winters.
5. Metodología de Box-Jenkins. Proporciona una colección más extensa de modelos de Predicción además de ser un procedimiento más sistemático para ayudar a identificar el modelo adecuado.

2.2 Métodos por análisis de regresiones

El análisis de regresión es una técnica estadística para estudiar la relación entre variables. El término regresión fue introducido por Francis Galton en 1886. Su trabajo se centró en la descripción de los rasgos físicos de los descendientes (variable A) a partir de los de sus padres (variable B). Estudiando la altura de padres e hijos a partir de más de mil registros de grupos familiares, se llegó a la conclusión de que los padres muy altos tenían una tendencia a tener hijos que heredaban parte de esta altura, pero que revelaban también una tendencia a regresar a la media. Galton generalizó esta tendencia bajo la "ley de la regresión universal": «Cada peculiaridad en un hombre es compartida por sus descendientes, pero en media, en un grado menor».

El objetivo de la regresión es descubrir la relación funcional entre la entrada y la salida de este sistema, para poder así predecir la salida de este cuando se le presenta un dato de entrada nuevo. En un análisis de regresión simple existe una variable respuesta o dependiente (y) y una

variable explicativa o independiente (x). El propósito es obtener una función sencilla de la variable explicativa, que sea capaz de describir lo más ajustadamente posible la variación de la variable dependiente. La variable explicativa puede estar formada por un vector de una sola característica o puede ser un conjunto de n características, atributos o dimensiones (regresión múltiple). La regresión se utiliza para predecir una medida basándonos en el conocimiento de otra y la intención final es que dado un vector de entrada x_{i+1} se persigue predecir un valor de salida y_{i+1} a partir de una función generada mediante la supervisión previamente observada de un conjunto de entrenamiento inicial. En la figura siguiente se muestra un ejemplo donde dada las primeras 9 observaciones para x con sus valores en y, se trata de predecir para x_{10} el valor de y_{10}.

X	Y
1	3.5
2	4
3	4.5
4	7
5	5.5
6	8
7	6.5
8	8
9	9.5
10	?

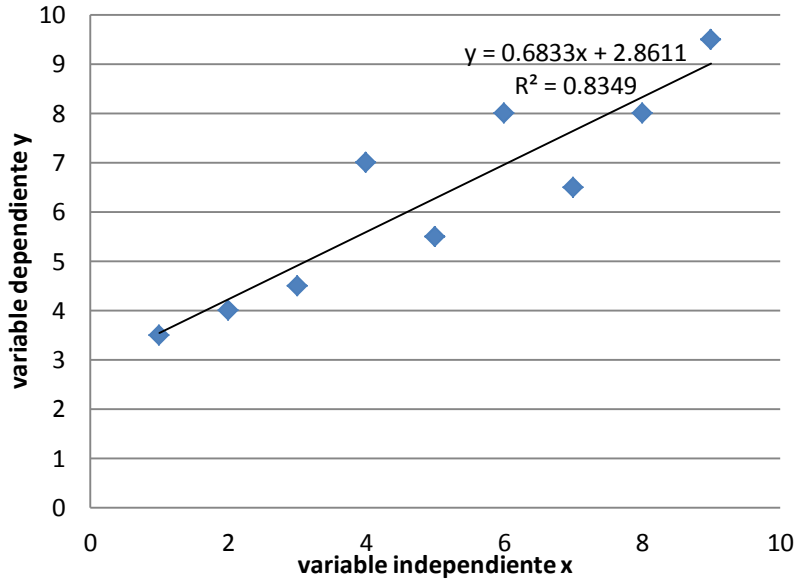

Figura. 1 Gráfico de regresión variable Y en función de X, desarrollado en Microsoft Excel

2.2.1 Regresión lineal

Este método consiste en construir a partir de los datos de entrada una función lineal (línea recta) que atraviese los puntos con la mínima variación entre cada observación y el punto predicho por la función.

Como los valores observados de la variable dependiente difieren generalmente de los que predice la función, esta posee un error. La función más eficaz es aquella que describe la variable dependiente con el menor error posible o, dicho en otras palabras, con la menor diferencia entre los valores observados y predichos. La diferencia entre los valores observados y predichos (el error de la función) se denomina variación residual o residuos. Para estimar los parámetros de la función se utiliza el ajuste por mínimos cuadrados. Es decir, se trata de encontrar la función en la cual la suma de los cuadrados de las diferencias entre los valores observados y esperados sea menor. Sin embargo, con este tipo de estrategia es necesario que los residuos o errores estén distribuidos normalmente y que varíen de modo similar a lo largo de todo el rango de valores de la variable dependiente. Estas suposiciones pueden comprobarse examinando la distribución de los residuos y su relación con la variable dependiente.

Para encontrar la función lineal con el mínimo de error, se calcula la derivada del error en función de los pesos a la siguiente ecuación:

$$E(w) = \sum_{i}^{n} (y_i - X_i w)^2$$

Encontrando la siguiente ecuación para el cálculo de los pesos:

$$w = (X^T X)^{-1} X^T y$$

2.2.2 Regresión no lineal

Este método consiste en conseguir una función polinomial de mayor grado que permita minimizar el error, la función polinomial más sencilla es la cuadrática $y = w_0 + w_1 x + w_2 x^2$ Pero puede usarse una función cúbica u otra de un orden aun mayor (orden k) $y = w_0 + w_1 x + w_2 x^2 + w_k x^k$ capaz de conseguir un ajuste casi perfecto a los datos.

Sin embargo, este método puede lograr un ajuste casi perfecto, error aproximadamente igual a cero, contemplando instancias ruidosas (outliers), logrando confundir al sistema de predicción ante la presencia de nuevas entradas (Zhang, 2009)

Este sobreajuste (overfitting) es un problema muy común y produce un modelo que no es capaz de generalizar. Normalmente, fronteras de decisión muy complejas producen sobreajuste, no funcionando adecuadamente con nuevas instancias.

2.3 Métodos Machine Learning

Machine Learning puede ser ampliamente definida como métodos computacionales que usan la experiencia para mejorar el desempeño de las predicciones, logrando ser estas mas precisas. Cuando nos referimos a experiencias hablamos específicamente de la información histórica recolectada que se utiliza para los procesos de entrenamiento. (Mehryar Mohri, Afshin Rostamizadeh, & Ameet Talwalkar, 2012)

Machine Learning puede ser aplicada en tareas de:

- Clasificación: en este caso la idea central de la aplicación de la técnica es identificar a que clase pertenece una nueva entrada, tales ejemplos se aplican en la clasificación de documentos, imágenes, diagnostico medico, etc.
- Regresión: predice un valor real para cada ítem, por ejemplos la predicción de la demanda, stocks de inventarios, variables económicas, tasas, etc.
- Ranking: utilizada para ordenar ítems basado en algún criterio, por ejemplo las búsquedas web.
- Clustering: este tipo de aplicaciones son fuertemente utilizadas en procesos comerciales para segmentar clientes y productos de esta forma facilitar los procesos de decisiones referentes a que vender y a quienes.
- Reducción de Dimensionalidad: Transforma la representación de los ítems inicial en una representación de baja dimensionalidad, perseverando las propiedades de la inicial representación. Un ejemplo de esto lo encontramos en el reprocesamiento de imágenes digitales.

Otras tareas donde encontramos la aplicación de Machine Learning son el procesamiento de lenguaje natural, reconocimiento de sonido, reconocimiento de caracteres ópticos (OCR). Reconocimiento de rostro, etc.

Entre las técnicas de Machine Learning aplicadas en tareas de regresión encontramos:

- Multilayer Perceptron Neural Network (MLP)
- Radial Basis Function Neural Network (RBF)
- Generalized Regression Neural Network (GRNN)
- K Nearest Neighbor Regression (KNN)
- Classification and regression Trees (CART)
- Support Vector Regression (SVR)
- Gaussian Processes (GP).
- Ridge Regression
- Kernel Ridge Regression

Para una mejor comprensión del trabajo y de los experimentos realizados, en las secciones siguientes explicaremos en que consisten las técnicas (Ridge Regression, Kernel Ridge Regression y Redes Neuronales (Perceptron y de Función Base Radial)) las cuales se seleccionaron para realizar los experimentos.

2.3.1 Ridge Regression

La técnica Ridge Regression (RR.) (Hoerl, 1970) , es un método para regresión que consiste en incluir a la función de minimización del error un término de penalización, que permite evitar el sobre ajuste y proporcionar un menor error de generalización.

$$F(w) = k\|w\|^2 + \sum (y_i - X_i w)^2$$

Donde ***k*** es el coeficiente ridge, al derivar la función y despejar *w:*

$$w = (X^T X + kI_n)^{-1} X^T y$$

La predicción puede ser escrita como. $\hat{y}(x) = xw = \sum_{i=1}^{n} w_i(x)_i$

2.3.2 Kernel Ridge Regression

Es una técnica que realiza un mapeo de los datos de entrada (considerados no lineales) en un espacio de características de más alta dimensión (donde corresponden a un modelo aproximadamente lineal) obteniendo errores de ajuste mucho menores que los conseguidos en el espacio de entrada inicial, y conservando la eficiencia del factor de regularización utilizada en la técnica Ridge Regression (Castillo, 2012)

Esta técnica se aplica si existe sospecha de relaciones no lineales en los datos de entrenamiento, en ese caso las técnicas de regresión anteriores serán incapaces de modelarlas adecuadamente con un error mínimo aceptable (el sesgo introducido en Ridge Regression ayuda pero a veces también resulta insuficiente).

Si el espacio de objetos X es mapeado en otro espacio euclídeo (llamado espacio de características) H: X → H y aplicamos Ridge Regression en el espacio de características, entonces la predicción puede ser rescrita como:

$$\hat{y} = Y'(K + aI_n)^{-1}k$$

Donde K es la matriz con elementos $(K)_{i,j} = Ҡ(x_i, x_j)$ note que esta representación está basada en los datos de entradas para el proceso de entrenamiento.

Mientras que k es el vector con elementos $k = Ҡ(x, x_i)$, donde x_i representa la matriz de datos de entrenamiento y x representa la matriz con los nuevos datos de entradas para la evaluación.

Ҡ Es el kernel, definido por: $Ҡ(\boldsymbol{x_1}, \boldsymbol{x_2}) = \boldsymbol{F}(\boldsymbol{x_1}).\boldsymbol{F}(\boldsymbol{x_2})$

Para todo $x_1, x_2 \in X$

Existen varias funciones de Kernel, dentro de las cuales enunciamos:

- Kernel Polinomial: $(\gamma . x_1' x_2 + \beta)^d$ d es el grado del polinomio.

- Kernel Gaussiano: $e^{\left(-\|x_1-x_2\|^2/2\delta^2\right)}$

2.3.3 Redes Neuronales

Las Redes de Neuronas Artificiales (Smit, 1993) (Artificial Neuronal Networks, ANN) son modelos de aprendizaje inspirados en el sistema nervioso de los animales. Se componen de una serie de unidades, denominadas neuronas, que de forma simplificada simulan la funcionalidad de las neuronas biológicas.

Las neuronas biológicas tienen tres componentes principales, las dendritas, el cuerpo de la célula o soma, y el axón. Las dendritas, son el árbol receptor de la red, son como fibras nerviosas que cargan de señales eléctricas el cuerpo de la célula. El cuerpo de la célula realiza la suma de esas señales de entrada. El axón es una fibra larga que lleva la señal desde el cuerpo de la célula hacia otras neuronas. El punto de contacto entre un axón de una célula y una dendrita de otra célula es llamado sinapsis, la longitud de la sinapsis es determinada por la complejidad del proceso químico que estabiliza la función de la red neuronal.

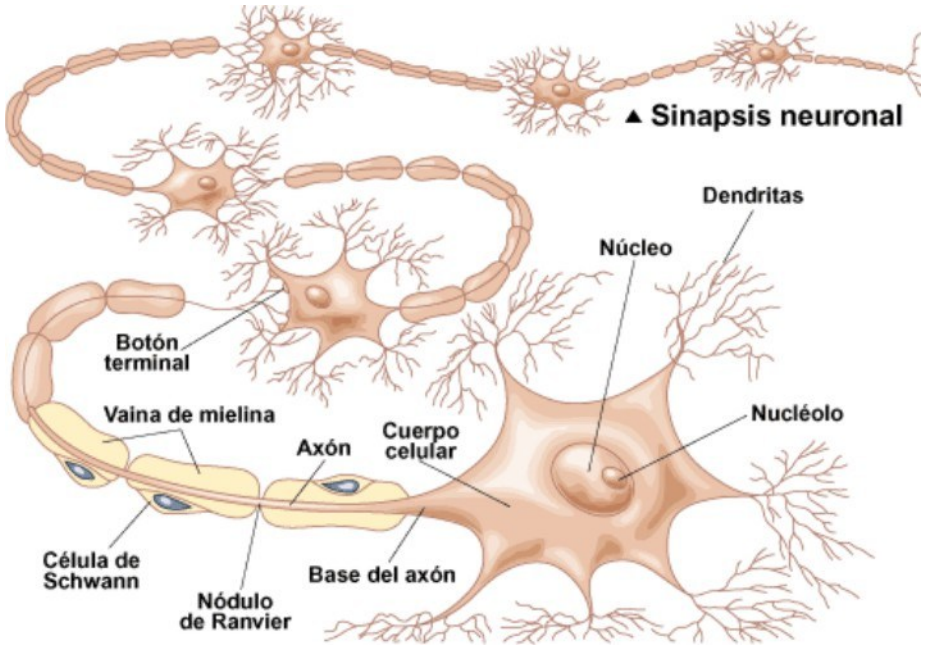

Figura. 2 Red Neuronal Biológica tomado de http://andrealezcano.blogspot.com/2011/04/rna-redes-neuronales-artificiales.html.

Una neurona artificial recibe una serie de entradas a través de interconexiones y emite una salida, esta salida se da a través de tres funciones:

1. Función de propagación: suele calcular una combinación de cada entrada modificada por el peso de su interconexión.
2. Función de Activación: recibe como entrada a la anterior, aunque puede o no utilizarse, esta función permite incluir la no linealidad en la función de salida de la red, existen diversas funciones de activación, tales son los ejemplos de Función Sigmoide, Tangente Sigmoide, Escalón, Lineal, Gaussiana, etc.
3. Función de transferencia, aplicada al valor devuelto por la función de activación.

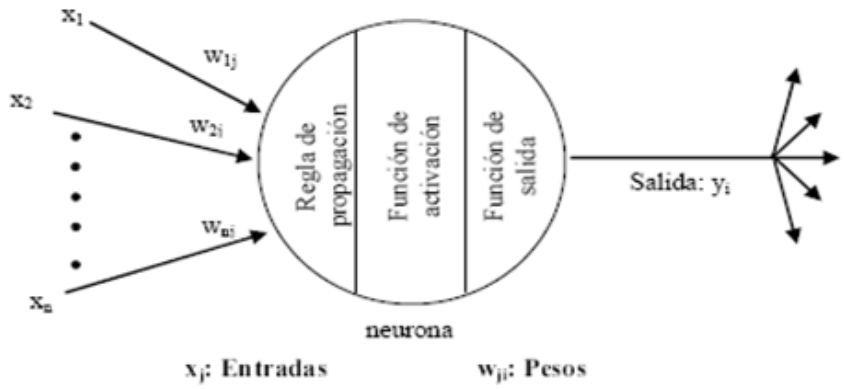

Figura. 3 Neurona Artificial tomada de

http://advancedtech.wordpress.com/2007/08/31/elementos-basicos-de-una-red-neuronal-artificial/

Estos modelos son extremadamente flexibles, pudiéndose obtener un número virtualmente infinito de modelos de redes distintos al variar el tipo de neuronas en cada capa y la forma de interconectarlas en red y el mecanismo de aprendizaje.

De lo mejor de las redes neuronales es la capacidad de aprender de la experiencia, almacenando conocimiento en los pesos asociados a las conexiones entre neuronas, el poder de utilizar información con alto nivel de ruido y/o incompleta para ser procesada con resultados satisfactorios, la capacidad de reconocer y organizar datos que no habían sido vistos con anterioridad. Todo esto hace que las redes neuronales se han una gran herramienta para el reconocimiento de patrones, clasificación y agrupación de datos.

Las redes neuronales se utilizan hoy en una gran cantidad de tareas, como por ejemplo estabilización de ruido en telefonía, reconocimiento de imágenes, filtrado de correo no deseados (spam), predicción en series de tiempo, clasificación de pacientes con determinadas sintomatologías, etc.

El entrenamiento de las redes neuronales puede ser clasificados en tres tipos: Entrenamiento supervisados, sin supervisión y entrenamiento por refuerzo. Los algoritmos de entrenamiento supervisados requieren el uso de ejemplos del que debería ser el comportamiento adecuado de la red en presencia de entradas específicas. Ejemplos de entrenamiento supervisado son las reglas de aprendizaje del perceptron, la regla delta (Widrow-Holf) y el algoritmo de retro propagación (Backpropagation). Por otro lado, los algoritmos de aprendizaje sin supervisión no requieren ejemplos del comportamiento adecuado, estos a través de un proceso de auto-organización modifican la red de tal forma que aprendan a clasificar un espacio de entrada determinado. Por último, en los algoritmos de entrenamiento por refuerzo la red no reciben ejemplos de comportamiento correcto, sino unas recompensas o calificación por la acción tomada, siendo entonces la meta de la red neuronal la maximización de la recompensa recibida.

2.3.3.1 Perceptron Multicapa (MLP)

Un Perceptron Multicapa (Multilayer Perceptron, MLP) es un modelo de red neuronal artificial formada por múltiples capas de neuronas. Este es uno de los modelos más usados de redes neuronales artificiales.

La arquitectura de un Perceptron multicapa consta de varias capas de neuronas, en las que las salidas de una capa son las entradas de la siguiente. Las conexiones entre neuronas tienen un valor de ponderación o peso y cada neurona tiene una función de activación con la que genera su salida en función de las entradas. Las capas pueden clasificarse en tres tipos:

- Capa de entrada: Constituida por aquellas neuronas que representan cada entrada a la red, estas neuronas no producen procesamiento.
- Capas Ocultas: Formada por aquellas neuronas cuyas entradas provienen de capas anteriores y las salidas pasan a neuronas de capa posteriores.
- Capa Salida: Neuronas cuyos valores de salida corresponden con las salidas de toda la red.

Estas redes son populares porque son capaces de aproximar cualquier función continua en un intervalo hasta el nivel deseado. (Funahashi, 1989). Usualmente para pronósticos o modelos de regresión se utilizan una capa de entrada con tantas neuronas como variables independientes del modelo, una capa oculta con función de activación Sigmoide donde los nodos se establecen experimentalmente, y una capa de salida con una neurona y función de activación lineal.

Para el aprendizaje de los MLP se suele usar el algoritmo de propagación hacia atrás conocido como Backpropagation (Webos, 1974), este algoritmo técnicamente aplica la optimización por gradiente descendente para encontrar los pesos de la red que minimicen el error, para pronóstico de series de tiempo por ejemplo, el error está en función del MSE (error cuadrático medio) o cualquier otra medida como MAPE (error porcentual medio absoluto).

Por otro lado, desde la Inteligencia Computacional se han propuesto diversas técnicas para el modelado y pronóstico de series de tiempo, de las disponibles, las redes neuronales artificiales (ANN) han mostrado ser más robustas que otras técnicas tradicionales, especialmente en la representación de relaciones complejas que exhiben comportamientos no lineales, por ejemplo véase (F. A. Villa, 2008) (M. Ghiassi, 2005). Masters (T. Masters, 1993), recomienda utilizar ANN en vez de alguna técnica tradicional por las siguientes razones:

- Poseen una amplia capacidad para aprender relaciones desconocidas a partir de un conjunto de ejemplos.
- Tienen una alta tolerancia a patrones extraños de ruido y componentes caóticas presentes en un conjunto de datos.
- Son suficientemente robustas para procesar información incompleta, inexacta o contaminada.
- No restringen el tipo de no linealidad de la serie de tiempo a la estructura matemática del modelo de red neuronal.

Para pronosticar una serie de tiempo con un MLP, se toma como punto de partida, que: una serie se define como una secuencia de T observaciones ordenadas en el tiempo:

$$y_t = \{y_i\}_1^t$$

Para la cual se pretende estimar una función que permita explicar y_t en función de sus rezagos $\{y_{t-1}, y_{t-2}, \dots, y_{t-p}\}$, es posible especificar matemáticamente la función y_t como un MLP, así:

$$y_t = \beta_* + \sum_{h=1}^{H} \beta_h x\, g\left(\alpha_h + \sum_{p=1}^{p} W_{p,h} x y_{t-p}\right) + \varepsilon_t$$

Se supone que ε_t sigue una distribución normal con media cero y varianza desconocida σ_2

H Representa el número de neuronas de la capa oculta.

P Es el número máximo de rezagos considerados (neuronas de entradas).

$g\left(\alpha_h + \sum_{p=1}^{p} W_{p,h} x y_{t-p}\right)$ Es la función de activación de la capa oculta.

Los parámetros $W = [\beta *, \beta h, \alpha\, h, wp, h]$, con h = 1, 2,…, H y p = 1, 2,…, P, son estimados usando el principio de máxima verosimilitud de los residuales, el cual equivale a la minimización del error cuadrático medio. Para resolverlo se han propuesto diversas técnicas de optimización:

Basadas en gradiente, tales como Backpropagation y Rprop - Resilient Backpropagation (Riedmiller, 1994) (Braun, 1993), Heurísticas, como estrategias evolutivas (D. M. Ortíz, 2007), entre otras.

Los MLP pueden adolecer del fenómeno del sobreajuste, básicamente por tres causas: La primera está relacionada con la existencia de datos extremos (outliers) en el conjunto de entrada,

esto hace que la varianza de los parámetros de la red sea alta; la segunda con la cantidad de neuronas en la capa de entrada y oculta, es decir, el tamaño óptimo de la red. Si se selecciona una cantidad alta o inadecuada de entradas, se sobre ajusta la red neuronal, y esta memoriza los datos de entrenamiento en vez de aprender el comportamiento de la serie, esto se evidencia cuando se produce un error de entrenamiento muy pequeño y un error de validación muy alto (F. A. Villa, 2008); la tercera, el subconjunto de entrenamiento no posea el patrón generador de la serie (Ma, 2009).

Las primeras dos causas se pueden controlar mediante el uso de la regularización, mientras que la tercera mediante la selección de una técnica adecuada de validación. Sin embargo, en la literatura más relevante no se ha considerado usar integralmente regularización y validación cruzada para controlar efectivamente el sobreajuste en redes neuronales.

La idea principal del método es estabilizar la solución usando algún tipo de función para penalizar la función objetivo. El método de regularización tiene como objetivo realizar un intercambio equilibrado entre la fiabilidad de los datos de entrenamiento y las bondades del modelo.

En los procedimientos de aprendizaje supervisado, el intercambio se realiza a través de la minimización el riesgo total (S. Haykin, 1999)

$$R(W) = \boldsymbol{\xi}_s(W) + \lambda \boldsymbol{\xi}_c(W)$$

Donde $\xi_s(W) = \sum_{t=1}^{T}(\hat{y}_t - y_t)^2$ se conoce como la medida estándar del rendimiento, se acostumbra el uso del error cuadrático medio (MSE) y $\xi_c(W)$ es la penalización compleja, conocida también como estrategia de regularización. λ Es el factor de regularización.

Dentro de las estrategias de regularización más utilizada en la literatura (C. Leung, 2010) es denomina Weight Decay, esta estrategia fue propuesta por (Hinton, 1989), su formulación es la siguiente:

$$\boldsymbol{\xi}_c(W) = \left\| {w_{p,h}}^2 \right\|$$

Donde $w_{p,h}$ son los pesos de las entrada p a la neurona h.

2.3.3.2 Redes Neuronales Función Base Radial

Estas redes neuronales son introducidas por Broomhead y lowe, 1998. Este método se caracteriza porque en su diseño consta de neuronas activadas mediante función de activación radial de carácter no lineal con centros gravitacionales propios y en la capa de salidas mediante funciones lineales.

El entrenamiento de la red con función de base radial es hacia adelante a diferencia de la red usando Backpropagation, de esta forma la salida de esta red está influenciada por una transformación no lineal originada en la capa oculta a través de una función radial y una lineal en la capa de salida.

La función radial en cada nodo de la capa oculta tiene como parámetros un centro y un ancho, el centro regularmente es un vector de igual dimensión del vector de entrada y cada nodo de capa oculta maneja un centro diferente. El ancho es el término empleado para identificar la amplitud de la campana de gauss originada en la función radial, es decir, la desviación estándar, muchos autores consideran el ancho un valor constante para todos los nodos en la capa oculta (Lowe, 1989) de esta forma reducen los pasos de la construcción de modelos.

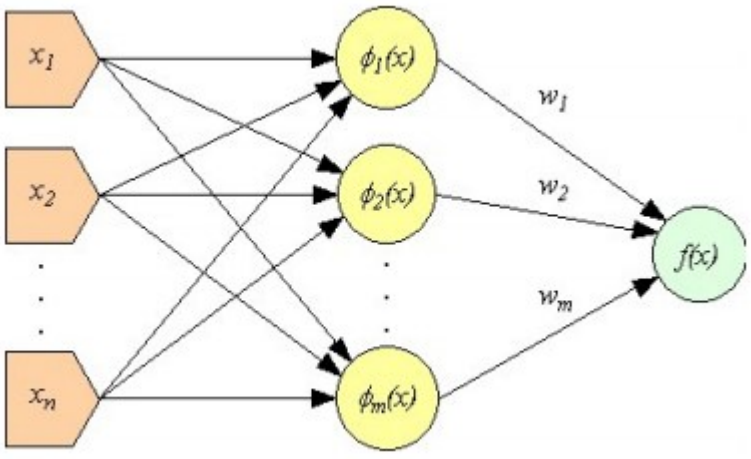

Figura. 4 Topología Redes Función Base Radial tomado de

http://sci2s.ugr.es/keel/pdf/keel/congreso/maeb09.pdf

La función de predicción de la red puede ser escrita como $f(x) = \sum_{i=1}^{m} w_i * \emptyset_m(x)$, donde $w_i$son los pesos de cada nodo de la capa oculta a la salida y $\emptyset_m(x)$ es la función radial de cada nodo en función de su radio y ancho puede escribirse matemáticamente como:

$$\emptyset_m(x) = e^{(-\beta * r^2)} \text{ y } r = \left(\sum_{i=1}^{n} (x_i(n) - c_{i,j})^2\right)^{1/2} para\ todo\ j = 1,2,\cdots,n$$

En la literatura se explican los diferentes métodos de aprendizajes de las redes de base radial, nosotros utilizamos la forma clásica en donde generamos los vectores de centros de forma aleatoria y se calculan los pesos de forma rápida aplicando la inversa de función radial al vector y,

Dada la función:

$f(x) = \sum_{i=1}^{m} w_i * \emptyset_m(x)$ Matricialmente $y = \emptyset w\ entonces\ w = \emptyset^{-1} y$

Este método tiene el riesgo de que la matriz no tenga inversa por tanto se aplica la teoría de regularización para perturbar la matriz mediante $\emptyset = \emptyset + \lambda I$ (Haykin, 1998) De esta manera sería un aprendizaje directo, provocando cambio a los pesos que están ubicados entre la capa oculta y la capa de salida.

2.3.4 Métodos de validación

En la literatura nosotros encontramos para machine learning el uso masivo de las técnicas de validación cruzadas (cross validation) como las más utilizadas por los investigadores para entrenamiento y validación del modelo.

Existen varios tipos de cross validation, la más utilizada es la de k-iteraciones o K-Folds, la cual consiste en dividir el conjunto de datos en k subconjunto. La técnica consiste en dejar uno de los subconjuntos como datos de validación y el resto (k−1) como datos de entrenamiento. El proceso de validación cruzada es repetido durante k iteraciones, con cada uno de los posibles subconjuntos de datos de validación. Finalmente, se selecciona la que mayor capacidad de generalización posea. (Arlot, 2010).

Cross validation asume que la data es independiente y distribuida idénticamente. Ambas asunciones podría no mantenerse para series de tiempo, como los datos están usualmente autocorrelacionados.

En la literatura se encontraron diferentes tipos de cross validation para series de tiempo, como lo muestra la figura 5.

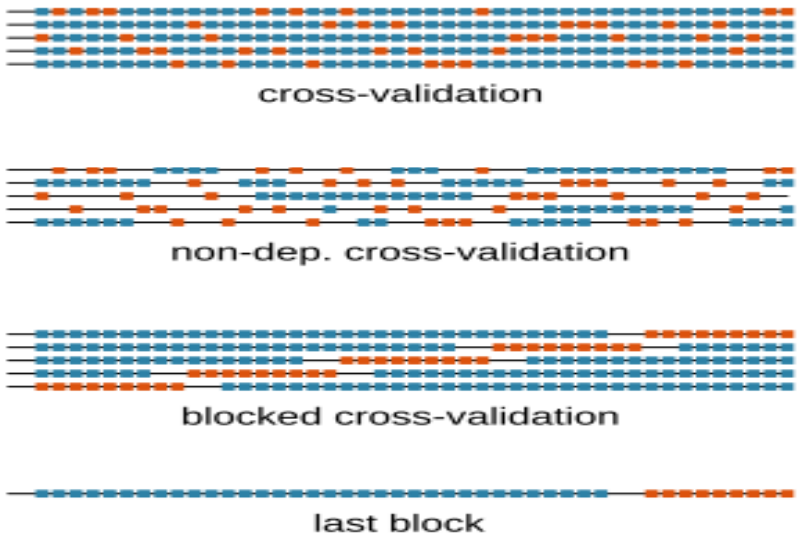

Figura. 5 Tipos de Cross Validation para series de tiempo.

En pronóstico de series de tiempo normalmente la última parte de la serie es reservada para testing. Usualmente este tipo de evaluación se le conoce como modelo de evaluación de último bloque. Según (Tashman, 2000), para series de tiempo existen inicialmente cuatro posibilidades para pronósticos:

- Fixed Origin: en este tipo de pronóstico, el conjunto de test no está disponible cuando se desarrolla el pronóstico, así cada pronóstico para el conjunto de pruebas tiene un horizonte diferente, dependiendo sobre esta distancia dentro del tiempo desde el conjunto de entrenamiento.

- Rolling Origin Update: El modelo es estimado usando el conjunto de entrenamiento, pero los valores pasados desde el conjunto de test son usados como entradas para el modelo. El modelo no es alterado durante este proceso.
- Rolling Origin Recalibration: en este tipo de pronóstico, el modelo es reajustado por cada predicción en el conjunto de prueba.
- Rolling Windows: como el Rolling Origin Recalibration el modelo es reajustado, pero el tamaño del conjunto de entrenamiento es fijo, de esta forma, los primeros valores son descartados dentro de los datos para entrenar el modelo. Este procedimiento es especialmente usado cuando la data no es estacionaria (y los cambios son lentos).

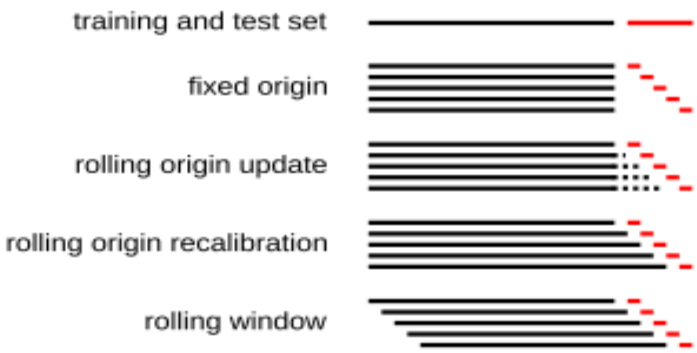

Figura. 6 Modelos de validación último bloque tomado de tesis "New Approaches in time series forecasting: methods, software and evaluation procedures" Christoph Norbert Bergmeir, 2013.

En esta investigación se soporta la validación del modelo basado en Rolling Origin Update, esta técnica de validación cruzada se basa en la selección del modelo teniendo en cuenta los

fundamentos teóricos en donde se asume dependencia entre las observaciones pasadas y el futuro, es decir, teniendo en cuenta la existencia de correlación entre el pasado y el futuro.

Rob J Hyndman sugiere para la predicción de series de tiempo el siguiente procedimiento de validación cruzada, según (Arlot, 2010).

1. Seleccione la observación en el periodo $k + i$ del conjunto de test y use las observaciones en el tiempo $1,2, \dots, k + i - 1$ para estimar el modelo de pronóstico. Compute el error de pronóstico en el periodo $k + i$.
2. Repita el paso anterior para $i = 1,2, \dots T - k$ donde T es el total número de observaciones.
3. Calcule la medida de exactitud del pronóstico basado en los errores obtenidos.

Este procedimiento es aplicable a pronóstico un paso adelante "One -step -ahead".

Para pronósticos multi (h)–step-ahead, el procedimiento es el siguiente:

1. Seleccione la observación en el periodo $k + h + i - 1$ del conjunto de test y use las observaciones en el tiempo $1,2, \dots, k + i - 1$ para estimar el modelo de pronóstico. Compute el h error de pronóstico en el periodo $k + h + i - 1$.
2. Repita el paso anterior para $i = 1,2, \dots T - k - h + 1$ donde T es el total número de observaciones.
3. Calcule la medida de exactitud del pronóstico basado en los errores obtenidos.

2.3.5 Evaluando la exactitud del pronóstico

El objetivo de las medidas de error, es obtener un claro y robusto resumen de la distribución del error. Es una práctica común calcular la medida de error calculando la función de perdida y computando el promedio. Dado y_t ser el valor observado y $\hat{y}_t$ el valor predicho, el error es calculado por $e_t = y_t - \hat{y}_t$.

Medidas dependiente de la escalas.

Las medidas de error estándar que encontramos, donde el error absoluto $AE_t = |y_t - \hat{y}_t|$ o error cuadrado $SE_t = (y_t - \hat{y}_t)^2$ es promediado dando lugar a error absoluto medio MAE o error cuadrático medio MSE y Raíz Cuadrático Medio RMSE (root mean squared error):

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t|$$

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}$$

Estas medidas de error son ampliamente usadas en la literatura, sin embargo son poco usadas para comparar los resultados de los pronósticos, en este trabajos se sugiere utilizar las medidas del error en porcentajes.

El porcentaje de error se puede representar como: $PE_t = 100\left|\frac{y_t - \hat{y}_t}{y_t}\right|$

Entonces el porcentaje de error medio absoluto puede expresarse como: $MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|100\frac{y_t - \hat{y}_t}{y_t}\right|$

El principal problema con respecto a estas medidas es que tienen que tratar con valores infinitos para $\mathrm{y_t} = 0$ o valores indefinidos para $\mathrm{y_t} = \hat{\mathrm{y}}_\mathrm{t} = 0$.

Las medidas sMAPE (Symmetric Mean Absolute Percentage Error) propuesta por (Makridakis, 1993), tratan de evitar estas situaciones. El cálculo para esta función de error es el siguiente:

$$sMAPE = \frac{1}{n}\sum_{t=1}^{n}\left|100\frac{y_t - \hat{y}_t}{m_t}\right| \quad m_t = \frac{|y_t| + |\hat{y}_t|}{2}$$

Pero persisten los problemas cuando $y_t = \hat{y}_t = 0$.

Hyndman and Koehler proponen una medida de error para evitar valores indeterminados e infinitos llamada SE (Scaled Erros). (R.J. Hyndman, 2006)

$$SE_t = \frac{y_t - \hat{y}_t}{\frac{1}{n-1}\sum_{i=2}^{n}|y_i - y_{i-1}|}$$

3. ESTADO DEL ARTE

Han sido pocos trabajos que tratan sobre el pronóstico de ocupación en los hoteles, muchos de los trabajos se derivan de los enfoques desarrollados para el pronóstico de las reservas en aerolíneas.

Existe un parecido entre el problema de predicción de las habitaciones y el de las sillas de un vuelo, sin embargo, existen variables en el problema del hotel diferente en el problema de aerolínea, por ejemplo él alargue y acorte de estadía son variables existentes en el hotel y no se presenta en el problema de la aerolínea.

Básicamente los enfoques para la predicción de la ocupación pueden ser agrupados en dos categorías, modelos de reservas históricos y modelos de reservas avanzados. Los modelos de reservas históricos consideran el pronóstico de ocupación basado en modelo de series de tiempo, básicamente el problema consiste en predecir la ocupación partiendo del conocimiento de la misma variable en periodos de tiempos anteriores sin tener en cuentas los datos de reservación. Los modelos basados en reservaciones consisten en usar los datos de las reservaciones realizadas,

utilizando el concepto de “Pick-Up”. Esto significa que dado K reservaciones para un día futuroT, se tiene la expectativa de “Pick-up” de N reservaciones desde ahora hasta T logrando un pronóstico de K + N para el día de llegada.

Por ejemplo (William P. Andrew , David A. Cranage , & Chau Kwor Lee, 1990) usaron el método ARIMA de Box-Jenkins y el método de suavizado exponencial Holt Winters, para pronosticar la ocupación mensual del hotel, ellos muestran que ambos métodos producen un bajo error cuadrático medio. En el trabajo de (Rajopadhye, 2001) Combina pronóstico a largo plazo mediante el método de Holt Winters con pronóstico a corto plazo utilizando datos de reservaciones realizadas, al final combina los pronósticos para obtener el valor final. Los métodos Holt Winter son populares en el modelado de series de tiempo, por ser métodos sencillos con resultados acertados.

Otros ejemplos lo encontramos en los métodos ARIMA los cuales son frecuentemente usados en el modelamiento de series de tiempo, ejemplos de la aplicación en el turismo lo encontramos en los trabajos de (Jiménez, Gazquez, & Sanchez, 2006), en este trabajo se realiza un análisis de la capacidad predictiva del método en una serie con estacionalidad procedente del sector turístico.

Otros trabajos se han encontrado con series turísticas donde se utilizan redes neuronales, por ejemplo (R. Law and N. Au, 1999) Estudia el pronóstico de la demanda de japoneses que viajaran a Hong Kong, los resultados de los experimentos muestran que las redes neuronales superan a los procedimientos de regresiones múltiples, promedio móviles, suavización exponencial.

Un ejemplo del uso del método basado en la información de reservas observadas "pick-up" lo encontramos en el trabajo de (Phumchusri, 2012) el cual desarrolla el pronóstico de habitaciones vendidas y utiliza en su modelo el pickup a 7,14,30 y 60 días de reservas, además incluye como variable independiente los días de la semana, en esta investigación se aplican varios modelos de pronósticos como lo son (El último día del año anterior, Promedio Móvil, Suavización exponencial, Método Pickup Aditivo y multiplicativo, Regresión Múltiple). Calculan varias medidas del error como MAD (error medio absoluto), MAPE (error porcentual medio absoluto).

En el trabajo de (Athanasius Zakhary, Amir F. Atiya, Hisham El-Shishiny, & Neamat El Gayar, 2009) proponen un modelo para el pronóstico de llegadas y ocupación utilizando la técnica de simulación Monte Carlos. Los métodos de Monte Carlos son algoritmos de computación fundamentado en técnicas matemáticas que se basan en la aplicación de simulaciones para calcular probabilidades heurísticamente, ampliamente usados en el campo de finanzas, gestión de proyectos, energía, manufactura, ingeniería, investigación y desarrollo, seguros, gas, petróleo y transporte. En este trabajo de investigación se considera como caso de estudio el problema del pronóstico de la demanda de habitaciones para el hotel plaza, Alexandria, Egipto. El propósito del modelo es dar mejores resultados que los enfoques existentes. El modelo propuesto tiene dos fases. La primera fase estima los parámetros relacionados en el proceso de reservación. En la segunda fase simulan los procesos de reservas en el tiempo, haciendo uso de los parámetros estimados obtenidos en la fase anterior.

En este estudio se comparan los resultados del método Monte Carlos con 5 técnicas:

1. Un modelo basado en reservas históricas, utilizando suavizado exponencial de Holt, se aplicó una estrategia de desestacionalización de la serie teniendo en cuenta la temporada y el día de la semana.
2. Pickup aditivo clásico utilizando promedio móvil.
3. Pickup aditivo avanzado utilizando promedio móvil.
4. Pickup aditivo clásico utilizando suavizado exponencial.
5. Pickup Multiplicativo clásico utilizando suavizado exponencial.

La medida utilizada para evaluar el desempeño de las técnicas fue el sMAPE (Symmetric Mean Absolute Percentage Error), el experimento se realizó para pronóstico semanal y diario, en todos los casos se obtuvo mejores resultados utilizando el método Simulación Monte Carlos.

Otros resultados interesantes en el pronóstico de ocupación utilizando métodos de pickup se presentan en el trabajo de (Athanasius Zakhary & Neamat El Gayar, A Comparative Study of the Pickup Method and its Variations Using a Simulated Hotel Reservation Data). En este estudio se realiza una comparación entre los siguientes métodos de pickup:

- Pickup aditivo, clásico, promedio simple.
- Pickup aditivo, clásico, Suavizado Exponencial.

- Pickup aditivo, Avanzado, promedio simple.
- Pickup aditivo, Avanzado, Suavizado Exponencial.
- Pickup Multiplicativo, clásico, promedio simple.
- Pickup Multiplicativo, clásico, Suavizado Exponencial.
- Pickup Multiplicativo, Avanzado, promedio simple.
- Pickup Multiplicativo, Avanzado, Suavizado Exponencial.

Se utilizan varias medidas de error dentro de las cuales relacionamos el MAE (Mean Absolute Error), MAPE (Mean Absolute Percentage Error), MSE (Mean Square Error), RMSE (Root Mean Square Error), NMAE (Mínimum Absolute Error Ratio), RMSE (Root Mean Square Error Ratio). El experimento se realiza utilizando (7, 15, 30 y 60 pasos adelante). En general los modelos pickup clásico superan en desempeño a los modelos de pickup avanzados. Por otro lado, el modelo de pickup multiplicativo, clásico y suavización exponencial han sido identificados como las mejores de las técnicas.

4. COMPARACIÓN DE TÉCNICAS Y EVALUACIÓN DE RESULTADOS.

4.1 Implementación

En esta sección se presentan y analizan los resultados del error de validación aplicando las cuatros técnicas de Machine Learning escogidas (Ridge Regression, Kernel Ridge Regression, Redes Neuronales Perceptron Multicapa y de Función Base Radial) en la predicción de la ocupación diaria.

Los datos utilizados son recolectados desde la base de datos de un hotel, de forma que los resultados experimentados están basados en información generada por el proceso real sin ningún tipo de transformación para su posterior análisis; la muestra es de 7 años de historia, inicia desde julio 1 del 2008 hasta junio 30 del 2014 para un total de 2191 días de ocupación en análisis. Adicionalmente son incluidas otras variables independientes en el modelado como por ejemplo los días de la semana, días festivo, meses del año, números de días festivo antes y después de la fecha de ocupación y por último se incluye información de las reservas realizadas con 7, 10, 15, 20, 30, 60, 90 días de anticipación.

La siguiente tabla muestra las variables recolectadas, las cuales son utilizadas en el modelado.

Tabla 1 Descripción de datos principales para la aplicación de las técnicas

Dato	**Descripción**
Fecha	Fecha de la ocupación
year	Año de ocupación
Month	Mes de ocupación
Day	Día de ocupación
Totalreservado7diasantes	Total Habitaciones Reservadas 7 días antes de la fecha de ocupación
Totalreservado10diasantes	Total Habitaciones Reservadas 10 días antes de la fecha de ocupación
Totalreservado15diasantes	Total Habitaciones Reservadas 15 días antes de la fecha de ocupación
Totalreservado20diasantes	Total Habitaciones Reservadas 20 días antes de la fecha de ocupación
Totalreservado30diasantes	Total Habitaciones Reservadas 30 días antes de la fecha de ocupación
Totalreservado60diasantes	Total Habitaciones Reservadas 60 días antes de la fecha de ocupación

Dato	Descripción
Totalreservado90diasantes	Total Habitaciones Reservadas 90 días antes de la fecha de ocupación
Festivo	Variable binaria para indicar si el día es festivo, toma los valores de 0: si el día no es festivo y 1: si el día es festivo
NúmeroFestivos7diasdespues	Número de festivos que existen entre la fecha de ocupación y 7 días después.
NúmeroFestivos7diasantes	Número de festivos que existen entre la fecha de ocupación y 7 días antes.
Temporada1alta2baja	Indicador de temporada toma valor de 1: si la temporada es alta 2: si la temporada es baja (este dato es basado en el conocimiento de los administradores del hotel y varía dependiendo entre hoteles y ciudades).
Lunes	Variable binaria que toma el valor de 1: si el día de ocupación es Lunes, 0: en caso contrario
Martes	Variable binaria que toma el valor de 1: si el día de ocupación es Martes, 0: en caso contrario
Miércoles	Variable binaria que toma el valor de 1: si el día de ocupación es Miércoles, 0: en caso contrario
Jueves	Variable binaria que toma el valor de 1: si el día de ocupación es Jueves, 0: en caso contrario
Viernes	Variable binaria que toma el valor de 1: si el día de ocupación es

Dato	Descripción
	Viernes, 0: en caso contrario
Sábado	Variable binaria que toma el valor de 1: si el día de ocupación es Sábado, 0: en caso contrario
Domingo	Variable binaria que toma el valor de 1: si el día de ocupación es Domingo, 0: en caso contrario
Enero	Variable binaria que toma el valor de 1: si el mes de ocupación es Enero, 0: en caso contrario
Febrero	Variable binaria que toma el valor de 1: si el mes de ocupación es Febrero, 0: en caso contrario
Marzo	Variable binaria que toma el valor de 1: si el mes de ocupación es Marzo, 0: en caso contrario
Abril	Variable binaria que toma el valor de 1: si el mes de ocupación es Abril, 0: en caso contrario
Mayo	Variable binaria que toma el valor de 1: si el mes de ocupación es Mayo, 0: en caso contrario
Junio	Variable binaria que toma el valor de 1: si el mes de ocupación es Junio, 0: en caso contrario
Julio	Variable binaria que toma el valor de 1: si el mes de ocupación es Julio, 0: en caso contrario
Agosto	Variable binaria que toma el valor de 1: si el mes de ocupación es Agosto, 0: en caso contrario

Dato	Descripción
Septiembre	Variable binaria que toma el valor de 1: si el mes de ocupación es Septiembre, 0: en caso contrario
Octubre	Variable binaria que toma el valor de 1: si el mes de ocupación es Octubre, 0: en caso contrario
Noviembre	Variable binaria que toma el valor de 1: si el mes de ocupación es Noviembre, 0: en caso contrario
Diciembre	Variable binaria que toma el valor de 1: si el mes de ocupación es Diciembre, 0: en caso contrario
Total ocupación	Corresponde al número de habitaciones ocupadas.

Una vez obtenidos los datos, se construyen los datasets utilizando tres esquemas, en el primer esquema los datasets utilizan como entradas las observaciones de la ocupación en días anteriores $x_{i+0}, x_{i+1}, \dots, x_{i+n-1}, ,$ y salida la observación x_{i+n}, donde x_i corresponde a la ocupación en el día i, y n es el número de rezagos utilizados; en el segundo esquema las entradas de los datasets están representadas por observaciones de la ocupación en días anteriores como en el primer esquema, adicionalmente se agregan otras variables de entradas a los datasets como los días de la semana, el número de festivos siete días antes y después del día de ocupación y la temporada; en el tercer esquema se utiliza información de las reservas realizadas con varios días de anticipación a la fecha de ocupación, también se incluyen el mes, días de la semana, temporada y número de festivos 7 días antes y después de la fecha de ocupación.

Los datasets son divididos en tres conjuntos (test, validación y entrenamiento), para el conjunto de test se reservan los 181 registros de la ocupación del año 2014, lo cuales corresponden al 8.26% de la muestra, una vez excluidos los registros para test se utilizan para validación los últimos 402 registros que corresponden al 20% del conjunto de entrenamiento y validación, dejando los registros restantes para entrenamiento. El entrenamiento y validación se realiza teniendo en cuenta el procedimiento de validación cruzada Rolling Origin Update, la predicción se realiza One-Step-ahead y multi (h)–step-ahead, con $h = 7$, la intención es predecir con 7 días de anticipación la ocupación del hotel; el desempeño de las técnicas se mide utilizando la medida MAPE (Mean Absolute Percentage Error) $= \frac{1}{n}\sum_{t=1}^{n}\left|100\frac{y_t-\hat{y}_t}{y_t}\right|$.

Para efectos de comparar los resultados se desarrolló un algoritmo con las siguientes características:

- Desarrollado en lenguaje de programación Python
- El Algoritmo permite a partir de un archivo texto, construir de forma automática cada datasets según el esquema seleccionado, y permite experimentar con diferentes parámetros como por ejemplo rezagos, grados de polinomios, factores de regularización, etc.
- Otra característica fundamentales es permitir realizar búsqueda exhaustiva del mejor modelo, basado en la combinación de parámetros experimentales y aplicando técnica de validación seleccionada. “El mejor de los modelos en cada caso es aquel con menor de error de validación MAPE(Mean Absolute Percentage Error)".

- MAPE= $\frac{1}{n}\sum_{t=1}^{n}\left|100\frac{y_t-\hat{y}_t}{y_t}\right|$.

Para el desarrollo de las técnicas se utilizaron las siguientes librerías:

Tabla 2 Librerías Python utilizadas en la investigación

Técnica	Librería
Ridge Regression	**Mlpy** Esta Liberia de Python Open Source, es construida a partir de Numpy, esta ofrece una amplia variedad implementaciones de algoritmos para machine learning. http://mlpy.sourceforge.net/ Para la construcción de modelos se utilizó la clase *class* mlpy.Ridge (*lmb=1.0*) donde *lmb* es el coeficiente ridge o factor de regularización. http://mlpy.sourceforge.net/docs/3.5/lin_regr.html#ridge-regression
Kernel Ridge Regression	**Mlpy** Para la construcción de modelos se utilizó la clase *class* mlpy.KernelRidge(*lmb=1.0*, *kernel=None*) http://mlpy.sourceforge.net/docs/3.5/nonlin_regr.html#kernel-ridge-regression Se utilizaron las siguientes clases para funciones de kernel:

Técnica	Librería
	class mlpy.KernelPolynomial(gamma=1.0, b=1.0, d=2.0) $(\mathbf{gamma}.x_1'x_2 + \mathbf{b})^d$ class mlpy.KernelGaussian(sigma=1.0) $e^{\left(-\|x_1-x_2\|^2/2\delta^2\right)}$
Redes Neuronales	ALGLIB http://www.alglib.net/#book Esta librería para análisis numérico, es soportada en varios lenguajes de programación como Python, c#, Pascal y VBA. Incluye varias características como lo son: Análisis de datos (clasificación, regresión incluyendo redes neuronales). Optimización, y búsqueda de soluciones no lineales, algebra lineal. En esta investigación se utilizaron los siguientes procedimientos para implementar redes neuronales. **Mlpcreate1** crea una red neuronal con función de activación lineal en la capa de salida y función Sigmoide en la capa oculta, utiliza bias. http://www.alglib.net/translator/man/manual.ipython.html#sub_mlpcreate1 En el entrenamiento utilice la función:

Técnica	Librería
	Mlptrainlbfgs http://www.alglib.net/translator/man/manual.ipython.html#sub_mlptrainlbfgs
Redes neuronales Función Base Radial	Código publicado por **Thomas Rückstieß** http://www.rueckstiess.net/research/snippets/show/72d2363e Se modificó el código implementando control de sobreajuste, utilizando método pseudoinversa.

De manera general el algoritmo construido obtiene la información a partir de archivos textos, con columnas separadas por coma, de tal forma que para cada esquema de datasets se utiliza un archivo específico, el cual debe ser ubicado en ("C:\datasets\Tesis\"), por ejemplo, si el análisis es por serie de tiempos debe proveerse el archivo ("C:\datasets\Tesis\SerieTime_Ocupacion.txt") , para el caso de series de tiempo con información adicional se utiliza el archivo ("C:\datasets\Tesis\Estadistica_ocupacion_tesis_dumis.txt"), y si el experimento se basa en información de reservas observadas se utiliza el archivo ("C:\datasets\Tesis\datos_pickup_addcancel.txt ").

La siguiente tabla muestra por cada archivo la información de campos utilizados

Tabla 3 Relación de columnas por tipos de archivos requeridos para aplicar el algoritmo de pronostico.

Archivo	Lista de columnas que conforman el archivo separadas por comas.
SerieTime_Ocupacion.txt	Fecha,year,Month,Day,Festivo,NumeroFestivos7diasdespues,NumeroFestivos7diasantes,Temporada1alta2baja,Lunes,Martes,Miercoles,Jueves,Viernes,Sabado,Domingo,Enero,Febrero,Marzo,Abril,Mayo,Junio,Julio,Agosto,Septiembre,Octubre,Noviembre,Diciembre,Totalocupacion,HabitacionesCanceladas,Resenoshows,TotalWallking,Totalreservadogeneral,TotalHabitaciones,e_TotalOcupacion,d_e_TotalOcupacion
Estadistica_ocupacion_tesis_dumis.txt	Fecha,Day,year,semana,Month,Festivo,NumeroFestivos7diasdespues,NumeroFestivos7diasantes,Temporada1alta2baja,Lunes,Martes,Miercoles,Jueves,Viernes,Sabado,Domingo,Enero,Febrero,Marzo,Abril,Mayo,Junio,Julio,Agosto,Septiembre,Octubre,Noviembre,Diciembre,Totalocupacion,HabitacionesCanceladas,Resenoshows,TotalWallking,Totalreservadogeneral,TotalHabitaciones
datos_pickup_addcancel.txt	Fecha,year,Month,Day,Totalreservado7diasantes_Lunes,Totalreservado7diasantes_Martes,Totalreservado7diasantes_Jueves,Totalreservado7diasantes_Miercoles,Totalreservado7diasantes_Viernes,Totalreservado7diasantes_Sabado,Totalreservado7diasantes_Domingo,Totalreservado10diasantes_Lunes,Totalreservado10diasantes_Martes,Totalreservado10diasantes_Miercoles,Totalreservado10diasante

Archivo	Lista de columnas que conforman el archivo separadas por comas.
	s_Jueves,Totalreservado10diasantes_Viernes,Totalreservado10diasantes_Sabado,Totalreservado10diasantes_Domingo,Totalreservado15diasantes_Lunes,Totalreservado15diasantes_Martes,Totalreservado15diasantes_Jueves,Totalreservado15diasantes_Miercoles,Totalreservado15diasantes_Viernes,Totalreservado15diasantes_Sabado,Totalreservado15diasantes_Domingo,Totalreservado20diasantes_Lunes,Totalreservado20diasantes_Martes,Totalreservado20diasantes_Miercoles,Totalreservado20diasantes_Jueves,Totalreservado20diasantes_Viernes,Totalreservado20diasantes_Sabado,Totalreservado20diasantes_Domingo,Totalreservado30diasantes_Lunes,Totalreservado30diasantes_Martes,Totalreservado30diasantes_Miercoles,Totalreservado30diasantes_Jueves,Totalreservado30diasantes_Viernes,Totalreservado30diasantes_Sabado,Totalreservado30diasantes_Domingo,Totalreservado60diasantes_Lunes,Totalreservado60diasantes_Martes,Totalreservado60diasantes_Miercoles,Totalreservado60diasantes_Jueves,Totalreservado60diasantes_Viernes,Totalreservado60diasantes_Sabado,Totalreservado60diasantes_Domingo,Totalreservado90diasantes_Lunes,Totalreservado90diasantes_Martes,Totalreservado90diasantes_Miercoles,Totalreservado90diasantes_Jueves,Totalreservado90diasantes_Viernes,Totalreservado90diasantes_Sabado,Totalreservado90diasantes_Domingo,Totalreservado7diasan

Archivo	Lista de columnas que conforman el archivo separadas por comas.
	tes,Totalreservado10diasantes,Totalreservado15diasantes,Totalreservado20diasantes,Totalreservado30diasantes,Totalreservado60diasantes,Totalreservado90diasantes,Totalreservado7diasantes_2,Totalreservado10diasantes_2,Totalreservado15diasantes_2,Totalreservado20diasantes_2,Totalreservado30diasantes_2,Totalreservado60diasantes_2,Totalreservado90diasantes_2,Festivo,NumeroFestivos7diasdespues,NumeroFestivos7diasantes,Temporada1alta2baja,Lunes,Martes,Miercoles,Jueves,Viernes,Sabado,Domingo,Enero,Febrero,Marzo,Abril,Mayo,Junio,Julio,Agosto,Septiembre,Octubre,Noviembre,Diciembre,Totalocupacion,HabitacionesCanceladas,Resenoshows,TotalWallking,Totalreservadogeneral,Totalreservascanceladas,TotalHabitaciones

Es importante aclarar que la información de las columnas HabitacionesCanceladas, Resenoshows, TotalWallking, Totalreservadogeneral y Totalreservascanceladas al no ser requeridas en este análisis, pueden ser registradas con valor cero, adicionalmente las columnas: Totalreservado7diasantes_2, Totalreservado10diasantes_2, Totalreservado15diasantes_2, Totalreservado20diasantes_2, Totalreservado30diasantes_2, Totalreservado60diasantes_2, Totalreservado90diasantes_2 corresponden a los valores de las reservas totales con 7, 10,15.20, 30, 60 y 90 días de anticipación al cuadrado, respectivamente.

Una vez construidos los archivos, se establecen los parámetros del experimento como por ejemplo, el tipo de técnica a utilizar, el número de datos reservados para entrenamiento, validación y pruebas, además por cada técnica se especifican sus parámetros, por ejemplo para el caso de redes neuronales artificiales, se debe establecer los rangos de neuronas capa oculta utilizada, adicionalmente, se debe establecer el rango de rezagos para los análisis de series de tiempos, grados de polinomios utilizados en la técnicas ridge regresión , y lista de coeficientes de regularización, también, se debe especificar la lista de parámetros de funciones de kernel utilizadas. En este trabajo se experimento utilizando entre 2 y 24 rezagos, polinomios de grado 1,2 y 3, factores de regularización de 0.01, 0.1, 0.15, 0.25 y 0.35. En todos los casos los parámetros se ajustaron teniendo en cuenta como criterio las combinaciones de los mismos que generaron un menor error sobre los datos de validación (MAPE), en las siguientes secciones se presentan los experimentos con los mejores resultados encontrados.

En la sección anexos se incluyen dos scripts Python, el primero corresponde al scripts utilizado para comparación de las técnicas y el segundo es el scripts aplicado sobre los datos de pruebas basados en el mejor modelo encontrado en esta investigación.

Los algoritmos utilizan la clase “clsparametrosexperimentales”, la cual provee una lista con los parámetros requeridos por cada una de las técnicas, los métodos (buildparameters, buildparametersAANs, buildparametersRBF) creados, permiten construir de forma rápida una combinación de todos los parámetros establecidos en la ejecución del código.

Al final de la ejecución del scripts, se graba un archivo en disco de nombre "MachineLearningSumary.txt" en la carpeta "C:\datasets\Tesis", el cual muestra por cada técnica el error de validación y parámetros utilizados, adicionalmente estos resultados son presentados por consola.

En caso del segundo scripts como lo mencionamos, el mejor modelo encontrado se aplica sobre los datos reservados para pruebas y al final se presenta un gráfico que compara el pronóstico contra lo observado del conjunto de validación, también se presentan los resultados las predicciones sobre el conjunto de test por consola y gráficamente.

A continuación presentemos los resultados en la aplicación de las técnicas en cada uno de los esquemas de construcción de datasets.

4.2 Análisis de resultados utilizando modelos de series de tiempo.

En este primer experimento los datasets son construidos teniendo en cuenta como variables independientes las observaciones de la ocupación en días anteriores, de esta forma, para predecir x_i se utilizan las siguientes variables independientes: $x_{i-n}, \dots, x_{i-2}, x_{i-1}, x_{i-n}^2, \dots, x_{i-2}^2, x_{i-1}^2, \dots, x_{i-n}^d, \dots, x_{i-2}^d, x_{i-1}^d$

Donde n es el número de rezagos y d es el grado del polinomio, la dificultad en el modelado consiste en encontrar la combinación de rezagos, grado polinomial y parámetros requeridos por cada técnica que permitan minimizar el error de validación.

En la aplicación de las técnicas Ridge Regression los datasets son construidos utilizando desde 2 hasta 24 rezagos, por cada número de rezagos se probó con polinomios grado 1 y 2; adicionalmente se utilizaron factores de regularización de $0.1, 0.15, 0.25, 0.35$, obteniendo como mejor error de validación en pronósticos un paso adelante un valor MAPE de 11.17998903, este valor se obtuvo utilizando 20 rezagos, polinomio grado 1 y factor de regularización ridge de 0.1.

La siguiente figura muestra el comportamiento gráfico entre la ocupación observada y el pronóstico de ocupación utilizando los datos de validación, las líneas rojas representan el valor predicho y las líneas negras representan el valor de la ocupación observada.

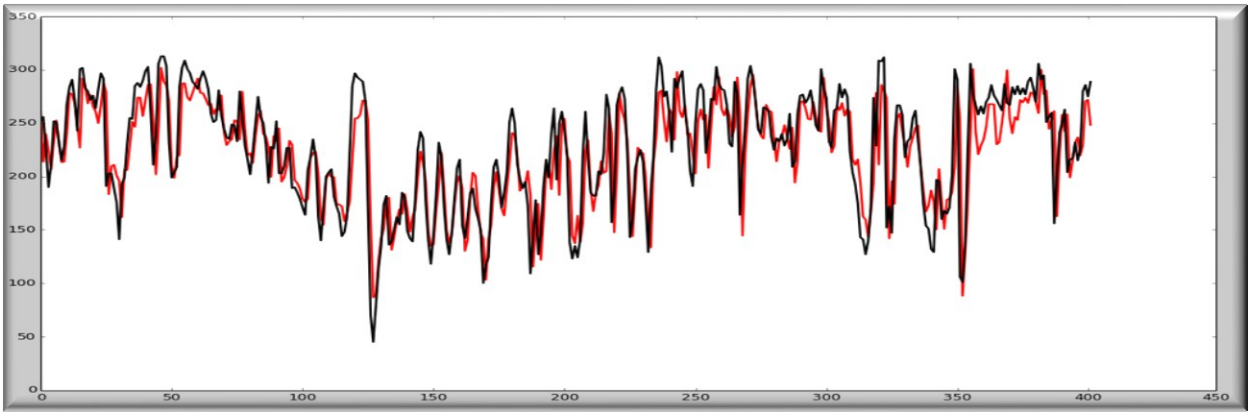

Figura. 7 Pronóstico vs ocupación diaria utilizando el mejor modelo Ridge Regression en pronósticos un paso adelante métodos de series de tiempo.

La aplicación de los pronósticos un paso adelante permite predecir la ocupación con un día de anticipación, sin embargo, esta característica puede ser desventajosa para la toma de decisiones,

en esta investigación enfocamos el interés a la predicción con h pasos adelante (multi (h)-step-ahead) con h=7.

Dada la serie $x_1, x_2, x_3, x_4, x_5, \ldots, x_{h+5}$, el primer registro del dataset para pronósticos con h pasos adelante y 5 rezagos es construido teniendo en cuenta las siguientes entradas $x_1, x_2, x_3, x_4, x_5, x_1^2, x_2^2, x_3^2, x_4^2, x_5^2, \ldots, x_1^d, x_2^d, x_3^d, x_4^d, x_5^d$ y salidas x_{h+5}, donde h corresponde al número de periodos adelante y d corresponde al grado del polinomio.

En la aplicación de técnica Ridge Regresión utilizando desde 2 rezagos hasta 24, polinomios grado 1 y 2, factores de regularización de 0.1,0.15,0.25 y 0.35 en pronósticos 7 pasos adelante, se encuentra con mejor desempeño el modelo construido con 22 rezagos, polinomio grado 1 y factor de regularización de 0.1, en este caso el error es de 19.5595665125.

La siguiente figura muestra el comportamiento gráfico entre la ocupación diaria observada y el pronóstico de la ocupación en los datos de validación utilizando pronósticos 7 pasos adelantes, las líneas rojas representan el valor predicho y las líneas negras representan el valor observado.

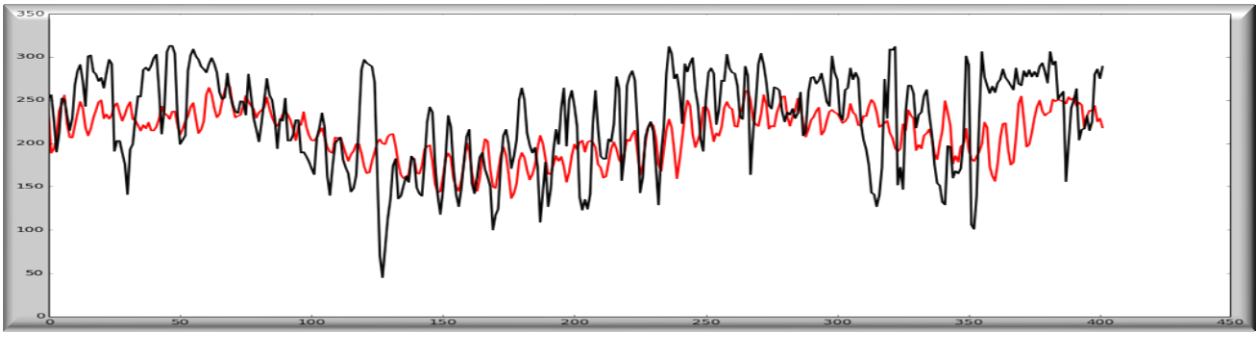

Figura. 8 Ocupación observada vs pronóstico de ocupación, mejor modelo Ridge Regression para pronósticos 7 pasos adelante, utilizando modelos de series de tiempo.

En la figura anterior, podemos notar que el pronóstico está alejado del valor observado, un pronóstico de 19% en un hotel de 100 habitaciones corresponde a un promedio de error de 19 habitaciones por debajo o por encima del valor real, esta métrica no es una medida deseada como soporte de toma de decisiones en los sistemas de Revenue Management por tanto en las siguientes secciones se evalúan otras técnicas buscando mejores predicciones.

Una vez utilizamos las técnicas Kernel Ridge Regression sobre la serie, utilizando desde 2 rezagos hasta 24, con Kernel Gaussiano definido por $e^{\left(-\|x_1-x_2\|^2/2\delta^2\right)}$, sigmas (δ) de 3, 4.5, 5, 1000, 2000, 2300 y 3000, el mejor desempeño se obtiene con 21 rezagos, δ = 2300 y factor de regularización de 0.015, el error de validación MAPE presentado es de 11.20557657. En el caso de pronósticos 7 pasos adelante el menor error de validación MAPE es de 19.68199533, con 23 rezagos, δ =3000 y coeficiente de regularización ridge de 0.15.

Los resultados aplicando la técnica Kernel Ridge Regression con kernel polinomial $(\gamma . x_1' x_2 + \beta)^d$, utilizando desde 2 rezagos hasta 24, con parámetros de kernel $d = 1{,}2 \quad \beta = 0.1$ y $\gamma = 1$, presentaron como menor error de validación en pronósticos un paso adelante un MAPE de 11.17998884, el cual se obtuvo con 20 rezagos, coeficiente de regularización ridge de 0.1 y parámetros $d = 1 \quad \beta = 0.1$ y $\gamma = 1$. En el caso de pronósticos 7 pasos adelante el mejor

error de validación MAPE es de 19.55956602, con 22 rezagos, coeficiente de regularización ridge 0.1 y parámetros d $= 1$ $\beta = 0.1$ y $\gamma = 1$.

En los experimentos con las técnicas de redes neuronales artificiales perceptron multicapa se utilizaron entre 10 y 100 nodos en la capa oculta y factor de regularización Weight Decay de 0.1, las redes neuronales se diseñan con función de activación sigmoide en capa oculta y función de activación lineal en capa de salida, además se utilizan bias en las capas de entradas y ocultas. En el entrenamiento se utilizaron desde 2 rezagos hasta 24, el número de rezagos corresponde al número de nodos de la capa de entrada, el entrenamiento es basado en el algoritmo L-BFGS. Para el caso de la aplicación de redes neuronales con función base radial se utilizaron entre 10 y 250 nodos en la capa oculta, el aprendizaje se realiza por el método de la pseudoinversa, se aplicaron factores de regularización de 0.1, 0.15 y 0.015, en la capa oculta se implementaron funciones de activación radial:

$\emptyset_m(x) = e^{(-\beta * r^2)}$ y $r = \left(\sum_{i=1}^{n}(x_i(n) - c_{i,j})^2\right)^{1/2}$ $para\ todo\ j = 1,2,\cdots,n$

Donde el parámetro β es de 0.00000015.

Los mejores errores de validación MAPE en pronósticos utilizando modelos de series de tiempos los encontramos utilizando Redes Neuronales Función Base Radial, en el caso de pronósticos un paso adelante el error encontrado fue de 11.23017994, este se obtuvo con 20 rezagos, 150 nodos en capa oculta y factor de regularización de 0.0015, en el caso de los pronósticos con 7 pasos adelante el menor error de validación fue de 19.72912448, utilizando 22 rezagos, 150 nodos capa oculta y factor de regularización de 0.0015.

Tabla 4 Mejores resultados aplicación de las técnicas en modelos de series de tiempos

Métodos de pronósticos	Horizonte	Técnicas con mejores resultados	MAPE	Parámetros utilizados
Análisis de resultados utilizando modelos de series de tiempo	One-step-ahead.	Ridge Regression	11.1799	20 rezagos, polinomio grado 1 y factor de regularización ridge de 0.1
		Kernel Ridge Regression - Kernel Gaussiano	11.2055	21 rezagos, δ = 2300 y factor de regularización de 0.015
		Kernel Ridge Regression - kernel polinomial	11.1799	20 rezagos, coeficiente de regularización ridge de 0.1 y parámetros $d = 1 \quad \beta = 0.1$ y $\gamma = 1$
		Redes Neuronales Función Base Radial	11.2301	20 rezagos, 150 nodos en capa oculta y factor de regularización de 0.0015, β de 0.00000015.
	7-step-ahead.	Ridge Regression	19.5595	22 rezagos, polinomio grado 1 y factor de regularización de 0.1
		Kernel Ridge Regression - Kernel Gaussiano	19.6819	23 rezagos, δ =3000 y coeficiente de regularización ridge de 0.15.
		Kernel Ridge Regression - kernel polinomial	19.55972	22 rezagos, coeficiente de regularización ridge 0.1 y parámetros $d = 1 \quad \beta = 0.1$ y $\gamma = 1$.
		Redes Neuronales Función Base Radial	19.7291	22 rezagos, 150 nodos capa oculta y factor de regularización de 0.0015, β de 0.00000015.

4.3 Análisis de resultados utilizando modelos de series de tiempo incluyendo variables adicionales.

En esta segunda fase de la investigación se realizan los experimentos de forma que los datasets se construyen teniendo en cuenta las observaciones de la ocupación histórica tal como se explicó en la sección 4.2, adicionalmente se incluyen las variables independientes días de semana, indicador de festivos, número de días festivos 7 días antes y después de la fecha de ocupación.

La siguiente tabla muestra las variables de entradas al modelo.

Tabla 5 Variables independientes método de series de tiempo adicionando variables adicionales.

Variables independientes	Descripción
Ocupación en $x_{i-Lags+1}, x_{i-Lags+2}, x_{i-Lags+k}$	Corresponde a cada valor de la ocupación en días inmediatamente anterior. Lags corresponde al total rezagos requeridos para encajar el modelo. K inicia desde 1 hasta Lags-1.
Festivo	Variable binaria para indicar si el día es festivo, toma los valores de 0: si el día no es festivo y 1: si el día es festivo
NúmeroFestivos7diasdespues	Número de festivos que existen entre la fecha de ocupación y 7 días después.

Variables independientes	Descripción
NúmeroFestivos7diasantes	Número de festivos que existen entre la fecha de ocupación y 7 días antes.
Temporada1alta2baja	Indicador de temporada toma valor de 1: si la temporada es alta 2: si la temporada es baja (este dato es basado en el conocimiento de los administradores del hotel y varia dependiendo entre hoteles y ciudades).
Lunes	Variable binaria que toma el valor de 1: si el día de ocupación es Lunes, 0: en caso contrario
Martes	Variable binaria que toma el valor de 1: si el día de ocupación es Martes, 0: en caso contrario
Miércoles	Variable binaria que toma el valor de 1: si el día de ocupación es Miércoles, 0: en caso contrario
Jueves	Variable binaria que toma el valor de 1: si el día de ocupación es Jueves, 0: en caso contrario
Viernes	Variable binaria que toma el valor de 1: si el día de ocupación es Viernes, 0: en caso contrario
Sábado	Variable binaria que toma el valor de 1: si el día de ocupación es Sábado, 0: en caso contrario
Domingo	Variable binaria que toma el valor de 1: si el día de ocupación es Domingo, 0: en caso contrario

Las variables Festivos, NúmeroFestivos7diasdespues, NúmeroFestivos7diasantes, son calculadas utilizando como soporte la Tabla 12 días festivos desde 2008 hasta 2014, relacionada en la sección anexos del documento.

El experimento se realiza utilizando para la construcción de los datasets desde 2 rezagos hasta 24, además se experimentan con polinomio grado 1 y 2 para los casos de los modelos Ridge Regression y Kernel Ridge Regression con kernel polinomial; los coeficientes de regularización utilizados son de $0.1, 0.15, 0.25, 0.35$, en el caso de Kernel Ridge Regression con función de kernel polinomial se utilizaron los parámetros $d = 1,2$ $\beta = 0.1$ y $\gamma = 1$, y sigmas de 3, 4.5, 5, 1000, 2000, 2300, 3000 para función de Kernel Gaussiano. El mejor desempeño en pronósticos un paso adelante arrojó un error de validación MAPE de 10.23517399 en este caso se utilizó 5 rezagos con la técnica Kernel Ridge Regression y función de kernel polinomial, los parámetros utilizados son: $d = 2$ $\beta = 0.1$ y $\gamma = 1$ y coeficiente de regularización ridge de 0.1, Para pronósticos 7 pasos adelantes los resultados arrojan con menor error de validación un MAPE de 19.6819, el cual se obtuvo con la técnica Kernel Ridge Regression , 23 rezagos, δ =3000 y factor de regularización de 0.15.

En los experimentos aplicando las técnicas de redes neuronales artificiales perceptron multicapa se utilizaron entre 10 y 100 nodos en la capa oculta, factor de regularización weight decay de 0.1, función de activación sigmoide en capa oculta y función de activación lineal en capa de salida, adicionalmente se incluyen bias en la capa oculta y de entrada; los datasets se construyen utilizando desde 2 rezagos hasta 24, el entrenamiento es basado en el algoritmo L-

BFGS. En la aplicación de las técnicas de redes neuronales de función base radial se utilizaron entre 10 y 250 nodos en la capa oculta, el aprendizaje se realiza por el método de la pseudoinversa y se utilizó los factores de regularización de 0.1, 0.15 y 0.015, en la capa oculta se implementaron funciones de activación radial:

$\emptyset_m(x) = e^{(-\beta * r^2)}$ y $r = \left(\sum_{i=1}^{n}(x_i(n) - c_{i,j})^2\right)^{1/2}$ $para\ todo\ j = 1,2,\cdots,n$

Donde el parámetro β para la función radial es de 0.00000015.

En este experimento el mejor desempeño para pronósticos un paso adelante arrojó un MAPE de 11.23586, el cual se obtuvo con una Red Neuronal de Función Base Radial, 20 rezagos, 150 nodos en capa oculta y factor de regularización de 0.0015, en el caso de los pronósticos con 7 pasos adelante el menor error de validación arrojó un MAPE de 19.71130 con una Red Neuronal de Función Base radial, 22 rezagos, 150 nodos capa oculta, β de 0.00000015 y factor de regularización de 0.0015.

Tabla 6 Mejores resultados aplicación de las técnicas en modelos de series de tiempos incluyendo información adicional

Métodos de pronósticos	Horizonte	Técnicas con mejores resultados	MAPE	Parámetros utilizados
Análisis de resultados utilizando modelos de series de tiempo incluyendo variables adicionales	One-step-ahead.	Kernel Ridge Regression - kernel polinomial	10.2351	5 rezagos , parámetros utilizados : $d = 2$ $\beta = 0.1$ y $\gamma = 1$ y coeficiente de regularización ridge de 0.1
		Redes Neuronales Función Base Radial	11.23586	20 rezagos, 150 nodos en capa oculta y factor de regularización de 0.0015
	7-step-ahead.	Ridge Regression	19.6819	23 rezagos, δ =3000 y factor de regularización de 0.15.
		Redes Neuronales Función Base Radial	19.7113	22 rezagos, 150 nodos capa oculta, β de 0.00000015 y factor de regularización de 0.0015.

En este experimento los resultados del error de validación MAPE para pronósticos un paso adelante es inferior a los resultados obtenidos en la sección 4.2, el valor MAPE es de 10.2351, este se obtuvo con la técnica Kernel Ridge Regression utilizando función de kernel polinomial, el dataset se construye con 5 rezagos incluyendo las variables adicionales mencionadas, coeficiente de regularización de 0.1 y parámetros $d = 2$ $\beta = 0.1$ y $\gamma = 1$. En el caso de pronósticos 7 pasos adelante, los modelos basados en métodos de series de tiempo muestran un mejor desempeño, el error de validación MAPE es de 19.5595665125, utilizando la técnica Ridge Regression con 22 rezagos, grado polinomial 1 y factor de regularización de 0.1.

Al final de estos experimentos se observa que adicionando al modelo de series de tiempo variables adicionales como el día de semana, temporada y festivos, se producen pronósticos de

mayor exactitud (cabe aclarar que la mejora en los pronósticos fue bastante modesta). En el problema del pronóstico de la ocupación diaria estas variables podrían aportar a los modelos mayor información del patrón generador de la ocupación, como por ejemplo el componente estacional y los puentes festivos, el menor error de validación en estos experimentos es de 10.23517399 un poco menor al resultado presentado por la técnica Ridge Regression de 11.17.

También se observa en la aplicación de pronósticos 7 pasos adelante, un mejor desempeño en los modelos basados en series de tiempo sin información adicional sobre su contrapartes que si incorporaban información adicional, arrojando un error de validación MAPE de 19.5595665125; el mejor modelo de pronóstico 7 pasos adelante se obtuvo utilizando la técnica Ridge Regression con 22 rezagos, grado polinomial 1 y factor de regularización de 0.1. Adicionalmente se observó un mejor desempeño de la técnica Kernel Ridge Regression frente a las Redes Neuronales tipo perceptron multicapa y redes Neuronales Función Base Radial en el modelado de series de tiempo. De igual forma los resultados muestran la superioridad de las Redes Neuronales Función Base Radial sobre las Redes Neuronales Perceptron Multicapa en el contexto de nuestra aplicación.

4.4 Análisis de resultados aplicando modelos de regresión vía reservas observadas

En esta sección aplicamos las cuatros técnicas de Machine Learning seleccionadas (Ridge Regression, Kernel Ridge Regression, Redes Neuronales Artificiales y de Función Base Radial), considerando para la construcción de los datasets la información de las reservas realizadas con 90, 60, 30, 20, 15, 10, y 7 días de anticipación, adicionalmente incluimos las variables meses del año, días de la semana, información de festivos y temporadas.

En este análisis decidimos incluir información de las reservas observadas con el objeto de encontrar mejores resultados en la predicción a los obtenidos en las secciones anteriores, tomando en consideración que las reservas realizadas dan una idea general de cómo puede ocuparse el hotel, sin embargo, considerando la variabilidad en el total de ocupación producidas por:

1. Walkin (clientes que compran sin reservas)
2. NoShows: clientes que reservan y no se presentan al hotel.
3. Cancelaciones y reservas de último momento
4. Alargues de estadías y acortes de estadía.

En este experimento se plantea estimar la ocupación con 7 días de anticipación de forma que una semana antes los administradores del hotel puedan contar con las predicciones como soporte a la toma de decisiones.

La siguiente tabla muestra las variables utilizadas en la construcción de los datasets.

Tabla 7 Variables utilizadas para construcción de los datasets en el análisis de regresión basado en la información de reservas observadas

Dato	Descripción
Totalreservado7diasantes	Total Habitaciones Reservadas 7 días antes de la fecha de ocupación.
Totalreservado10diasantes	Total Habitaciones Reservadas 10 días antes de la fecha de ocupación
Totalreservado15diasantes	Total Habitaciones Reservadas 15 días antes de la fecha de ocupación
Totalreservado20diasantes	Total Habitaciones Reservadas 20 días antes de la fecha de ocupación
Totalreservado30diasantes	Total Habitaciones Reservadas 30 días antes de la fecha de ocupación
Totalreservado60diasantes	Total Habitaciones Reservadas 60 días antes de la fecha de ocupación
Totalreservado90diasantes	Total Habitaciones Reservadas 90 días antes de la fecha de ocupación
Festivo	Variable binaria para indicar si el día es festivo, toma los valores de 0: si el día no es festivo y 1: si el día es festivo
NúmeroFestivos7diasdespues	Número de festivos que existen entre la fecha de ocupación y

Dato	Descripción
	7 días después.
NúmeroFestivos7diasantes	Número de festivos que existen entre la fecha de ocupación y 7 días antes.
Temporada1alta2baja	Indicador de temporada toma valor de 1: si la temporada es alta 2: si la temporada es baja (este dato es basado en el conocimiento de los administradores del hotel y varía dependiendo del hotel, países y ciudades).
Lunes	Variable binaria que toma el valor de 1: si el día de ocupación es Lunes, 0: en caso contrario
Martes	Variable binaria que toma el valor de 1: si el día de ocupación es Martes, 0: en caso contrario
Miércoles	Variable binaria que toma el valor de 1: si el día de ocupación es Miércoles, 0: en caso contrario
Jueves	Variable binaria que toma el valor de 1: si el día de ocupación es Jueves, 0: en caso contrario
Viernes	Variable binaria que toma el valor de 1: si el día de ocupación es Viernes, 0: en caso contrario
Sábado	Variable binaria que toma el valor de 1: si el día de ocupación es Sábado, 0: en caso contrario
Domingo	Variable binaria que toma el valor de 1: si el día de ocupación es Domingo, 0: en caso contrario

Dato	Descripción
Enero	Variable binaria que toma el valor de 1: si el mes de ocupación es Enero, 0: en caso contrario
Febrero	Variable binaria que toma el valor de |1: si el mes de ocupación es Febrero, 0: en caso contrario
Marzo	Variable binaria que toma el valor de 1: si el mes de ocupación es Marzo, 0: en caso contrario
Abril	Variable binaria que toma el valor de 1: si el mes de ocupación es Abril, 0: en caso contrario
Mayo	Variable binaria que toma el valor de 1: si el mes de ocupación es Mayo, 0: en caso contrario
Junio	Variable binaria que toma el valor de 1: si el mes de ocupación es Junio, 0: en caso contrario
Julio	Variable binaria que toma el valor de 1: si el mes de ocupación es Julio, 0: en caso contrario
Agosto	Variable binaria que toma el valor de 1: si el mes de ocupación es Agosto, 0: en caso contrario
Septiembre	Variable binaria que toma el valor de 1: si el mes de ocupación es Septiembre, 0: en caso contrario
Octubre	Variable binaria que toma el valor de 1: si el mes de ocupación es Octubre, 0: en caso contrario
Noviembre	Variable binaria que toma el valor de 1: si el mes de

Dato	Descripción
	ocupación es Noviembre, 0: en caso contrario
Diciembre	Variable binaria que toma el valor de 1: si el mes de ocupación es Diciembre, 0: en caso contrario
Total ocupación	Corresponde al número de habitaciones ocupadas.

Las siguientes tablas muestran los mejores resultados obtenidos en cuanto al error de validación en la aplicación de las 4 técnicas de machine learning escogidas. Los modelos Ridge Regression y kernel Ridge Regression se entrenaron con los coeficientes de regularización ridge de 0.1, 0.15, 0.25 y 0.35., adicionalmente se utilizaron los parámetros $d = 1{,}2$ $\beta = 0.1$ y $\gamma = 1$ para la función de kernel polinomial $(\gamma . x_1' x_2 + \beta)^d$, en el caso de la función de kernel gaussiano $e^{\left(-\|x_1 - x_2\|^2 / 2\delta^2\right)}$ se utilizaron los siguientes valores de sigmas (δ): 1, 10, 100,1000. , en el entrenamiento de redes neuronales artificiales perceptron multicapa se experimentó desde 10 nodos capa oculta hasta 200 nodos, factor de regularización weight decay de 0.1,0.2,0.3,0.4,0.5 y un máximo de 1000 iteraciones; el algoritmo utilizado para el entrenamiento es L-BFGS . en el caso de las redes artificiales función base radial se utilizó factores de regularización de 0.0015, 0.15, parámetros β =0.0000000001,0.0000000015,0.0000000150 y función de activación radial definida por:

$\emptyset_m(x) = e^{(-\beta * r^2)}$ y $r = \left(\sum_{i=1}^{n} (x_i(n) - c_{i,j})^2\right)^{1/2}$ $para\ todo\ j = 1{,}2{,}\cdots, n$

El entrenamiento se realiza con el método de la pseudoinversa.

Tabla 5 Resultados de experimento aplicando técnicas de Ridge Regression y Kernel Ridge Regression basado en modelos de reservas acumuladas con días de anticipación

Algoritmo	Función Kernel	Grado	Factor Regularización Ridge	MAPE	β
Ridge Regression		1	0.15	8.69622519	
Kernel Ridge Regression	Polinomial	1	0.1	8.69586851	0.1
Kernel Ridge Regression	Polinomial	1	0.15	8.69622452	0.1
Kernel Ridge Regression	Polinomial	1	0.25	8.69693643	0.1
Kernel Ridge Regression	Polinomial	1	0.35	8.69764658	0.1

Tabla 6 Resultados de experimento aplicando Redes Neuronales modelo basado en reservas acumuladas con días de anticipación.

Algoritmo	MAPE	Nodos Input	Nodos Hidden	β
Perceptron Multilayer	12.8917492	30	50	5.000000000000
Perceptron Multilayer	14.8481573	30	70	5.000000000000

Algoritmo	MAPE	Nodos Input	Nodos Hidden	β
Perceptron Multilayer	13.1829745	30	100	5.000000000000
Red Neuronal Function Base Radial	29.7113491	30	50	0.000000000100
Red Neuronal Function Base Radial	29.6734385	30	70	0.000000001500
Red Neuronal Function Base Radial	26.3208694	30	150	0.000000015000

En el experimento anterior, al aplicar las técnicas de Ridge Regression y kernel Ridge Regression construyendo los datasets utilizando polinomios de grado 2, notamos un aumento significativo en el número de columnas de entradas del dataset, nótese que para variables binarias como los meses de año, festivos y días de la semana no existe cambio alguno en los valores de las nuevas columnas al aumentar el grado, sin embargo, existe un incremento considerable en el número de columnas.

En este segundo experimento, el dataset es construido aumentando el grado del polinomio sobre las variables x_i (reservas realizadas con i días de anticipación), de forma que el dataset mantiene 35 columnas a diferencia del esquema anterior donde con polinomio grado 2 el dataset se construye con 58 columnas. La siguiente tabla muestra las columnas utilizadas en la construcción del dataset.

Tabla 8 Variables independientes Ridge Regression, utilizando polinomio grado 2, modelos basados en reservas observadas con días de anticipación.

Índice Columna	Descripción
[0]	Totalreservado7diasantes
[1]	Totalreservado10diasantes
[2]	Totalreservado15diasantes
[3]	Totalreservado20diasantes
[4]	Totalreservado30diasantes
[5]	Totalreservado60diasantes
[6]	Totalreservado90diasantes
[7]	Totalreservado7diasantes^2
[8]	Totalreservado10diasantes^2
[9]	Totalreservado15diasantes^2
[10]	Totalreservado20diasantes^2
[11]	Totalreservado30diasantes^2
[12]	Totalreservado60diasantes^2
[13]	Totalreservado90diasantes^2
[14]	Festivo
[15]	NumeroFestivos7diasdespues
[16]	NumeroFestivos7diasantes

Índice Columna	Descripción
[17]	Temporada1alta2baja
[18]	Lunes
[19]	Martes
[20]	Miércoles
[21]	Jueves
[22]	Viernes
[23]	Sábado
[24]	Domingo
[25]	Enero
[26]	Febrero
[27]	Marzo
[28]	Abril
[29]	Mayo
[30]	Junio
[31]	Julio
[32]	Agosto
[33]	Septiembre
[34]	Octubre
[35]	Noviembre
[36]	Diciembre

En la siguiente tabla presentamos los resultados aplicando las técnicas de Ridge Regression y Kernel Ridge Regression con función de Kernel Gaussiano, encontrando como mejor error de validación MAPE un valor de 8.2012, aplicando Ridge Regression con factor de regularización de 0.1.

Tabla 9 Resultados evaluación MAPE utilizando modelos basados en información de reservas observadas sobre datos de validación, aplicando las técnicas Ridge Regression y Kernel Ridge Regression.

Algoritmo	Función Kernel	δ	Factor Regularización Ridge	MAPE
Ridge Regression		1	0.1	8.201201258
Ridge Regression		1	0.15	8.201426401
Ridge Regression		1	0.25	8.201875871
Ridge Regression		1	0.35	8.202324258
Kernel Ridge Regression	Gaussiano	1	0.1	29.73061467
Kernel Ridge Regression	Gaussiano	300	0.1	27.05498485
Kernel Ridge Regression	Gaussiano	500	0.1	24.36636626
Kernel Ridge Regression	Gaussiano	750	0.1	21.48318292
Kernel Ridge	Gaussiano	1000	0.1	19.47177774

Algoritmo	Función Kernel	δ	Factor Regularización Ridge	MAPE
Regression				
Kernel Ridge Regression	Gaussiano	1	0.15	29.73061467
Kernel Ridge Regression	Gaussiano	300	0.15	27.0912133
Kernel Ridge Regression	Gaussiano	500	0.15	24.4231303
Kernel Ridge Regression	Gaussiano	7e50	0.15	21.56728098
Kernel Ridge Regression	Gaussiano	1000	0.15	19.5203994
Kernel Ridge Regression	Gaussiano	1	0.25	29.73061467
Kernel Ridge Regression	Gaussiano	300	0.25	27.16056995
Kernel Ridge Regression	Gaussiano	500	0.25	24.53812568
Kernel Ridge Regression	Gaussiano	750	0.25	21.72223919
Kernel Ridge Regression	Gaussiano	1000	0.25	19.63527553

Algoritmo	Función Kernel	δ	Factor Regularización Ridge	MAPE
Kernel Ridge Regression	Gaussiano	1	0.35	29.73061467
Kernel Ridge Regression	Gaussiano	300	0.35	27.22485562
Kernel Ridge Regression	Gaussiano	500	0.35	24.64416338
Kernel Ridge Regression	Gaussiano	750	0.35	21.86622392
Kernel Ridge Regression	Gaussiano	1000	0.35	19.74988846

Los resultados encontrados en este experimento muestran la superioridad de las técnica ridge regresión en el pronóstico de la ocupación, utilizando el esquema de construcción de datasets basado en las reservas observadas con 90, 60, 30, 20, 15,10 y 7 días de anticipación, adicionando información de festivos, temporadas, días de semana y meses del año. El error MAPE encontrado es de 8.2012 sobre el conjunto de validación y de 8.656144 sobre el conjunto de test.

4.5 Resultados sobre los datos de pruebas aplicando el mejor modelo encontrado.

Después de haber realizado el procedimiento de validación se encuentra que el modelo con mejor desempeño fue obtenido utilizando la técnica Ridge Regresión y un polinomio de grado 2, factor de regularización de 0.1 e información de reservas realizadas con 7, 10, 15, 20, 30, 60 y 90 días de anticipación, incluyendo información de festivos, días de la semana, temporadas y meses del año. El MAPE para dicho modelo fue de 8.2012.

La siguiente tabla muestra los coeficientes obtenidos por cada variable de entrada en el dataset.

Tabla 10 Coeficientes encontrados aplicando Ridge Regression, polinomio grado 2 en modelo basado en la información de reservas observadas con días de anticipación e información adicional.

I	Descripción	θ_i
[0]	Totalreservado7diasantes	1.199205859
[1]	Totalreservado10diasantes	0.321944857
[2]	Totalreservado15diasantes	-0.10532905
[3]	Totalreservado20diasantes	0.185487408
[4]	Totalreservado30diasantes	-0.138332234
[5]	Totalreservado60diasantes	-0.166403361
[6]	Totalreservado90diasantes	0.236738337
[7]	Totalreservado7diasantes^2	-0.000955095

I	Descripción	θ_i
[8]	Totalreservado10diasantes^2	-0.000506283
[9]	Totalreservado15diasantes^2	-0.0000661
[10]	Totalreservado20diasantes^2	-0.000607394
[11]	Totalreservado30diasantes^2	0.000607247
[12]	Totalreservado60diasantes^2	0.000421893
[13]	Totalreservado90diasantes^2	-0.00075207
[14]	Festivo	-6.656313827
[15]	NumeroFestivos7diasdespues	4.123584239
[16]	NumeroFestivos7diasantes	0
[17]	Temporada1alta2baja	-0.851522024
[18]	Lunes	-10.66943998
[19]	Martes	-11.00994349
[20]	Miércoles	2.224852204
[21]	Jueves	10.67036592
[22]	Viernes	5.528722851
[23]	Sábado	5.331033543
[24]	Domingo	-2.075591052
[25]	Enero	3.958494882
[26]	Febrero	-2.106205326
[27]	Marzo	-3.40569802

I	Descripción	θ_i
[28]	Abril	-3.816596667
[29]	Mayo	-2.57385578
[30]	Junio	-1.239637036
[31]	Julio	-1.743962338
[32]	Agosto	-0.10601468
[33]	Septiembre	6.180688018
[34]	Octubre	3.060090607
[35]	Noviembre	4.732522983
[36]	Diciembre	-2.939826642

El siguiente gráfico muestra el pronóstico de la ocupación sobre los 402 datos del conjunto de validación, con líneas rojas se muestra el gráfico del pronóstico de ocupación y líneas negras se presenta la ocupación real observada.

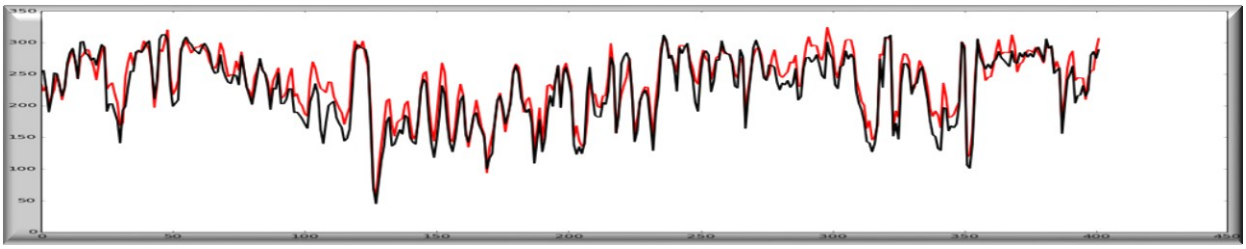

Figura. 9 Gráfico de ocupación observada vs pronóstico de ocupación sobre los datos de validación utilizando el mejor modelo encontrado.

El modelo es utilizado para predecir la ocupación sobre los datos de pruebas, los resultados arrojan un MAPE de 8.616544, a continuación se presentan los resultados del pronóstico de la ocupación y el valor observado sobre cada registro de prueba.

Tabla 11 Resultados pronósticos sobre datos de pruebas

Year	Month	Day	Ocupación	Pronóstico	MAPE
2014	1	1	254	284.69241	12.083625
2014	1	2	260	286.67808	10.260800
2014	1	3	303	318.97755	5.273120
2014	1	4	309	308.06346	0.303087
2014	1	5	295	292.88295	0.717645
2014	1	6	286	279.88089	2.139548
2014	1	7	303	298.34409	1.536603
2014	1	8	305	303.98220	0.333704
2014	1	9	305	307.15706	0.707233
2014	1	10	302	308.92406	2.292735
2014	1	11	269	271.89979	1.077991
2014	1	12	253	267.15414	5.594521
2014	1	13	225	233.86919	3.941863
2014	1	14	215	238.45945	10.911373
2014	1	15	214	242.28577	13.217648

Year	Month	Day	Ocupación	Pronóstico	MAPE
2014	1	16	277	287.11138	3.650319
2014	1	17	296	310.10789	4.766179
2014	1	18	299	309.69231	3.576022
2014	1	19	281	291.03198	3.570101
2014	1	20	277	284.91260	2.856533
2014	1	21	294	297.28396	1.116994
2014	1	22	296	301.29604	1.789201
2014	1	23	300	314.24052	4.746841
2014	1	24	299	302.07749	1.029261
2014	1	25	302	295.64237	2.105176
2014	1	26	308	291.88562	5.231943
2014	1	27	308	291.12669	5.478348
2014	1	28	304	290.19044	4.542619
2014	1	29	285	295.08213	3.537590
2014	1	30	270	289.77213	7.323012
2014	1	31	191	228.18514	19.468661
2014	2	1	214	238.14372	11.282110
2014	2	2	205	218.70177	6.683789
2014	2	3	227	239.87158	5.670299
2014	2	4	241	257.66501	6.914942

Year	Month	Day	Ocupación	Pronóstico	MAPE
2014	2	5	270	277.78895	2.884796
2014	2	6	294	287.12455	2.338588
2014	2	7	272	269.52782	0.908888
2014	2	8	273	273.91944	0.336791
2014	2	9	257	259.86873	1.116236
2014	2	10	227	238.73667	5.170338
2014	2	11	238	255.32046	7.277505
2014	2	12	260	256.98726	1.158747
2014	2	13	251	253.76700	1.102392
2014	2	14	277	267.59573	3.395042
2014	2	15	263	256.45095	2.490134
2014	2	16	248	253.87551	2.369159
2014	2	17	250	249.77902	0.088390
2014	2	18	210	226.75213	7.977203
2014	2	19	201	229.16571	14.012789
2014	2	20	240	267.21523	11.339680
2014	2	21	249	279.29746	12.167656
2014	2	22	263	285.22981	8.452399
2014	2	23	209	244.04267	16.766828
2014	2	24	233	243.06730	4.320730

Year	Month	Day	Ocupación	Pronóstico	MAPE
2014	2	25	249	251.95875	1.188253
2014	2	26	240	236.79885	1.333811
2014	2	27	227	234.40451	3.261897
2014	2	28	178	213.14099	19.742127
2014	3	1	227	224.65495	1.033060
2014	3	2	235	228.06991	2.948975
2014	3	3	193	182.82814	5.270391
2014	3	4	151	156.87673	3.891871
2014	3	5	159	174.68481	9.864660
2014	3	6	194	224.62244	15.784765
2014	3	7	205	231.83233	13.088943
2014	3	8	203	224.38130	10.532658
2014	3	9	186	203.34510	9.325325
2014	3	10	159	176.39726	10.941671
2014	3	11	160	172.41186	7.757414
2014	3	12	165	177.32994	7.472693
2014	3	13	223	227.22762	1.895795
2014	3	14	241	250.44429	3.918792
2014	3	15	242	238.26221	1.544542
2014	3	16	232	241.72586	4.192182

Year	Month	Day	Ocupación	Pronóstico	MAPE
2014	3	17	175	199.58796	14.050261
2014	3	18	198	212.29214	7.218252
2014	3	19	192	204.65278	6.589991
2014	3	20	189	200.12561	5.886564
2014	3	21	194	205.81091	6.088099
2014	3	22	243	227.29694	6.462163
2014	3	23	248	224.69181	9.398463
2014	3	24	202	214.57585	6.225670
2014	3	25	231	250.31503	8.361485
2014	3	26	251	270.70623	7.851086
2014	3	27	282	282.54276	0.192469
2014	3	28	258	275.53157	6.795182
2014	3	29	246	255.88265	4.017337
2014	3	30	193	213.91843	10.838565
2014	3	31	142	171.31079	20.641400
2014	4	1	141	159.59277	13.186360
2014	4	2	144	158.17701	9.845148
2014	4	3	180	200.70564	11.503136
2014	4	4	189	214.98554	13.748961
2014	4	5	200	224.26824	12.134122

Year	Month	Day	Ocupación	Pronóstico	MAPE
2014	4	6	235	240.41356	2.303644
2014	4	7	196	199.30224	1.684816
2014	4	8	188	185.04563	1.571471
2014	4	9	162	153.62706	5.168480
2014	4	10	134	126.54490	5.563505
2014	4	11	123	151.41900	23.104878
2014	4	12	140	153.47863	9.627592
2014	4	13	155	167.67278	8.175987
2014	4	14	189	204.23297	8.059772
2014	4	15	155	183.23210	18.214261
2014	4	16	185	223.67583	20.905855
2014	4	17	288	303.04284	5.223210
2014	4	18	281	296.88396	5.652656
2014	4	19	196	238.49836	21.682838
2014	4	20	56	77.31167	38.056551
2014	4	21	72	87.59849	21.664567
2014	4	22	134	154.44627	15.258410
2014	4	23	182	206.10930	13.246869
2014	4	24	260	285.56563	9.832934
2014	4	25	266	290.49740	9.209548

Year	Month	Day	Ocupación	Pronóstico	MAPE
2014	4	26	247	276.05419	11.762829
2014	4	27	198	224.16592	13.215113
2014	4	28	136	176.03474	29.437311
2014	4	29	120	159.07585	32.563206
2014	4	30	124	161.73076	30.428035
2014	5	1	132	168.85288	27.918851
2014	5	2	145	174.29058	20.200400
2014	5	3	168	205.61330	22.388866
2014	5	4	139	146.94513	5.715921
2014	5	5	156	169.70493	8.785211
2014	5	6	202	199.19206	1.390071
2014	5	7	206	191.55505	7.012112
2014	5	8	217	233.16986	7.451551
2014	5	9	176	212.47175	20.722585
2014	5	10	249	218.26990	12.341404
2014	5	11	227	196.55880	13.410221
2014	5	12	150	160.40318	6.935456
2014	5	13	186	192.58949	3.542738
2014	5	14	211	248.03920	17.554125
2014	5	15	262	256.41378	2.132144

Year	Month	Day	Ocupación	Pronóstico	MAPE
2014	5	16	244	239.96569	1.653407
2014	5	17	270	267.81117	0.810677
2014	5	18	163	178.28977	9.380229
2014	5	19	153	162.33609	6.102017
2014	5	20	184	183.43882	0.304991
2014	5	21	189	201.58550	6.658993
2014	5	22	233	251.30293	7.855333
2014	5	23	244	258.90280	6.107707
2014	5	24	201	228.18660	13.525671
2014	5	25	197	217.96601	10.642644
2014	5	26	162	184.98313	14.187117
2014	5	27	157	181.66418	15.709670
2014	5	28	128	148.80101	16.250790
2014	5	29	142	183.53193	29.247840
2014	5	30	134	165.47825	23.491233
2014	5	31	156	182.74235	17.142530
2014	6	1	135	151.32658	12.093764
2014	6	2	79	92.76842	17.428377
2014	6	3	103	123.00208	19.419499
2014	6	4	153	163.59103	6.922241

Year	Month	Day	Ocupación	Pronóstico	MAPE
2014	6	5	236	214.02841	9.309996
2014	6	6	241	244.34576	1.388282
2014	6	7	247	261.51653	5.877137
2014	6	8	210	232.86280	10.887045
2014	6	9	195	211.77879	8.604507
2014	6	10	173	194.43809	12.391957
2014	6	11	173	183.52831	6.085729
2014	6	12	184	196.95992	7.043435
2014	6	13	183	202.86685	10.856205
2014	6	14	178	210.82244	18.439571
2014	6	15	162	190.65202	17.686430
2014	6	16	161	187.47932	16.446781
2014	6	17	191	190.28709	0.373250
2014	6	18	260	263.67604	1.413860
2014	6	19	275	303.63854	10.414014
2014	6	20	283	303.03589	7.079819
2014	6	21	275	297.93843	8.341248
2014	6	22	240	251.07131	4.613047
2014	6	23	198	213.39301	7.774245
2014	6	24	260	257.71114	0.880331

Year	Month	Day	Ocupación	Pronóstico	MAPE
2014	6	25	278	284.11059	2.198055
2014	6	26	286	291.30255	1.854039
2014	6	27	270	285.48373	5.734715
2014	6	28	283	274.37214	3.048715
2014	6	29	276	253.46414	8.165165
2014	6	30	179	176.84141	1.205902
				MAPE	8.616544

El siguiente gráfico muestra el pronóstico de la ocupación con líneas rojas y la ocupación observada líneas negras sobre los datos de pruebas.

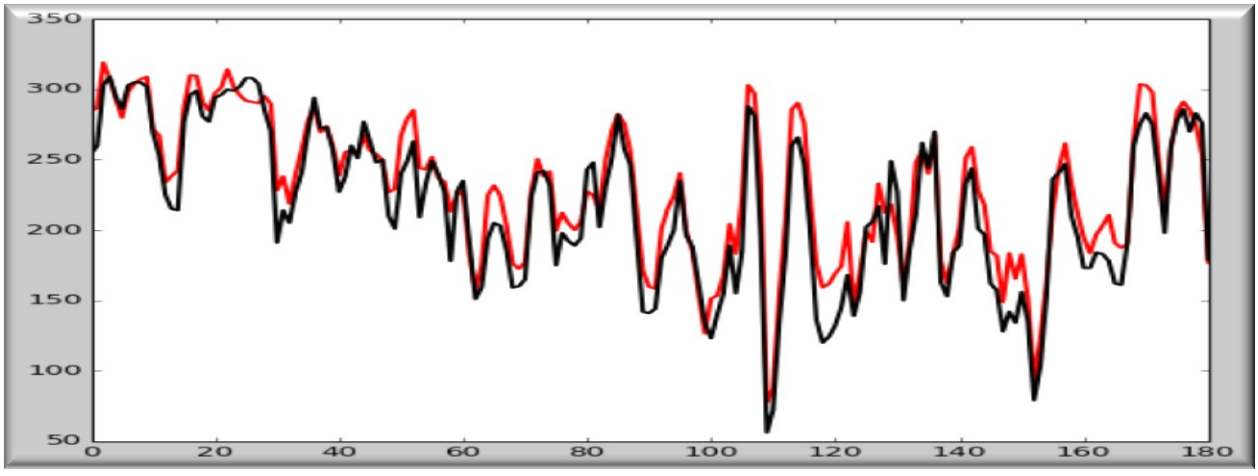

Figura. 10 Gráfico de pronóstico de ocupación vs reservas observadas, resultados utilizando mejor modelo encontrado sobre los datos de pruebas.

Los resultados encontrados en esta investigación arrojan un modelo con menor error de predicción que los obtenidos por los administradores del hotel donde se realizó este trabajo, el modelo de esta investigación predice la ocupación con un MAPE de 8.2 % sobre los datos de validación, 8.6 % sobre datos de prueba y 7 días de anticipación, mientras que en el hotel se observan cifras entre 13% y 19% de errores en pronósticos, utilizando técnicas tradicionales (métodos de descomposición, métodos ingenuos) respectivamente. Estos resultados encontrados ofrecen una mayor precisión en el pronóstico, permitiendo con esto un mayor apoyo en el proceso de toma decisiones. Por otra parte en la revisión literaria encontramos investigaciones como el trabajo de (Phumchusri, 2012), el cual obtiene un MAPE de 8.5% con 7 días de anticipación, aplicando regresión múltiple en el problema de pronóstico de la ocupación, en su trabajo compara los métodos de pickup y modelos ingenuos (el valor del mismo día del último año) los cuales arrojan un MAPE de 9.65 y 21.89 respectivamente. Por otro lado, dentro de la literatura podemos encontrar investigaciones que tratan con problemas de predicciones de series de tiempo con errores bajos, sin embargo no se realiza una comparativa debido a que en muchos de estos casos se puede apreciar el patrón de comportamiento de la serie, como es el caso de "Serie pasajeros de una aerolíneas" con un comportamiento marcado y patrón definido, las aplicación de las técnicas probablemente podrían encontrar un error de pronósticos muy por debajo de los presentados en este trabajo.

5. CONCLUSIONES Y TRABAJOS FUTUROS

En este trabajo se compararon las técnicas de Machine Learning Ridge Regression, Kernel Ridge Regression y Redes Neuronales (Perceptron Multicapa y Redes Neuronales Función Base Radial) con miras a identificar cuál ofrece mejores resultados en la predicción de la demanda de habitaciones y ocupación en el sector hotelero, los resultados se analizaron utilizando metodología de series de tiempos, series de tiempos con variables adicionales como festivos, día de la semana y análisis de Regression utilizando información de pickup (reservas realizadas con anterioridad).Los resultados encontrados muestran la superioridad de las técnica Ridge Regression en el pronóstico de la ocupación, utilizando el esquema de construcción de datasets basado en las reservas observadas con 90, 60, 30, 20, 15, 10 y 7 días de anticipación, adicionando información de festivos, temporadas, días de semana y meses del año. El error MAPE encontrado es de 8.2012 sobre el conjunto de validación y de 8.656144 sobre el conjunto de test, estos resultados son muy aproximados por tanto es buen estimador de que no existe presencia de overfitting.

Otros hallazgos de esta investigación muestran que adicionando al modelo de series de tiempo variables adicionales como el día de semana, temporada y festivos, se producen pronósticos de mayor exactitud (cabe aclarar que la mejora en los pronósticos fue bastante modesta), además se observa un mejor desempeño de la técnica Kernel Ridge Regression frente a

las Redes Neuronales tipo perceptron multicapa y redes Neuronales Función Base Radial en el modelado de series de tiempo. De igual forma los resultados muestran la superioridad de las Redes Neuronales Función Base Radial sobre las Redes Neuronales Perceptron Multicapa en el contexto de nuestra aplicación.

El pronóstico de ocupación hotelera es un problema que puede ser tratado aplicando técnicas de Machine Learning especialmente la técnica de Ridge Regresión que probaron ser capaces de predecir con un grado de acierto satisfactorio, apoyando la toma de decisiones en la industria hotelera.

En general, en nuestro país estamos en mora de aplicar de manera más amplia la inteligencia artificial en el campo hotelero. Con los avances en la tecnología, la alta capacidad de los equipos de cómputo y lenguajes especializados de programación, como por ejemplo Python que ofrecen librerías de uso libre con algoritmos de Machine Learning, es más asequible la aplicación de este tipo de técnicas. Logrando con esto soluciones que pueden impactar de manera sustancial la competitividad en la industria hotelera.

Para trabajos futuros, se sugiere incluir en los experimentos otras variables independientes asociadas con eventos y días especiales, diferentes a días festivos, con el objeto de evaluar si estas variables son representativas y puedan contribuir en la minimización del error de validación atrapando parte de patrón generador de la ocupación diaria, referente a conciertos, cumbres, festivales, días cívicos, etc.

Un aspecto importante sugerido para trabajos futuros es implementar procedimientos para selección de parámetros en la construcción de modelo basado en el error de validación (Wrapper Feature Selection), tal es el caso del PSO (Particle Swarm Optimization). Este procedimiento puede ser aplicado para la estimación del factor de regularización, sigma (kernel gaussiano), beta y gamma (kernel polinomial), todo esto con el objetivo de hacer una búsqueda más inteligente de los parámetros del modelo, logrando ser mas eficiente a diferencia de una búsqueda exhaustiva que en algunas ocasiones por la cantidad de parámetros y combinaciones de los mismos pueden llegar a tomar un tiempo demasiado alto dificultando con esto la posibilidad de obtener información oportuna para la toma de decisiones.

REFERENCIAS

Andrew, C. A. (1990). Forecasting hotel occupancy rates with time series models: an empirical analysis.

Arlot, S. (2010). A survey of cross-validation procedures. 4*, 65-66.*

Braun, M. R. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. 586-591. (I. Press, Ed.) Proceedings of the IEEE International Conference on Neuronal Networks.

C. Leung, H. W. (2010). On the selection of weight decay parameter for faulty networks. IEEE Transactions on Neuronal Networks, 8(21)*, 1232-1244.*

Castillo, I. J. (2012). Implementación de ridge regresión como una convolución para el reconocimiento de eventos de series temporal y video-peliculas.

D. M. Ortíz, F. V. (2007). Una comparación entre Estarategias Evolutivas y RPROP para la estimación de Redes Neuronales. Avances en Sistemas e Informática., 4, 2*, 135-144.*

El Gayar, N. &.-S. (2008). A proposed decision support model for hotel room revenue management.

F. A. Villa, J. D. (2008). Una aproximación a la regularización de redes cascada-correlación para la predicción de series de tiempos. Investigación Operacional, 28*, 151-161.*

Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. 2*, 183-192.*

Gökşen, S. (Julio de 2012). Implementing Revenue Management. Amsterdam.

Haykin, S. (1998). Neural Networks A Comprehensive Foundation. 2*, 245. Pearson Education.*

Hinton, G. (1989). Connectionist learning procedures. Artificial Intelligent(40)*, 185-243.*

Hoerl, A. E. (Feb de 1970). Ridge Regression:Applications to Nonorthogonal Problems. 12(1)*, 69-82. (Technometrics, Ed.)*

Jiménez, J., Gazquez, J., & Sanchez. (2006). La Capacidad Predictiva de los Modelos Box-Jenkis (ARIMA) y Holts Winters.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, 2(12)*, 1137-1142.*

Lee. (1990). Airline Reservations Forecasting. Prentice Hall, Inc., Erewhon.

M. Ghiassi, H. S. (2005). A dynamic artificial neural network model for forecasting time series events. International Journal of Forecasting, 21(2)*, 341-362.*

Ma, V. C. (2009). Another Look at statistical learning theory ans regularization. Neuronal Networks, 7(22)*, 958-969.*

Makridakis. (1993). Accuracy measures: theoretical and practical concerns. 527-529.

Neamat El Gayar, & Abdeltawad M.A. Hendawi. (s.f.). A Proposed Decision Support Model for Hotel Revenue Managment.

Phumchusri, N. (2012). Hotel Room Demand Forecasting via Observed Reservation Information.

R. Law and N. Au. (1999). A neural network model to forecast Japanese demand for travel to Hong Kong.

R.J. Hyndman, A. K. (2006). Another look at measures of forecast accuracy, International Journal of Forecasting. 22, 4.

Rajopadhye. (2001). Forecasting Uncertain Hotel Room Demand.

Riedmiller, M. (1994). Advanced supervised learning in multi-layer perceptrins from backpropagation to adaptive learning algorithms. 16*, 265-278. Computer Standars and Interfases.*

S. Arlot, A. C. (2010). A survey of cross-validation procedures for model selection, Statistics Surveys 4. 40–79.

S. Haykin, N. N. (1999). New Jersey: Prentice Hall.

Smit, M. (1993). Neuronal Networks for Statistical Modeling. John Wiley & Sons. New York, USA.

T. Masters, P. n. (1993). New York: Academic Press.

Tashman, L. (2000). Out-of-sample tests of forecasting accuracy: an analysis and review, International Journal of Forecasting. 16, 4*, 437–450.*

Weatherford, L. R, & Kimes, S. E. (January de 2003). A comparison of forecasting. International Journal of Forecasting 99 (19), S.E*, 401-415.*

Webos, P. (1974). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. PhD thesis, Harvard University, Cambridge, MA.

Zhang, J. (2009). "Risk Minimization". Statistical Learning Theory. http://www.stat.purdue.edu/~jianzhan/STAT598Y/NOTES/slt02.pdf.

ANEXOS

Tabla 12 días festivos desde 2008 hasta 2014

Día	**Meses**									
Año/Festivo	**1**	**3**	**4**	**5**	**6**	**7**	**8**	**10**	**11**	**12**
2008										
Año Nuevo	1									
Batalla de Boyacá							7			
Corpus Christi				26						
Día de la Ascensión				5						
Día de la Independencia						20				
Día de la Inmaculada Concepción										8
Día de la Raza								13		
Día de los Reyes Magos	7									
Día de Navidad										25
Día de San José		24								
Día del Trabajo				1						
Domingo de Ramos		16								
Domingo de Resurrección		23								
Independencia de Cartagena									17	

Jueves Santo		20								
La asunción de la Virgen							18			
Sagrado Corazón					2					
San Pedro y San Pablo					30					
Todos los Santos									3	
Viernes Santo		21								
2009										
Año Nuevo	1									
Batalla de Boyacá							7			
Corpus Christi					15					
Día de la Ascensión				25						
Día de la Independencia						20				
Día de la Inmaculada Concepción										8
Día de la Raza								12		
Día de los Reyes Magos	12									
Día de Navidad										25
Día de San José		23								
Día del Trabajo				1						
Domingo de Ramos			5							
Domingo de Resurrección			12							

Independencia de Cartagena									16	
Jueves Santo			9							
La asunción de la Virgen							17			
Sagrado Corazón					22					
San Pedro y San Pablo					29					
Todos los Santos									2	
Viernes Santo			10							
2010										
Año Nuevo	1									
Batalla de Boyacá							7			
Corpus Christi					7					
Día de la Ascensión				17						
Día de la Independencia						20				
Día de la Inmaculada Concepción										8
Día de la Raza								18		
Día de los Reyes Magos	11									
Día de Navidad										25
Día de San José		22								
Día del Trabajo				1						
Domingo de Ramos		28								

Domingo de Resurrección			4							
Independencia de Cartagena									15	
Jueves Santo			1							
La asunción de la Virgen							16			
Sagrado Corazón					14					
San Pedro y San Pablo						5				
Todos los Santos									1	
Viernes Santo			2							
2011										
Año Nuevo	1									
Batalla de Boyacá							7			
Corpus Christi					27					
Día de la Ascensión					6					
Día de la Independencia						20				
Día de la Inmaculada Concepción										8
Día de la Raza								17		
Día de los Reyes Magos	10									
Día de Navidad										25
Día de San José		21								
Día del Trabajo				1						

Domingo de Ramos			17							
Domingo de Resurrección			24							
Independencia de Cartagena									14	
Jueves Santo			21							
La asunción de la Virgen							15			
Sagrado Corazón						4				
San Pedro y San Pablo						4				
Todos los Santos									7	
Viernes Santo			22							
2012										
Año Nuevo	1									
Batalla de Boyacá							7			
Corpus Christi					11					
Día de la Ascensión				21						
Día de la Independencia						20				
Día de la Inmaculada Concepción										8
Día de la Raza								15		
Día de los Reyes Magos	9									
Día de Navidad										25
Día de San José		19								

Día del Trabajo				1						
Domingo de Ramos			1							
Domingo de Resurrección			8							
Independencia de Cartagena									12	
Jueves Santo			5							
La asunción de la Virgen							20			
Sagrado Corazón					18					
San Pedro y San Pablo						2				
Todos los Santos									5	
Viernes Santo			6							
2013										
Año Nuevo	1									
Batalla de Boyacá							7			
Corpus Christi					3					
Día de la Ascensión				13						
Día de la Independencia						20				
Día de la Inmaculada Concepción										8
Día de la Raza								14		
Día de los Reyes Magos	7									
Día de Navidad										25

Día de San José		25								
Día del Trabajo				1						
Domingo de Ramos		24								
Domingo de Resurrección		31								
Independencia de Cartagena									11	
Jueves Santo		28								
La asunción de la Virgen							19			
Sagrado Corazón					10					
San Pedro y San Pablo						1				
Todos los Santos									4	
Viernes Santo		29								
2014										
Año Nuevo	1									
Batalla de Boyacá							7			
Corpus Christi					23					
Día de la Ascensión					2					
Día de la Independencia						20				
Día de la Inmaculada Concepción										8
Día de la Raza								13		
Día de los Reyes Magos	6									

Día de Navidad										25
Día de San José		24								
Día del Trabajo				1						
Domingo de Ramos			13							
Domingo de Resurrección			20							
Independencia de Cartagena									17	
Jueves Santo			17							
La asunción de la Virgen							18			
Sagrado Corazón					30					
San Pedro y San Pablo					30					
Todos los Santos									3	
Viernes Santo			18							

CÓDIGO FUENTE DESARROLLADO

#++#

```
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#FP++
#Autor      : Ing. Fabián Pallares Cabrera
#Fecha      : Cartagena ABRIL 06 2014
#Title      : Machine Learning (Predicción Series de Tiempo).
#Objetive   : Experimentar varias técnicas con esquemas diferentes de construcción
Del dataset (series de tiempo, series de tiempo con variables adicionales (regresión con retardos) y regresión con
información de reservas observadas   con días de anticipación.
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
import numpy as np
from StringIO import StringIO
from matplotlib import pyplot as plt
from sklearn import cross_validation

import mlpy as MLobjecto
from pylab import *
from numpy import *
import cPickle

#Librerias requeridas para manejo de redes RBF!
from scipy import *
from scipy.linalg import norm, pinv
from scipy import optimize
```

```
#Libreria ALGLIB
import xalglib

#Especificaciones del archivo
rutalog='C:\datasets\Tesis'



#Red neuronal implementada con ALGLIB++
class PML:

    def __init__(self,inputs,hidden,output):
        #Creación de PML Feedfoward con bias (Para problemas de regresión)!
        self._inputs=inputs
        self._hidden=hidden
        self._output=output
        self.network=xalglib.mlpcreate1(inputs,hidden,output)

    def learn(self,xinputs,youtput,weightdecay):
        xy=np.ones((xinputs.shape[0],xinputs.shape[1]+1))
        xy[:,0:xinputs.shape[1]]=xinputs
        xy[:,xinputs.shape[1]]=youtput
        xyz=xy.tolist()
        rep = xalglib.mlptrainlbfgs(self.network,xyz,len(xy),weightdecay,5,0.01,1000)

    def pred(self,xinput):
        xtest=xinput[0].tolist()
        ytest=[0]
```

```
        y = xalglib.mlpprocess(self.network, xtest, ytest)
        return y[0]

    def pred1(self,xinput):
        yresul=np.zeros(xinput.shape[0])
        ytest=[xinput.shape[0]]
        for i in range(xinput.shape[0]):
            xtest=xinput[i].tolist()
            y = xalglib.mlpprocess(self.network, xtest, ytest)
            yresul[i]=y[0]
        return yresul

#Manejo de redes neuronales función de base radial RBF!
#http://www.rueckstiess.net/research/snippets/show/72d2363e
class RBF:

    def __init__(self, indim, numCenters, outdim,_beta):
        self.indim = indim
        self.outdim = outdim
        self.numCenters = numCenters
        self.centers = [random.uniform(0, 1, indim) for i in xrange(numCenters)]
        self.beta =_beta
        self.W = random.random((self.numCenters, self.outdim))

    def _basisfunc(self, c, d):
        assert len(d) == self.indim
        return exp(-self.beta * norm(c-d)**2)
```

```
def _calcAct(self, X):
    # calculate activations of RBFs
    G = zeros((X.shape[0], self.numCenters), float)
    for ci, c in enumerate(self.centers):
        for xi, x in enumerate(X):
            G[xi,ci] = self._basisfunc(c, x)
    return G


def learn(self, X, Y,FactorRegularizacion):
    """ X: matrix of dimensions n x indim
        y: column vector of dimension n x 1 """

    # choose random center vectors from training set
    rnd_idx = random.permutation(X.shape[0])[:self.numCenters]
    self.centers = [X[i,:] for i in rnd_idx]

    # calculate activations of RBFs
    G = self._calcAct(X)

    #Calculando W incluyendo regularización (FP++)
    Gc=dot(transpose(G),G)
    n_col=Gc.shape[0]
    self.W= dot(dot(inv(Gc+FactorRegularizacion*eye(n_col)),transpose(G)),Y)

    #self.W = dot(pinv(G), Y)
```

```
        return 0



    def pred(self, X):
        """ X: matrix of dimensions n x indim """
        G = self._calcAct(X)
        Y = dot(G, self.W)
        return Y


    def pred1(self, X):
        return pred(X)


#Clase parámetros para realizar el experimento.
class clsparametrosexperimentales:
    file=""                    #Nombre del archivo y ruta de los datos!
    columndata=0               #Columna que provee la serie de tiempo!
    items=[]                   #Colección de ítems con parámetros experimentales!


    #clsitemevaluate: especifica los parámetros de evaluación del algoritmo!
    class clsitemevaluate:
        algoritmo=""                        #(RR)Ridge Regresion,(KRR)Kernel Ridge Regresion,(PML)Perceptron
Multicapa ,(RNFBR)Red Neuronal Funcion Radial
        Horizonteprediccion=15              #Nro. de registros para predecir en periodos seguidos en el tiempo.
        Lastperiodoslearning=0              #Cuantos periodos se utilizan para el entrenamiento (Basado en el origen de
rodadura).
```

```
                    #0:desde la primerfa observación x: representa el numero de periodos que se utilizaran
desde "recordsvalidation" hacia atrás!
                    #de "recordsvalidation" hacia adelante deben existir "recordstest"
    RecordOriginLearning=0          #Registro de inicio para la predicción!
    polynomial_grade=1              #Grado del polinomio regresión ridge regresion
    lags=2                      #Observaciones pasadas
    Lamdaridge=0.0                  #Coeficiente Ridge
    Mape_validation=0               #Promedio del porcentaje de error de validacion cruzada
    Mape_test=0                 #Error absoluto medio de test
    validationtype=""               #(RFO)"rolling forecasting origin", (CV) "Cross Validation K-Fold"
    TesterrorType=0                 #Indica si porcada predicción se recalibra el algoritmo (Computacionalmente
mas extenso)!
    recordsvalidation=12            #registros de predicción validación!
    recordstest=12                  #registros de predicción contemplado en el proceso de test!
    kfolds=1                    #Numero de Fold para cuando el tipo de validación es Cross Validation.
    Sigma=1                     #Sigma Modelo Kernel Ridge Regresión (Kernell Gausiano)!.
    Modelo=0                    #Modelo algoritmo!
    typekernel="kernel_polynomial"      #kernel_linear, kernel_polynomial,kernel_gaussian
    beta=1                      #Beta kernel polinomial!
    kernelridge_grade=1             #Grado de la función de kernel polinomica!
    nodoshidden=1                   #nodos capa oculta!
    nodosinput=1                #nodos capa de entrada!
    Kernel_gama=1                   #Gama, en Kernel polinomial!
    RBF_beta=5                  #gama utilizado, función de base radial!
    WeightDecay=0.001               #Coeficiente de regularización!
    Kx_learning=[]                  #Matriz utilizada para las funciones de Kernel para transformación.
    maxrecord_learning_and_validation=0
```

```
        maxrecord_learning=0
        maxrecord=0
        ModeloAnalisis="SerieTiempo"      #SerieTiempo: para cuando el análisis es por serie de tiempo, <> establece el análisis según pickup.

    #Constructor: ente método si inicializan los parámetros para entrenamiento y validación!
    def __init__(self):
        item=self.clsitemevaluate()

    #Método para generar los parámetros experimentales!
    def buildparameters(self,_Lastperiodoslearning,_lagini,_laggs,_gradesI,_gradesF,_listlmda,_Algoritmo,_validationtype,_typekernel,_beta,_kernelridge_grade,_sigma,_recordsvalidation,_TesterrorType,_recordstest,horizonte,_modeloanalisis):
        #self.items=[]
        for Lastperiodoslearning in _Lastperiodoslearning:
            for grade in range(_gradesI,_gradesF+1):
                for laag in range(_lagini,_laggs+1):
                    i=0
                    for lamda in _listlmda:
                        for kgrade in _kernelridge_grade:
                            for sig in _sigma:
                                item=self.clsitemevaluate()
                                item.algoritmo=_Algoritmo
                                item.polynomial_grade=grade
                                item.lags=laag
                                item.Lamdaridge=lamda
```

```
                    item.validationtype=_validationtype
                    item.recordsvalidation=_recordsvalidation
                    item.Sigma=sig
                    item.typekernel=_typekernel
                    item.beta=_beta
                    item.kernelridge_grade=kgrade
                    item.TesterrorType=_TesterrorType
                    item.recordstest=_recordstest
                    item.Horizonteprediccion=horizonte
                    item.Lastperiodoslearning=Lastperiodoslearning
                    item.RecordOriginLearning=0
                    item.maxrecord_learning_and_validation=0
                    item.maxrecord_learning=0
                    item.maxrecord=0
                    item.ModeloAnalisis=_modeloanalisis
                    self.items.append(item)

    def
buildparametersAANs(self,_Lastperiodoslearning,_inputsinicial,_inputsfinal,_hideninicial,_hidenfinal,_validationtyp
e,_recordsvalidation,_recordstest,_TesterrorType,WeightDecay,horizonte,_modeloanalisis):
        #self.items=[]
        for Lastperiodoslearning in _Lastperiodoslearning:
            for inp in range(_inputsinicial,_inputsfinal+1):
                for hid in range(_hideninicial,_hidenfinal+1):
                    item=self.clsitemevaluate()
                    item.algoritmo="PML"
                    item.polynomial_grade=1
```

```
                item.lags=inp
                item.Lamdaridge=1
                item.validationtype=_validationtype
                item.recordsvalidation=_recordsvalidation
                item.Sigma=1
                item.typekernel=""
                item.beta=0
                item.kernelridge_grade=1
                item.nodoshidden=hid
                item.nodosinput=inp
                item.recordstest=_recordstest
                item.TesterrorType=_TesterrorType
                item.WeightDecay=WeightDecay
                item.Horizonteprediccion=horizonte
                item.Lastperiodoslearning=Lastperiodoslearning
                item.RecordOriginLearning=0
                item.maxrecord_learning_and_validation=0
                item.maxrecord_learning=0
                item.maxrecord=0
                item.ModeloAnalisis=_modeloanalisis
                self.items.append(item)

    def
buildparametersRBF(self,_Lastperiodoslearning,_inputsinicial,_inputsfinal,_hideninicial,_hidenfinal,_validationtype,
beta,WeightDecay,_recordsvalidation,_recordstest,_TesterrorType,horizonte,_modeloanalisis):
        #self.items=[]
        for Lastperiodoslearning in _Lastperiodoslearning:
```

```
for inp in range(_inputsinicial,_inputsfinal+1):
    for hid in range(_hideninicial,_hidenfinal+1):
        item=self.clsitemevaluate()
        item.algoritmo="RBF"
        item.polynomial_grade=1
        item.lags=inp
        item.Lamdaridge=1
        item.validationtype=_validationtype
        item.recordsvalidation=_recordsvalidation
        item.Sigma=1
        item.typekernel=""
        item.beta=0
        item.kernelridge_grade=1
        item.nodoshidden=hid
        item.nodosinput=inp
        item.recordstest=_recordstest
        item.TesterrorType=_TesterrorType
        item.RBF_beta=beta
        item.WeightDecay=WeightDecay
        item.Horizonteprediccion=horizonte
        item.Lastperiodoslearning=Lastperiodoslearning
        item.RecordOriginLearning=0
        item.maxrecord_learning_and_validation=0
        item.maxrecord_learning=0
        item.maxrecord=0
        item.ModeloAnalisis=_modeloanalisis
        self.items.append(item)
```

```
#Clsdataset: permite obtener un data set, estableciendo información del archivo, Lags y orden del polinomio.
class clsdataset:
    columndata=0                #Columnas que contiene la serie
    rows=0                      #Números de registros a leer de la serie
    lags=0                      #Números de observaciones a tener en cuenta
    polynomial_grade=1          #Grado del polinomio
    file=""                     #Archivo de datos
    Serie=[]                    #Serie Original!

    #Inicializando DataSet con las entradas y las salidas del modelo!
    DataSet_input=[]
    DataSet_output=[]

    #Preparar el data set, cargando el @file y dependiendo de los rezagos y el grade del polinomio!
    def LoadDataSerie(self,_file,bias,normalizar,kpasosalante):
        self.makedummis=0
        self.file=_file
        DataSerie=np.loadtxt(_file,skiprows=1,delimiter=',')

        if normalizar==1:
            nmax=np.max(DataSerie[:,self.columndata])
            for i in range(DataSerie[:,self.columndata].shape[0]):
                DataSerie[i,self.columndata]=DataSerie[i,self.columndata]/nmax

        dataanalisis=DataSerie[:,self.columndata]
```

```
        if self.rows==0:
            dserie=DataSerie[:,self.columndata]
            self.rows=dserie.size-self.lags-kpasosalante


        self.nobias=bias

        #Inicializar el data set!
        self._columnas=self.lags*self.polynomial_grade+self.nobias
        self.DataSet_input=np.ones((self.rows,self._columnas))
        self.DataSet_output=np.ones((self.rows))

        #Preparar data sets, para cuando se analiza los datos con variables dummis(Año y dias).
        self.DataoutputAll= DataSerie[self.lags+kpasosalante:self.rows+self.lags+kpasosalante,:]
        #Preparar data sets, para cuando se analiza los datos con rezagos (Lags)
        self.DataSet_output= DataSerie[self.lags+kpasosalante:self.rows+self.lags+kpasosalante,self.columndata]
        for fila in range(0,self.rows):
            for columna in range(0,self.lags):
                for grado in range(0,self.polynomial_grade):

self.DataSet_input[fila,columna+self.lags*(grado)]=DataSerie[fila+columna,self.columndata]**(grado+1)


    def LoadDataregresionserie(self,_file,bias,kpasosalante=0):
        self.makedummis=1
        self.file=_file
```

```
        DataSerie=np.loadtxt(_file,skiprows=1,delimiter=',')

        dataanalisis=DataSerie[:,self.columndata]

        if self.rows==0:
            dserie=DataSerie[:,self.columndata]
            self.rows=dserie.size-self.lags-kpasosalante

        self.nobias=bias

        #Inicializar el data set!
        self._columnas=self.lags*self.polynomial_grade+self.nobias+12
        self.DataSet_input=np.ones((self.rows,self._columnas))
        self.DataSet_output=np.ones((self.rows))

        #Preparar data sets, para cuando se analiza los datos con variables dummis (Año y dias).
        self.DataoutputAll= DataSerie[self.lags+kpasosalante:self.rows+self.lags+kpasosalante,:]

        #Preparar data sets, para cuando se analiza los datos con rezagos (Lags)
        self.DataSet_output= DataSerie[self.lags+kpasosalante:self.rows+self.lags+kpasosalante,self.columndata]

        for fila in range(0,self.rows):
            for columna in range(0,self.lags):
                for grado in range(0,self.polynomial_grade):

self.DataSet_input[fila,columna+self.lags*(grado)]=DataSerie[fila+columna,self.columndata]**(grado+1)
            for col in range(5,15):
```

```
            self.DataSet_input[fila,self.lags*self.polynomial_grade+col-3]=self.DataoutputAll[fila,col]

    def LoadDataSerie_pickup(self,_file,analisis,bias):
        self.makedummis=0
        self.file=_file
        DataSerie=np.loadtxt(_file,skiprows=1,delimiter=',')
        self.rows=DataSerie.shape[0]

        if analisis=="pickup_add_90..60":

self.DataSet_input=DataSerie[:,[58,59,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89]]

#self.DataSet_input=DataSerie[:,[58,59,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..30":

self.DataSet_input=DataSerie[:,[57,58,59,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89]]

#self.DataSet_input=DataSerie[:,[57,58,59,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..20":
```

```
self.DataSet_input=DataSerie[:,[56,57,58,59,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89]
]

#self.DataSet_input=DataSerie[:,[56,57,58,59,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,8
5,86,87,88,89]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..15":

self.DataSet_input=DataSerie[:,[55,56,57,58,59,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,
89]]

#self.DataSet_input=DataSerie[:,[55,56,57,58,59,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,8
3,84,85,86,87,88,89]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..10":

self.DataSet_input=DataSerie[:,[54,55,56,57,58,59,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,
88,89]]

#self.DataSet_input=DataSerie[:,[54,55,56,57,58,59,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,8
1,82,83,84,85,86,87,88,89]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..7":
```

```
            #self.DataSet_input=DataSerie[:,[53,54,55,56,57,58,59,67,71,72,73,74,75,76]]

self.DataSet_input=DataSerie[:,[53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,
80,81,82,83,84,85,86,87,88,89]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..60_sin_meses":
            self.DataSet_input=DataSerie[:,[58,59,67,68,69,70,71,72,73,74,75,76,77]]
            #self.DataSet_input=DataSerie[:,[58,59,65,66,67,68,69,70,71,72,73,74,75,76,77]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..30_sin_meses":
            self.DataSet_input=DataSerie[:,[57,58,59,67,68,69,70,71,72,73,74,75,76,77]]
            #self.DataSet_input=DataSerie[:,[57,58,59,64,65,66,67,68,69,70,71,72,73,74,75,76,77]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..20_sin_meses":
            self.DataSet_input=DataSerie[:,[56,57,58,59,67,68,69,70,71,72,73,74,75,76,77]]
            #self.DataSet_input=DataSerie[:,[56,57,58,59,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..15_sin_meses":
            self.DataSet_input=DataSerie[:,[55,56,57,58,59,67,68,69,70,71,72,73,74,75,76,77]]
            #self.DataSet_input=DataSerie[:,[55,56,57,58,59,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..10_sin_meses":
```

```
            self.DataSet_input=DataSerie[:,[54,55,56,57,58,59,67,68,69,70,71,72,73,74,75,76,77]]
            #self.DataSet_input=DataSerie[:,[54,55,56,57,58,59,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..7_sin_meses":
            self.DataSet_input=DataSerie[:,[53,54,55,56,57,58,59,67,68,69,70,71,72,73,74,75,76,77]]

#self.DataSet_input=DataSerie[:,[53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77]]
            self.DataSet_output=DataSerie[:,90]



        _col=self.DataSet_input.shape[1]
        _fil=self.DataSet_input.shape[0]
        _colsize=_col*self.polynomial_grade+bias
        datax=np.ones((_fil,_colsize))

        for fila in range(0,_fil):
            for grado in range(0,self.polynomial_grade):
                for columna in range(0,_col):
                    datax[fila,columna+_col*(grado)]=DataSerie[fila,columna]**(grado+1)



        self.DataoutputAll=DataSerie
        self._columnas=self.DataSet_input.shape[1]

    def MakeDataSet(self):
        DataSerie=self.Serie
```

```
        if self.rows==0:
            dserie=DataSerie[:,self.columndata]
            self.rows=dserie.size-self.lags

        #inicializar el data set!
        self.DataSet_input=np.ones((self.rows,self.lags*self.polynomial_grade+1))
        self.DataSet_output=np.ones((self.rows))

        #Preparar data sets, para cuando se analiza los datos con rezagos (Lags)
        self.DataSet_output=DataSerie[self.lags:self.rows+self.lags,self.columndata]
        for fila in range(0,self.rows):
            for columna in range(0,self.lags):
                for grado in range(0,self.polynomial_grade):

self.DataSet_input[fila,columna+self.lags*(grado)]=DataSerie[fila+columna,self.columndata]**(grado+1)
    def GetSerie(self):
        self.Serie=np.loadtxt(self.file,skiprows=1,delimiter=',')
        if self.rows==0:
            dserie=self.Serie[:,self.columndata]
            self.rows=dserie.size-self.lags

#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#Función Entrenar para encajar el Modelo
#Esta función predice varios periodos adelantes (h_ahead).
#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
def Entrenar(op,X_train,Y_train):
```

```
#Establecer algoritmo para entrenamiento!
if op.algoritmo=="RR":
    Modelo = MLobjecto.Ridge(op.Lamdaridge)
if op.algoritmo=="KRR":
    Modelo = MLobjecto.KernelRidge(op.Lamdaridge)

if op.algoritmo=="RR":
    Modelo.learn(X_train, Y_train)

#Fit Kernell Ridge Regresión!
if op.algoritmo=="KRR":
    op.Kx_learning=X_train
    if op.typekernel=="kernel_linear":
        KX_train=X_train
        K = MLobjecto.kernel_linear(KX_train, KX_train)
        Modelo.learn(K, Y_train)
    if op.typekernel=="kernel_polynomial":
        KX_train=X_train
        K = MLobjecto.kernel_polynomial(KX_train, KX_train,1, op.beta,op.kernelridge_grade)
        Modelo.learn(K, Y_train)
    if op.typekernel=="kernel_gaussian":
        KX_train=X_train
        K = MLobjecto.kernel_gaussian(KX_train, KX_train, op.Sigma)
        Modelo.learn(K, Y_train)

#Fit Perceptron Alglib!
if op.algoritmo=='PML':
```

```
        Modelo = PML(op.nodosinput,op.nodoshidden,1)
        Modelo.learn(X_train, Y_train,op.WeightDecay)

    if op.algoritmo=="RBF":
        Modelo=RBF(op.nodosinput,op.nodoshidden,1,op.RBF_beta)
        Modelo.learn(X_train, Y_train,op.WeightDecay)

    return Modelo

#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#Función Forecasting para predecir varios periodos futuros dependiendo del parámetro en análisis!
#Esta función predice varios periodos adelantes (h_ahead).
#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
def forcasting(op,X_validation,Y_validation,h_ahead,Y_validation_ahead,KX_train,y_validation_all_data):
    #Defininiendo estructuras para error de predicción y el valor de la predicción!
    error_mae=np.zeros((h_ahead))
    prediccion=np.zeros((h_ahead))

    for it in range(h_ahead):
        Y_validation=Y_validation_ahead[it]

        #Predecir basado en el modelo encajado!
        if op.algoritmo=="KRR":
            if op.typekernel=="kernel_linear":
                Kt = MLobjecto.kernel_linear(X_validation, KX_train)
            if op.typekernel=="kernel_polynomial":
                Kt = MLobjecto.kernel_polynomial(X_validation, KX_train,1, op.beta,op.kernelridge_grade)
```

```
    if op.typekernel=="kernel_gaussian":
        Kt = MLobjecto.kernel_gaussian(X_validation, KX_train, op.Sigma)
    DataSet_Predictor_Z= Modelo.pred(Kt)
else:
    DataSet_Predictor_Z= Modelo.pred(X_validation)

prediccion[it]=DataSet_Predictor_Z
#error_mae[it]=np.sqrt(((100*(Y_validation-DataSet_Predictor_Z)/Y_validation))**2)
error_mae[it]=abs(100*(Y_validation-DataSet_Predictor_Z)/Y_validation)

if op.ModeloAnalisis=="Serietiempo" or op.ModeloAnalisis=="regresion_retardos":
    if h_ahead>1:
        #Por cada iteración se debe modificar x_validation con el nuevo pronóstico y cambiar el y_validation
        for x_laag in range(0,op.lags-1):
            X_validation[0,x_laag]=X_validation[0,x_laag+1]
        X_validation[0,op.lags-1]=DataSet_Predictor_Z

        #Recalcular los inputs siguiente!
        for columna in range(0,op.lags):
            for grado in range(0,op.polynomial_grade):
                X_validation[0,columna+op.lags*(grado)]=X_validation[0,columna]**(grado+1)
            if odataset.makedummis==1:
                for col in range(5,15):
                    X_validation[0,op.lags*op.polynomial_grade+col-3]=y_validation_all_data[it,col]


#al final computo el promedio de los errores mape.
```

```
    return  np.average(error_mae), prediccion

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
#++++++++++++++++++++++++++++++++++ INICIO
++++++++++++++++++++++++++++++++++++++++++++++++++++++
#Inicializando objetos de parámetros para establecer el experimento y objeto pára armnar dataset !!
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++


#Iniciando parámetros para realizar los experimentos!
oparameters = clsparametrosexperimentales()


#Establezca el nivel de análisis y presentación de la data.
_showplotforcast=0          #1: Mostrar el gráfico  con el análisis 0:No mostrar gráfico


#-----------------------------------------------------------------------------------------------------#
#EN ESTA SECCIÓN SE ESTABLECEN TODOS LOS PARÁMETROS DE LOS EXPERIMENTOS
#-----------------------------------------------------------------------------------------------------#


_Listamodelos=["Serietiempo"]
for _modelanalisis in _Listamodelos:


oparameters.buildparameters([0],21,24,1,2,[0.1,0.15,0.25,0.35],"RR","RFO","",1,[1],[1],402,0,181,1,_modelanalisis)


oparameters.buildparameters([0],2,24,1,1,[0.1,0.15,0.25,0.35],"KRR","RFO","kernel_gaussian",0.1,[1],[3,4.5,5],402,
0,181,1,_modelanalisis)
```

oparameters.buildparameters([0],2,24,1,1,[0.015,0.15,0.25,0.35],"KRR","RFO","kernel_gaussian",0.1,[1],[1000,2000,2300,3000],402,0,181,1,_modelanalisis)

oparameters.buildparameters([0],20,21,1,1,[0.3],"KRR","RFO","kernel_polynomial",0.1,[1,2],[1],402,0,181,1,_modelanalisis)

oparameters.buildparametersAANs([0],2,24,50,50,"RFO",402,181,0,0.1,1,_modelanalisis)

oparameters.buildparametersAANs([0],2,24,50,50,"RFO",402,181,0,0.25,1,_modelanalisis)

oparameters.buildparametersRBF([0],6,24,150,150,"RFO",0.00000015,0.0015,402,181,0,1,_modelanalisis)

_Listamodelosregresionretardos=["regresion_retardos"]

for _modelanalisis in _Listamodelosregresionretardos:

oparameters.buildparameters([0],21,24,1,2,[0.1,0.15,0.25,0.35],"RR","RFO","",1,[1],[1],402,0,181,1,_modelanalisis)

oparameters.buildparameters([0],2,24,1,1,[0.1,0.15,0.25,0.35],"KRR","RFO","kernel_gaussian",0.1,[1],[3,4.5,5],402,0,181,1,_modelanalisis)

oparameters.buildparameters([0],2,24,1,1,[0.015,0.15,0.25,0.35],"KRR","RFO","kernel_gaussian",0.1,[1],[1000,2000,2300,3000],402,0,181,1,_modelanalisis)

oparameters.buildparameters([0],20,21,1,1,[0.3],"KRR","RFO","kernel_polynomial",0.1,[1,2],[1],402,0,181,1,_modelanalisis)

oparameters.buildparametersAANs([0],2,24,50,50,"RFO",402,181,0,0.1,1,_modelanalisis)

oparameters.buildparametersAANs([0],2,24,50,50,"RFO",402,181,0,0.25,1,_modelanalisis)

oparameters.buildparametersRBF([0],6,24,150,150,"RFO",0.00000015,0.0015,402,181,0,1,_modelanalisis)

```
_Listademodelospickup=["pickup_add_90..7"]
for _modelanalisis in _Listademodelospickup:
    oparameters.buildparameters([0],1,1,1,2,[0.1],"RR","RFO","",1,[1],[1],402,0,181,1,_modelanalisis)

oparameters.buildparameters([0],1,1,1,1,[0.1,0.15,0.25,0.35],"KRR","RFO","kernel_gaussian",1,[1],[1,10,100,1000],
402,0,181,1,_modelanalisis)

oparameters.buildparameters([0],1,1,1,1,[0.1,0.15,0.25,0.35],"KRR","RFO","kernel_polynomial",0.1,[1,2],[1],402,0,
181,1,_modelanalisis)
    oparameters.buildparametersAANs([0],1,1,50,50,"RFO",402,181,0,0.1,1,_modelanalisis)
    oparameters.buildparametersAANs([0],1,1,70,70,"RFO",402,181,0,0.5,1,_modelanalisis)
    oparameters.buildparametersAANs([0],1,1,100,100,"RFO",402,181,0,0.7,1,_modelanalisis)
    oparameters.buildparametersRBF([0],1,1,50,50,"RFO",0.0000000001,0.0015,402,181,0,1,_modelanalisis)
    oparameters.buildparametersRBF([0],1,1,70,70,"RFO",0.0000000015,0.15,402,181,0,1,_modelanalisis)
    oparameters.buildparametersRBF([0],1,1,150,150,"RFO",0.0000000150,0.5,402,181,0,1,_modelanalisis)

#Instanciar objeto para administrar DATASET!
odataset=clsdataset()

#Abrir archivo en disco para grabar resultado del experimento!
rutalog=rutalog+'\MachineLearningSumary.txt'
File=open(rutalog,'w')
File.write("Algoritmo" + "\t" + "type" +"\t" + "Laags" +"\t" +"Polynomial_grade" +"\t"+ "Sigma"
+"\t"+"Lamdaridge"+"\t" +"Mape_test"+"\t" +"Mape_validation"+"\t"+"Mape_test"+"\t"
+"Mape_validation"+"\t"+"Horizinte learning validación"+"\t"+"Horizinte learning test"+"\t"+"nodos
```

```
input"+"\t"+"nodos hidden"+"\t"+"Kernel
Grade"+"\t"+"KRR_beta"+"\t"+"RBF_beta"+"\t"+"weighdecay"+"\t"+"modeloanalizado"+"\n")

#recorrer el objeto de parametros para evaluar modelos y encontrar el mejor modelo !
besterror=-1
bestmodelo=0
for op in oparameters.items:

    #Inicializando los parametros para construir el dataset!
    odataset.columndata=oparameters.columndata
    odataset.rows=np.int(0)
    odataset.file=oparameters.file
    odataset.lags=op.lags
    odataset.polynomial_grade=op.polynomial_grade #para caso ridge regresión

    #Cargando la serie y modelando el dataset basado en los parámetros establecidos de rezagos y grado del
polinomio.
    if op.ModeloAnalisis=="Serietiempo":
        odataset.columndata=27
        odataset.LoadDataSerie("C:\datasets\Tesis\SerieTime_Ocupacion.txt",1,0,0)
    else :
        if op.ModeloAnalisis=="regresion_retardos":
            odataset.columndata=28
            odataset.LoadDataregresionserie("C:\datasets\Tesis\Estadistica_ocupacion_tesis_dumis.txt",1,6)
        else:
            #odataset.LoadDataSerie_pickup("C:\datasets\Tesis\datos_pickup.txt",op.ModeloAnalisis)
            odataset.LoadDataSerie_pickup("C:\datasets\Tesis\datos_pickup_addcancel.txt",op.ModeloAnalisis,0)
```

```
#Actualizando número de columnas basado en los datos encajados!
op.nodosinput=odataset._columnas  #El número de columnas corresponde al no de nodos capa de entrada de la red.

#Calculando el máximo número de observaciones para entrenamiento y validación (separamos los registros para test).
op.maxrecord_learning_and_validation=odataset.rows-op.recordstest
op.maxrecord_learning=op.maxrecord_learning_and_validation - op.recordsvalidation
op.maxrecord=odataset.rows

#Establecer el registro de inicio del Análisis #Según las técnicas de Rolling Forecasting Origin!
if op.Lastperiodoslearning==0:
    op.RecordOriginLearning=0
else:
    op.RecordOriginLearning=op.maxrecord_learning-op.Lastperiodoslearning

if op.validationtype=="RFO":
    #!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    #Encajar el Modelo y Luego Predecir basado en el conjunto de validación, calculando el error
    #Armar datos de entrenamiento y validación!
    #!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    X_train=odataset.DataSet_input[op.RecordOriginLearning:op.maxrecord_learning,:]
    Y_train=odataset.DataSet_output[op.RecordOriginLearning:op.maxrecord_learning]
```

```
#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#Entrenar (encajar) el algoritmo basado en los datos de entrenamiento
#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Modelo = Entrenar(op,X_train,Y_train)

#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#Recorrer el registros para validar el modelo!
#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#Contador del registros de pronostico!
_cont=0
registros=op.recordsvalidation
scores =  np.zeros((registros))
h=op.Horizonteprediccion #establece el numero de predicciones futuras
h=0

for rec in range(0,registros-h):

   #Preparar conjunto de test!
   indexmaslearningandvalidaction=op.maxrecord_learning_and_validation-registros+rec
   indexorigen=0

   #En cada Iteración se obtiene el conjunto de validación
   X_validation =odataset.DataSet_input[indexmaslearningandvalidaction:indexmaslearningandvalidaction+1,:]
   Y_validation = odataset.DataSet_output[indexmaslearningandvalidaction:indexmaslearningandvalidaction+1]

   if h==0:
```

```
            hpre=1

        Y_validation_ahead=
odataset.DataSet_output[indexmaslearningandvalidaction:indexmaslearningandvalidaction+hpre]

y_validation_all_data=odataset.DataoutputAll[indexmaslearningandvalidaction:indexmaslearningandvalidaction+hpr
e]

        #Pronosticando en cada iteracción!
        predicion=np.zeros(hpre)

scores[_cont],prediccion=forcasting(op,X_validation,Y_validation,hpre,Y_validation_ahead,X_train,y_validation_al
l_data)
        _cont=_cont+1

    #Calculando el error de validación!
    op.Mape_validation = np.average(scores)

    #Escribir en el archivo de resumen de resultado las evaluaciones realizadas!
    CadenaResultado=op.algoritmo + "\t" + op.typekernel +"\t" + str(op.lags) +"\t" +str(op.polynomial_grade)+"\t"
+str(op.Sigma)+"\t" +str(op.Lamdaridge)+"\t" +str(op.Mape_test)+"\t" +str(op.Mape_validation)+"\t"
+str(op.Mape_test)+"\t" +str(op.Mape_validation)+"\t" +str(op.recordsvalidation)+"\t" +str(op.recordstest)+ "\t"
+str(op.nodosinput) + "\t" + str(op.nodoshidden)+ "\t" + str(op.kernelridge_grade)+ "\t" + str(op.beta)+ "\t" +
str(op.RBF_beta)+ "\t" + str(op.WeightDecay)+ "\t" + str(op.ModeloAnalisis)+"\n"
    File.write(CadenaResultado)

    #En cada iteración se selecciona el mejor modelo (Aquel con menor error de validación)!
```

```
        cadena="Algoritmo:"+op.algoritmo + " Inputs:"+str(op.nodosinput)+" Hidden:"+str(op.nodoshidden)+"
sigma:"+str(op.Sigma)+" beta:"+str(op.beta)+" Lags:"+str(op.lags)+" grado:"+str(op.polynomial_grade)+"
Lamda:"+str(op.Lamdaridge)+" WeightDecay:"+str(op.WeightDecay)+ " Error: "+str(op.Mape_validation)+
"RBF_Beta: "+str(op.RBF_beta) + "Modelo Analizado: "+str(op.ModeloAnalisis)
        op.Modelo=Modelo
        print(cadena)
        if op.Mape_validation<besterror or besterror<0:
            bestparameters=op
            bestmodelo=Modelo
            besterror=op.Mape_validation

#Cerrar el archivo de resumen de resultados!
File.close

#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#Testear el mejor modelo!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#Contador del registros de pronostico!

_cont=0
registros=bestparameters.recordstest
scores=np.zeros((registros))
h=bestparameters.Horizonteprediccion

X_train=odataset.DataSet_input[bestparameters.maxrecord_learning:bestparameters.maxrecord_learning_and_valida
tion,:]
```

```
Y_train=odataset.DataSet_output[bestparameters.maxrecord_learning:bestparameters.maxrecord_learning_and_valid
ation]
Kx_train=bestparameters.Kx_learning



if bestparameters.algoritmo=="KRR":
    if bestparameters.typekernel=="kernel_linear":
        Kt = MLobjecto.kernel_linear(X_train, Kx_train)
    if bestparameters.typekernel=="kernel_polynomial":
        Kt = MLobjecto.kernel_polynomial(X_train, Kx_train,1,bestparameters.beta,bestparameters.kernelridge_grade)


    if bestparameters.typekernel=="kernel_gaussian":
        Kt = MLobjecto.kernel_gaussian(X_train, Kx_train, bestparameters.Sigma)


    ypred= bestmodelo.pred(Kt)
else:
    if bestparameters.algoritmo=="PML":
        ypred=bestmodelo.pred1(X_train)
    else:
        ypred=bestmodelo.pred(X_train)



plt.clf()
plt.plot(ypred, linewidth = 2, label = 'Predicción', color='red')
plt.plot(Y_train, linewidth = 2, label = 'serie', color='black')
plt.title="Pronostico sobre serie de validación!"
plt.show()
```

```
#Recorrer los registros para calcular el error de pronostico!
if _showplotforcast==1:
    h=bestparameters.Horizonteprediccion
    for rec in range(0,registros-h):
        #Preparar conjunto de entrenamiento!
        indexmaslearningandvalidaction=bestparameters.maxrecord_learning_and_validation+rec
        indexorigen=0

        #En cada Iteracción se obtiene el conjunto de validación
        X_validation =odataset.DataSet_input[indexmaslearningandvalidaction:indexmaslearningandvalidaction+1,:]
        Y_validation = odataset.DataSet_output[indexmaslearningandvalidaction:indexmaslearningandvalidaction+1]
        Y_validation_ahead=
odataset.DataSet_output[indexmaslearningandvalidaction:indexmaslearningandvalidaction+h]

y_validation_all_data=odataset.DataoutputAll[indexmaslearningandvalidaction:indexmaslearningandvalidaction+h]

        #Pronosticando en cada iteracción!
        predicion=np.zeros(h)
        scores=np.zeros(h)

        predicion=np.zeros(h)

scores[_cont],prediccion=forcasting(bestparameters,X_validation,Y_validation,h,Y_validation_ahead,Kx_train,y_val
idation_all_data)
        _cont=_cont+1
```

```
#Graficar los resultados, gráfico  de validacion
plt.clf()
cadena="Forcast vs observation error:"
#plt.title(cadena)
plt.plot(prediccion, linewidth = 2, label = 'Predicción', color='red')
plt.plot(Y_validation_ahead, linewidth = 2, label = 'serie', color='black')
plt.ylabel="Serie"
plt.show()
```

```
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#Autor    : Ing. Fabián Pallares Cabrera
#Fecha    : Cartagena Abril 06 2014
#titulo   : Machine Learning (Predicción Series de Tiempo).
#Objetivos: Predicción de la ocupación utilizando mejor modelo sobre datos de test
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
import numpy as np
from StringIO import StringIO
from matplotlib import pyplot as plt
from sklearn import cross_validation

import mlpy as MLobjecto
from pylab import *
from numpy import *
import cPickle

#Librerías requeridas para manejo de redes RBF!
from scipy import *
from scipy.linalg import norm, pinv
from scipy import optimize
import xalglib

#Ruta del utilizada para grabar el archivo de resultados.
rutalog='C:\datasets\Tesis'



#Red neuronal implementada con ALGLIB++
```

```
class PML:

    def __init__(self,inputs,hidden,output):
        #creación de PML Feedfoward con bias (Para problemas de regresión)!
        self._inputs=inputs
        self._hidden=hidden
        self._output=output
        self.network=xalglib.mlpcreate1(inputs,hidden,output)

    def learn(self,xinputs,youtput,weightdecay):
        xy=np.ones((xinputs.shape[0],xinputs.shape[1]+1))
        xy[:,0:xinputs.shape[1]]=xinputs
        xy[:,xinputs.shape[1]]=youtput
        xyz=xy.tolist()
        rep = xalglib.mlptrainlbfgs(self.network,xyz,len(xy),weightdecay,5,0.01,1000)

    def pred(self,xinput):
        xtest=xinput[0].tolist()
        ytest=[0]
        y = xalglib.mlpprocess(self.network, xtest, ytest)
        return y[0]

    def pred1(self,xinput):
        yresul=np.zeros(xinput.shape[0])
        ytest=[xinput.shape[0]]
        for i in range(xinput.shape[0]):
            xtest=xinput[i].tolist()
```

```
            y = xalglib.mlpprocess(self.network, xtest, ytest)
            yresul[i]=y[0]
        return yresul

#Manejo de redes neuronales función de base radial RBF!
#http://www.rueckstiess.net/research/snippets/show/72d2363e
class RBF:

    def __init__(self, indim, numCenters, outdim,_beta):
        self.indim = indim
        self.outdim = outdim
        self.numCenters = numCenters
        self.centers = [random.uniform(0, 1, indim) for i in xrange(numCenters)]
        self.beta =_beta
        self.W = random.random((self.numCenters, self.outdim))

    def _basisfunc(self, c, d):
        assert len(d) == self.indim
        return exp(-self.beta * norm(c-d)**2)



    def _calcAct(self, X):
        # calculate activations of RBFs
        G = zeros((X.shape[0], self.numCenters), float)
        for ci, c in enumerate(self.centers):
            for xi, x in enumerate(X):
                G[xi,ci] = self._basisfunc(c, x)
```

```
    return G

def learn(self, X, Y,FactorRegularizacion):
    """ X: matrix of dimensions n x indim
        y: column vector of dimension n x 1 """

    # choose random center vectors from training set
    rnd_idx = random.permutation(X.shape[0])[:self.numCenters]
    self.centers = [X[i,:] for i in rnd_idx]

    # calculate activations of RBFs
    G = self._calcAct(X)

    #Calculando W incluyendo regularización (FP++)
    Gc=dot(transpose(G),G)
    n_col=Gc.shape[0]
    self.W= dot(dot(inv(Gc+FactorRegularizacion*eye(n_col)),transpose(G)),Y)

    #self.W = dot(pinv(G), Y)
    return 0

def pred(self, X):
    """ X: matrix of dimensions n x indim """
    G = self._calcAct(X)
    Y = dot(G, self.W)
    return Y
```

```
    def pred1(self, X):
        return pred(X)


#Clase parámetros para realizar el experimento.
class clsparametrosexperimentales:
    file=""                    #Nombre del archivo y ruta de los datos!
    columndata=0               #Columna que provee la serie de tiempo!
    items=[]                   #Colección de ítems con parámetros experimentales!


    #clsitemevaluate: especifica los parámetros de evaluación del algoritmo!
    class clsitemevaluate:
        algoritmo=""                    #(RR)Ridge Regression,(KRR)Kernel Ridge Regression,(PML)Perceptron
Multicapa ,(FBR)Red Neuronal Funcion Radial
        Horizonteprediccion=15          #Nro. de registros para predecir en periodos seguidos en el tiempo.
        Lastperiodoslearning=0          #Cuantos periodos se utilizan para el entrenamiento (Basado en el origen de
rodadura).
                                        #0: Desde la primera observación x: representa el numero de periodos que se utilizaran
desde "recordsvalidation" hacia atrás!
                                        #de "recordsvalidation" hacia adelante deben existir "recordstest"
        RecordOriginLearning=0          #Registro de inicio para la predicción!
        polynomial_grade=1              #Grado del polinomio regresión ridge regression
        lags=2                          #Observaciones pasadas
        Lamdaridge=0.0                  #Coeficiente Ridge
        Mape_validation=0               #Promedio del porcentaje de error de validación cruzada
        Mape_test=0                     #Error absoluto medio de test
        validationtype=""               #(RFO)"rolling forecasting update", (CV) "Cross Validation K-Fold"
```

```
        TesterrorType=0                 #Indica si porcada predicción se recalibra el algoritmo (Computacionalmente
mas extenso)!
        recordsvalidation=12            #registros de predicción validación!
        recordstest=12                  #registros de predicción contemplado en el proceso de test!
        kfolds=1                        #Numero de Fold para cuando el tipo de validación es Cross Validation.
        Sigma=1                         #Sigma Modelo Kernel Ridge Regresión (Kernel Gaussiano)!.
        Modelo=0                        #Modelo algoritmo!
        typekernel="kernel_polynomial"      #kernel_linear, kernel_polynomial,kernel_gaussian
        beta=1                          #Beta kernel polinomial!
        kernelridge_grade=1             #Grado de la función de kernel
polinomica!
        nodoshidden=1                   #nodos capa oculta!
        nodosinput=1                    #nodos capa de entrada!
        Kernel_gama=1                   #Gama, en Kernel polinomial!
        RBF_beta=5                      #gama utilizado, función de base radial!
        WeightDecay=0.001               #Coeficiente de regularización!
        Kx_learning=[]                  #Matriz utilizada para las funciones de Kernel para transformación.
        maxrecord_learning_and_validation=0
        maxrecord_learning=0
        maxrecord=0
        ModeloAnalisis="SerieTiempo"    #SerieTiempo: para cuando el análisis es por serie de tiempo, <> establece
el análisis según pickup.

    #Constructor: ente método si inicializan los parámetros para entrenamiento y validación!
    def __init__(self):
        item=self.clsitemevaluate()
```

```
    #Método para generar los parámetros experimentales!
    def
buildparameters(self,_Lastperiodoslearning,_lagini,_laggs,_gradesI,_gradesF,_listlmda,_Algoritmo,_validationtype,_
typekernel,_beta,_kernelridge_grade,_sigma,_recordsvalidation,_TesterrorType,_recordstest,horizonte,_modeloanali
sis):
        #self.items=[]
        for Lastperiodoslearning in _Lastperiodoslearning:
            for grade in range(_gradesI,_gradesF+1):
                for laag in range(_lagini,_laggs+1):
                    i=0
                    for lamda in _listlmda:
                        for kgrade in _kernelridge_grade:
                            for sig in _sigma:
                                item=self.clsitemevaluate()
                                item.algoritmo=_Algoritmo
                                item.polynomial_grade=grade
                                item.lags=laag
                                item.Lamdaridge=lamda
                                item.validationtype=_validationtype
                                item.recordsvalidation=_recordsvalidation
                                item.Sigma=sig
                                item.typekernel=_typekernel
                                item.beta=_beta
                                item.kernelridge_grade=kgrade
                                item.TesterrorType=_TesterrorType
                                item.recordstest=_recordstest
                                item.Horizonteprediccion=horizonte
```

```
                    item.Lastperiodoslearning=Lastperiodoslearning
                    item.RecordOriginLearning=0
                    item.maxrecord_learning_and_validation=0
                    item.maxrecord_learning=0
                    item.maxrecord=0
                    item.ModeloAnalisis=_modeloanalisis
                    self.items.append(item)

    def buildparametersAANs(self,_Lastperiodoslearning,_inputsinicial,_inputsfinal,_hideninicial,_hidenfinal,_validationtype,_recordsvalidation,_recordstest,_TesterrorType,WeightDecay,horizonte,_modeloanalisis):
        #self.items=[]
        for Lastperiodoslearning in _Lastperiodoslearning:
            for inp in range(_inputsinicial,_inputsfinal+1):
                for hid in range(_hideninicial,_hidenfinal+1):
                    item=self.clsitemevaluate()
                    item.algoritmo="PML"
                    item.polynomial_grade=1
                    item.lags=inp
                    item.Lamdaridge=1
                    item.validationtype=_validationtype
                    item.recordsvalidation=_recordsvalidation
                    item.Sigma=1
                    item.typekernel=""
                    item.beta=0
                    item.kernelridge_grade=1
                    item.nodoshidden=hid
```

```
                item.nodosinput=inp
                item.recordstest=_recordstest
                item.TesterrorType=_TesterrorType
                item.WeightDecay=WeightDecay
                item.Horizonteprediccion=horizonte
                item.Lastperiodoslearning=Lastperiodoslearning
                item.RecordOriginLearning=0
                item.maxrecord_learning_and_validation=0
                item.maxrecord_learning=0
                item.maxrecord=0
                item.ModeloAnalisis=_modeloanalisis
                self.items.append(item)

    def
buildparametersRBF(self,_Lastperiodoslearning,_inputsinicial,_inputsfinal,_hideninicial,_hidenfinal,_validationtype,
beta,WeightDecay,_recordsvalidation,_recordstest,_TesterrorType,horizonte,_modeloanalisis):
        #self.items=[]
        for Lastperiodoslearning in _Lastperiodoslearning:
            for inp in range(_inputsinicial,_inputsfinal+1):
                for hid in range(_hideninicial,_hidenfinal+1):
                    item=self.clsitemevaluate()
                    item.algoritmo="RBF"
                    item.polynomial_grade=1
                    item.lags=inp
                    item.Lamdaridge=1
                    item.validationtype=_validationtype
                    item.recordsvalidation=_recordsvalidation
```

```
            item.Sigma=1
            item.typekernel=""
            item.beta=0
            item.kernelridge_grade=1
            item.nodoshidden=hid
            item.nodosinput=inp
            item.recordstest=_recordstest
            item.TesterrorType=_TesterrorType
            item.RBF_beta=beta
            item.WeightDecay=WeightDecay
            item.Horizonteprediccion=horizonte
            item.Lastperiodoslearning=Lastperiodoslearning
            item.RecordOriginLearning=0
            item.maxrecord_learning_and_validation=0
            item.maxrecord_learning=0
            item.maxrecord=0
            item.ModeloAnalisis=_modeloanalisis
            self.items.append(item)

#Clsdataset: permite obtener un data set, estableciendo información del archivo, lags y orden del polinomio.
class clsdataset:
    columndata=0                #Columnas que contiene la serie
    rows=0                      #Números de registros a leer de la serie
    lags=0                      #Números de observaciones a tener en cuenta
    polynomial_grade=1          #Grado del polinomio
    file=""                     #Archivo de datos
    Serie=[]                    #Serie Original!
```

```
#Inicializando DataSet con las entradas y las salidas del modelo!
DataSet_input=[]
DataSet_output=[]

#Preparar el data set, cargando el @file y dependiendo de los rezagos y el grade del polinomio!
def LoadDataSerie(self,_file,bias,normalizar,kpasosalante):
    self.makedummis=0
    self.file=_file
    DataSerie=np.loadtxt(_file,skiprows=1,delimiter=',')

    if normalizar==1:
        nmax=np.max(DataSerie[:,self.columndata])
        for i in range(DataSerie[:,self.columndata].shape[0]):
            DataSerie[i,self.columndata]=DataSerie[i,self.columndata]/nmax



    dataanalisis=DataSerie[:,self.columndata]

    if self.rows==0:
        dserie=DataSerie[:,self.columndata]
        self.rows=dserie.size-self.lags-kpasosalante



    self.nobias=bias

    #Inicializar el data set!
```

```
        self._columnas=self.lags*self.polynomial_grade+self.nobias
        self.DataSet_input=np.ones((self.rows,self._columnas))
        self.DataSet_output=np.ones((self.rows))

        #Preparar data sets, para cuando se analiza los datos con variables dummis (Año y días).
        self.DataoutputAll= DataSerie[self.lags+kpasosalante:self.rows+self.lags+kpasosalante,:]
        #Preparar data sets, para cuando se analiza los datos con rezagos (Lags)
        self.DataSet_output= DataSerie[self.lags+kpasosalante:self.rows+self.lags+kpasosalante,self.columndata]
        for fila in range(0,self.rows):
            for columna in range(0,self.lags):
                for grado in range(0,self.polynomial_grade):

self.DataSet_input[fila,columna+self.lags*(grado)]=DataSerie[fila+columna,self.columndata]**(grado+1)


    def LoadDataregresionserie(self,_file,bias,kpasosalante=0):
        self.makedummis=1
        self.file=_file
        DataSerie=np.loadtxt(_file,skiprows=1,delimiter=',')

        dataanalisis=DataSerie[:,self.columndata]

        if self.rows==0:
            dserie=DataSerie[:,self.columndata]
            self.rows=dserie.size-self.lags-kpasosalante

        self.nobias=bias
```

```
        #Inicializar el data set!
        self._columnas=self.lags*self.polynomial_grade+self.nobias+12
        self.DataSet_input=np.ones((self.rows,self._columnas))
        self.DataSet_output=np.ones((self.rows))

        #Preparar data sets, para cuando se analiza los datos con variables dummis(Año y dias).
        self.DataoutputAll= DataSerie[self.lags+kpasosalante:self.rows+self.lags+kpasosalante,:]

        #Preparar data sets, para cuando se analiza los datos con rezagos (Laags)
        self.DataSet_output= DataSerie[self.lags+kpasosalante:self.rows+self.lags+kpasosalante,self.columndata]

        for fila in range(0,self.rows):
            for columna in range(0,self.lags):
                for grado in range(0,self.polynomial_grade):

self.DataSet_input[fila,columna+self.lags*(grado)]=DataSerie[fila+columna,self.columndata]**(grado+1)
            for col in range(5,15):
                self.DataSet_input[fila,self.lags*self.polynomial_grade+col-3]=self.DataoutputAll[fila,col]

    def LoadDataSerie_pickup(self,_file,analisis,bias):
        self.makedummis=0
        self.file=_file
        DataSerie=np.loadtxt(_file,skiprows=1,delimiter=',')
        self.rows=DataSerie.shape[0]

        if analisis=="pickup_add_90..60":
```

```
self.DataSet_input=DataSerie[:,[58,59,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..30":

self.DataSet_input=DataSerie[:,[57,58,59,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..20":

self.DataSet_input=DataSerie[:,[56,57,58,59,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89]
]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..15":
self.DataSet_input=DataSerie[:,[55,56,57,58,59,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,
89]]
self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..10":

self.DataSet_input=DataSerie[:,[54,55,56,57,58,59,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,
88,89]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..7":
```

```
self.DataSet_input=DataSerie[:,[53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,
80,81,82,83,84,85,86,87,88,89]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..60_sin_meses":
            self.DataSet_input=DataSerie[:,[58,59,67,68,69,70,71,72,73,74,75,76,77]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..30_sin_meses":
            self.DataSet_input=DataSerie[:,[57,58,59,67,68,69,70,71,72,73,74,75,76,77]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..20_sin_meses":
            self.DataSet_input=DataSerie[:,[56,57,58,59,67,68,69,70,71,72,73,74,75,76,77]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..15_sin_meses":
            self.DataSet_input=DataSerie[:,[55,56,57,58,59,67,68,69,70,71,72,73,74,75,76,77]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..10_sin_meses":
            self.DataSet_input=DataSerie[:,[54,55,56,57,58,59,67,68,69,70,71,72,73,74,75,76,77]]
            self.DataSet_output=DataSerie[:,90]

        if analisis=="pickup_add_90..7_sin_meses":
            self.DataSet_input=DataSerie[:,[53,54,55,56,57,58,59,67,68,69,70,71,72,73,74,75,76,77]]
            self.DataSet_output=DataSerie[:,90]
```

```
        _col=self.DataSet_input.shape[1]
        _fil=self.DataSet_input.shape[0]
        _colsize=_col*self.polynomial_grade+bias
        datax=np.ones((_fil,_colsize))

        for fila in range(0,_fil):
            for grado in range(0,self.polynomial_grade):
                for columna in range(0,_col):
                    datax[fila,columna+_col*(grado)]=DataSerie[fila,columna]**(grado+1)



        self.DataoutputAll=DataSerie
        self._columnas=self.DataSet_input.shape[1]

def MakeDataSet(self):
    DataSerie=self.Serie
    if self.rows==0:
        dserie=DataSerie[:,self.columndata]
        self.rows=dserie.size-self.lags



    self.DataSet_input=np.ones((self.rows,self.lags*self.polynomial_grade+1))
    self.DataSet_output=np.ones((self.rows))

    #Preparar data sets, para cuando se analiza los datos con rezagos (Lags)
```

```
        self.DataSet_output=DataSerie[self.lags:self.rows+self.lags,self.columndata]
        for fila in range(0,self.rows):
            for columna in range(0,self.lags):
                for grado in range(0,self.polynomial_grade):

self.DataSet_input[fila,columna+self.lags*(grado)]=DataSerie[fila+columna,self.columndata]**(grado+1)
    def GetSerie(self):
        self.Serie=np.loadtxt(self.file,skiprows=1,delimiter=',')
        if self.rows==0:
            dserie=self.Serie[:,self.columndata]
            self.rows=dserie.size-self.lags

def Entrenar(op,X_train,Y_train):

    #Establecer algoritmo para entrenamiento!
    if op.algoritmo=="RR":
        Modelo = MLobjecto.Ridge(op.Lamdaridge)
    if op.algoritmo=="KRR":
        Modelo = MLobjecto.KernelRidge(op.Lamdaridge)

    if op.algoritmo=="RR":
        Modelo.learn(X_train, Y_train)

    #Fit Kernell Ridge Regresión!
    if op.algoritmo=="KRR":
        op.Kx_learning=X_train
        if op.typekernel=="kernel_linear":
```

```
            KX_train=X_train
            K = MLobjecto.kernel_linear(KX_train, KX_train)
            Modelo.learn(K, Y_train)
        if op.typekernel=="kernel_polynomial":
            KX_train=X_train
            K = MLobjecto.kernel_polynomial(KX_train, KX_train,1, op.beta,op.kernelridge_grade)
            Modelo.learn(K, Y_train)
        if op.typekernel=="kernel_gaussian":
            KX_train=X_train
            K = MLobjecto.kernel_gaussian(KX_train, KX_train, op.Sigma)
            Modelo.learn(K, Y_train)

    #Fit Perceptron Alglib!
    if op.algoritmo=='PML':
        Modelo = PML(op.nodosinput,op.nodoshidden,1)
        Modelo.learn(X_train, Y_train,op.WeightDecay)

    if op.algoritmo=="RBF":
        Modelo=RBF(op.nodosinput,op.nodoshidden,1,op.RBF_beta)
        Modelo.learn(X_train, Y_train,op.WeightDecay)

    return Modelo


def forcasting(op,X_validation,Y_validation,h_ahead,Y_validation_ahead,KX_train,y_validation_all_data):
    #Defininiendo estructuras para error de predicción y el valor de la predicción!
    error_mae=np.zeros((h_ahead))
```

```
prediccion=np.zeros((h_ahead))

for it in range(h_ahead):
    Y_validation=Y_validation_ahead[it]

    #Predecir basado en el modelo encajado!
    if op.algoritmo=="KRR":
        if op.typekernel=="kernel_linear":
            Kt = MLobjecto.kernel_linear(X_validation, KX_train)
        if op.typekernel=="kernel_polynomial":
            Kt = MLobjecto.kernel_polynomial(X_validation, KX_train,1, op.beta,op.kernelridge_grade)
        if op.typekernel=="kernel_gaussian":
            Kt = MLobjecto.kernel_gaussian(X_validation, KX_train, op.Sigma)
        DataSet_Predictor_Z= Modelo.pred(Kt)
    else:
        DataSet_Predictor_Z= Modelo.pred(X_validation)

    prediccion[it]=DataSet_Predictor_Z
    error_mae[it]=abs(100*(Y_validation-DataSet_Predictor_Z)/Y_validation)

    if op.ModeloAnalisis=="Serietiempo" or op.ModeloAnalisis=="regresion_retardos":
        if h_ahead>1:
            #Por cada iteración se debe modificar x_validation con el nuevo pronóstico y cambiar el y_validation
            for x_laag in range(0,op.lags-1):
                X_validation[0,x_laag]=X_validation[0,x_laag+1]
            X_validation[0,op.lags-1]=DataSet_Predictor_Z
```

```
            #Recalcular los inputs siguiente!
            for columna in range(0,op.lags):
                for grado in range(0,op.polynomial_grade):
                    X_validation[0,columna+op.lags*(grado)]=X_validation[0,columna]**(grado+1)
                if odataset.makedummis==1:
                    for col in range(5,15):
                        X_validation[0,op.lags*op.polynomial_grade+col-3]=y_validation_all_data[it,col]



    #al final computo el promedio de los errores mape.
    return  np.average(error_mae), prediccion

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
#++++++++++++++++++++++++++++++++++++ INICIO
++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#Inicializando objetos de parámetros para establecer el experimento y objeto para armar dataset !!
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++


#Iniciando parámetros para realizar los experimentos!
oparameters = clsparametrosexperimentales()


#Establezca el nivel de análisis y presentación de la data.
_showplotforcast=1          #1:Mostrar el gráfico con el análisis 0:No mostrar gráfico
```

```
#establecer los parámetros para entrenamiento y validación.
oparameters.buildparameters([0],1,1,1,1,[0.1],"RR","RFO","",1,[1],[1],402,0,181,1,"pickup_add_90..7")



#Instanciar objeto para administrar DATASET!
odataset=clsdataset()

#Abrir archivo en disco para grabar resultado del experimento!
rutalog=rutalog+'\MachineLearningSumary.txt'
File=open(rutalog,'w')
File.write("Algoritmo" + "\t" + "type" +"\t" + "Laags" +"\t" +"Polynomial_grade" +"\t"+ "Sigma"
+"\t"+"Lamdaridge"+"\t" +"Mape_test"+"\t" +"Mape_validation"+"\t"+"Mape_test"+"\t"
+"Mape_validation"+"\t"+"Horizinte learning validación"+"\t"+"Horizinte learning test"+"\t"+"nodos
input"+"\t"+"nodos hidden"+"\t"+"Kernel
Grade"+"\t"+"KRR_beta"+"\t"+"RBF_beta"+"\t"+"weighdecay"+"\t"+"modeloanalizado"+"\n")

#recorrer el objeto de parámetros para evaluar modelos y encontrar el mejor modelo !
besterror=-1
bestmodelo=0
for op in oparameters.items:

    #Inicializando los parámetros para construir el dataset!
    odataset.columndata=oparameters.columndata
    odataset.rows=np.int(0)
    odataset.file=oparameters.file
    odataset.lags=op.lags
    odataset.polynomial_grade=op.polynomial_grade #para caso ridge regresión
```

```
   #Cargando la serie y modelando el dataset basado en los parámetros establecidos de rezagos y grado del
polinomio.
   if op.ModeloAnalisis=="Serietiempo":
      odataset.columndata=27
      odataset.LoadDataSerie("C:\datasets\Tesis\SerieTime_Ocupacion.txt",1,0,0)
   else :
      if op.ModeloAnalisis=="regresion_retardos":
         odataset.columndata=28
         odataset.LoadDataregresionserie("C:\datasets\Tesis\Estadistica_ocupacion_tesis_dumis.txt",1,0)
      else:
         #odataset.LoadDataSerie_pickup("C:\datasets\Tesis\datos_pickup.txt",op.ModeloAnalisis)
         odataset.LoadDataSerie_pickup("C:\datasets\Tesis\datos_pickup_addcancel.txt",op.ModeloAnalisis,0)

   #Actualizando número de columnas basado en los datos encajados!
   op.nodosinput=odataset._columnas  #El número de columnas corresponde al no de nodos capa de entrada de la
red.

   #Calculando el máximo número de observaciones para entrenamiento y validación (separamos los registros para
test).
   op.maxrecord_learning_and_validation=odataset.rows-op.recordstest
   op.maxrecord_learning=op.maxrecord_learning_and_validation - op.recordsvalidation
   op.maxrecord=odataset.rows

   #Establecer el registro de inicio del Análisis #Según las técnicas de Rolling Forcasting Origin!
   if op.Lastperiodoslearning==0:
      op.RecordOriginLearning=0
```

```
else:
    op.RecordOriginLearning=op.maxrecord_learning-op.Lastperiodoslearning

if op.validationtype=="RFO":
    X_train=odataset.DataSet_input[op.RecordOriginLearning:op.maxrecord_learning,:]
    Y_train=odataset.DataSet_output[op.RecordOriginLearning:op.maxrecord_learning]

    Modelo = Entrenar(op,X_train,Y_train)

    _cont=0
    registros=op.recordsvalidation
    scores =  np.zeros((registros))
    h=op.Horizonteprediccion
    h=0

    for rec in range(0,registros-h):

        #Preparar conjunto de test!
        indexmaslearningandvalidaction=op.maxrecord_learning_and_validation-registros+rec
        indexorigen=0

        #En cada Iteración se obtiene el conjunto de validación
        X_validation =odataset.DataSet_input[indexmaslearningandvalidaction:indexmaslearningandvalidaction+1,:]
        Y_validation = odataset.DataSet_output[indexmaslearningandvalidaction:indexmaslearningandvalidaction+1]

        if h==0:
            hpre=1
```

```
        Y_validation_ahead=
odataset.DataSet_output[indexmaslearningandvalidaction:indexmaslearningandvalidaction+hpre]

y_validation_all_data=odataset.DataoutputAll[indexmaslearningandvalidaction:indexmaslearningandvalidaction+hpr
e]

        #Pronosticando en cada iteración!
        predicion=np.zeros(hpre)

scores[_cont],prediccion=forcasting(op,X_validation,Y_validation,hpre,Y_validation_ahead,X_train,y_validation_al
l_data)
        _cont=_cont+1

    #Calculando el error de validación!
    op.Mape_validation = np.average(scores)

    #Escribir en el archivo de resumen de resultado las evaluaciones realizadas!
    CadenaResultado=op.algoritmo + "\t" + op.typekernel +"\t" + str(op.lags) +"\t" +str(op.polynomial_grade)+"\t"
+str(op.Sigma)+"\t" +str(op.Lamdaridge)+"\t" +str(op.Mape_test)+"\t" +str(op.Mape_validation)+"\t"
+str(op.Mape_test)+"\t" +str(op.Mape_validation)+"\t" +str(op.recordsvalidation)+"\t" +str(op.recordstest)+ "\t"
+str(op.nodosinput) + "\t" + str(op.nodoshidden)+ "\t" + str(op.kernelridge_grade)+ "\t" + str(op.beta)+ "\t" +
str(op.RBF_beta)+ "\t" + str(op.WeightDecay)+ "\t" + str(op.ModeloAnalisis)+"\n"
    File.write(CadenaResultado)

    #En cada iteración se selecciona el mejor modelo (Aquel con menor error de validación)!
```

```
        cadena="Algoritmo:"+op.algoritmo + " Inputs:"+" Lambda:"+str(op.Lamdaridge)+" MAPE validación:
"+str(op.Mape_validation)
        op.Modelo=Modelo
        print(cadena)
        if op.Mape_validation<besterror or besterror<0:
            bestparameters=op
            bestmodelo=Modelo
            besterror=op.Mape_validation

#Cerrar el archivo de resumen de resultados!
File.close

#Testear el mejor modelo
#Contador del registros de pronóstico!

_cont=0
registros=bestparameters.recordstest
scores=np.zeros((registros))
h=bestparameters.Horizonteprediccion

X_train=odataset.DataSet_input[bestparameters.maxrecord_learning:bestparameters.maxrecord_learning_and_valida
tion,:]
Y_train=odataset.DataSet_output[bestparameters.maxrecord_learning:bestparameters.maxrecord_learning_and_valid
ation]
Kx_train=bestparameters.Kx_learning
```

```
if bestparameters.algoritmo=="KRR":
    if bestparameters.typekernel=="kernel_linear":
        Kt = MLobjecto.kernel_linear(X_train, Kx_train)
    if bestparameters.typekernel=="kernel_polynomial":
        Kt = MLobjecto.kernel_polynomial(X_train, Kx_train,1,bestparameters.beta,bestparameters.kernelridge_grade)

    if bestparameters.typekernel=="kernel_gaussian":
        Kt = MLobjecto.kernel_gaussian(X_train, Kx_train, bestparameters.Sigma)

    ypred= bestmodelo.pred(Kt)
else:
    if bestparameters.algoritmo=="PML":
        ypred=bestmodelo.pred1(X_train)
    else:
        ypred=bestmodelo.pred(X_train)



plt.clf()
plt.plot(ypred, linewidth = 2, label = 'Predicción', color='red')
plt.plot(Y_train, linewidth = 2, label = 'serie', color='black')
plt.title="Pronóstico sobre serie de validación!"
plt.show()

#Recorrer los registros para calcular el error de pronóstico!
if _showplotforcast==1:
    X_test=odataset.DataSet_input[bestparameters.maxrecord_learning_and_validation:bestparameters.maxrecord,:]
    Y_test=odataset.DataSet_output[bestparameters.maxrecord_learning_and_validation:bestparameters.maxrecord]
```

```
if bestparameters.algoritmo=="RR":
    ypred=bestmodelo.pred(X_test)

print "y_Observado" +"\t"+"y_pronsotico"+"\t"+"MAPE test"
for i in range(ypred.size):
   print str(Y_test[i])+"\t"+str(ypred[i])+"\t"+str(abs(100*(Y_test[i]-ypred[i])/Y_test[i]))

mape_i=[abs(100*(Y_test[i]-ypred[i])/Y_test[i]) for i in range(ypred.size) ]
mape=average(mape_i)
print("MAPE general test:",mape)
plt.clf()
plt.plot(ypred, linewidth = 2, label = 'Predicción', color='red')
plt.plot(Y_test, linewidth = 2, label = 'serie', color='black')
plt.title="Pronóstico sobre serie de validación!"
plt.show()
```