Facultad 2

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Título: *Implementación en MATLAB del algoritmo MTS para problemas de predicción con salidas compuestas.*

Autor:

Zuleyma García Albear.

Tutor:

MSc. Héctor Raúl González Diez.

La Habana, junio 2017
Año 59 de la Revolución

Declaración de Autoría

Declaro que Zuleyma García Albear es el único autor del presente trabajo de Diploma que tiene por título “Implementación en MATLAB del algoritmo MTS para problemas de predicción con salidas compuestas” y se concede a la Universidad de Ciencias Informáticas los derechos patrimoniales de la misma, para hacer uso en su beneficio.

Para que así conste firmo la presente a los ___ días del mes de _____ del año _____.

_____ _____

Zuleyma García Albear. MSc. Héctor Raúl González Diez.

Agradecimientos

(Opcional): Pequeños textos que el autor redacta a su gusto y estilo personal, dedicados a las personas o instituciones que en su opinión lo ameriten.

Zuleyma García Albear

Aquí van todas las personas a quien se le quiera agradecer

Dedicatoria

(Opcional): Pequeños textos que el autor redacta a su gusto y estilo personal, dedicados a las personas o instituciones que en su opinión lo ameriten.

Zuleyma García Albear

Aquí van todas las personas a quien se le quiera dedicar el TRABAJO DE DIPLOMA

Resumen

La presente investigación tiene como objetivo implementar en MATLAB el algoritmo de transformación Multi-target Stacking (MTS) para problemas de predicción con salidas compuestas. En la investigación se partió del estudio de los diferentes algoritmos de predicción con salidas compuestas existentes así como las diferentes herramientas disponibles para su implementación. Los problemas de predicción con salidas compuestas son aquellos que permiten predecir un conjunto de variables de salida de manera simultánea a partir de un conjunto de entrenamiento dado. Se emplearon 12 bases de datos reales para evaluar el algoritmo MTS mediante el promedio del error cuadrático medio entre lo predicho del algoritmo y lo medido en cada base de datos para las variables de salida.

Palabras claves: MTS, MATLAB, problemas de predicción con salidas compuestas.

Índice General

Introducción

Una de las ramas de la inteligencia artificial es el aprendizaje automático el cual permite a las computadoras aprender de forma automática modelos para resolver nuevos problemas o mejorar el comportamiento en problemas ya observados. El objetivo principal de todo proceso en esta rama es utilizar la evidencia conocida para poder crear una hipótesis y dar respuesta a situaciones desconocidas. Los algoritmos de aprendizaje automático pueden clasificarse principalmente, dependiendo del tipo de tarea que se desea resolver, en problemas supervisados o predictivos y no supervisados o descriptivos (Friedman, Hastie, & Tibshirani, 2001).

El aprendizaje automático supervisado intenta definir un modelo matemático a partir de la información de las clases de un conjunto de datos de entrenamiento. Dependiendo del problema que se desea clasificar la tarea puede ser de clasificación binaria (la variable objetivo toma valores de 0,1), clasificación multiclases (la variable objetivo puede tomar valores en un conjunto acotado de 1-m) o regresión (la variable objetivo toma valores reales). Cuando el criterio es una variable cuantitativa se suele hablar de problemas de predicción o regresión, mientras que cuando es una variable cualitativa o categórica se habla entonces de problemas de clasificación.

Nuevos problemas de clasificación han requerido su estudio desde una nueva perspectiva del aprendizaje automático dado que la variable objetivo puede no estar formada por una única etiqueta o una única variable. La forma de modelar las variables de salida en este tipo de problemas es bien diversa y se encuentran en el campo de la predicción estructurada. Estos problemas de predicción estructurada modelan sus variables de salida en formas de: grafos, jerarquías o vectores con valores reales o binarios (Bakir et al., 2007). En el caso de ser un problema con varias variables de salidas modelados en forma de vector el problema se conoce como problemas de predicción con salidas compuestas (Spyromitros-Xioufis, Tsoumakas, William, & Vlahavas, 2014). Los algoritmos de predicción con salidas compuestas obtienen mejores resultados en cuanto al

error predictivo si el mismo problema fuera modelado como múltiples problemas de una sola salida debido a que explotan la interdependencia entre las variables.

Aplicaciones de problemas de predicción con salidas compuestas han sido publicados con resultados prominentes en el ámbito de la predicción de la calidad de la vegetación (D Kocev, 2009) donde se obtienen los datos utilizando el enfoque de “hábitat por hectáreas”. Esta técnica de evaluación rápida produce múltiples puntuaciones que describen la condición de los diversos atributos de la vegetación de un sitio dado. Para el modelado de este tipo de datos se compararon dos enfoques: el primero de ellos donde se aprende un modelo para cada una de las variables de salidas expresada en la puntuación (Modelo clásico) y un segundo caso donde se construye un modelo único que toma en cuenta todas las variables de forma simultánea (Tratamiento de la interdependencia). Los resultados de este trabajo demuestran las ventajas de un enfoque de modelado como problema de predicción con salidas compuestas sobre los modelos clásicos.

Otras áreas de aplicación de este tipo de problemas ampliamente estudiados en la literatura científica podemos mencionar: la quimiometría (Džeroski, Demšar, & Grbović, 2000) al inferir concentraciones de varios analitos de calibración multivariable usando datos multivariados espectrales, la predicción de los precios de los billetes de avión para una determinada aerolínea tomando como variables de entrada detalles de vuelos observados durante varios días como (precio, paradas, fecha de salida) (Spyromitros-Xioufis et al., 2014) y en la predicción de la concentración de metales más costosos de medir utilizando metales que son más baratos en el muestreo.

Un artículo de revisión publicado recientemente sobre los problemas de predicción con salidas compuestas establece dos grandes categorías (Borchani, Varando, Bielza, & Larrañaga, 2015) para este tipo de tarea: El primer enfoque contiene los métodos que transforman el problemas de salidas múltiples en diferentes problemas de predicción de una salida(basados en transformación);

mientras que el otro enfoque se basa en adaptar métodos clásicos conocidos de aprendizaje automático para estimar de manera simultánea las múltiples salidas(basados en adaptación). En ambas categorías los métodos pueden tener en cuenta o no la interdependencia entre las variables de salidas.

En el caso de los algoritmos basados en transformación generalmente se diseñan en dos etapas, la primera de ella permite estimar cada salida de manera independiente para luego en una segunda etapa establecer alguna estrategia que combine las diferentes salidas. Entre los algoritmos más representativos del estado del arte en esta categoría (Por su capacidad predictiva) se encuentra el Multi Target Staking (MTS) (Predicción con salidas compuestas basados en apilamiento por sus siglas en inglés). Este algoritmo fue extendido a la predicción con salidas compuesta luego de haber sido diseñado para problemas de clasificación multietiqueta en el trabajo (Wolpert, 1992). El algoritmo MTS obtuvo los mejores resultados en cinco de las doce bases de datos en los que fue evaluado (Spyromitros-Xioufis et al., 2014).

Otros algoritmos competitivos en el ámbito de la predicción con salidas compuestas basados en transformación son: Regressor Chains (ERC) y (ERCC) (Spyromitros-Xioufis et al., 2014), Curds and Wheys (C&W) (Breiman & Friedman, 1997), Reduced Rank Regression (RRR) (Izenman, 1975), K Nearest Neighbor (KNN-SP) (Pugelj & Džeroski, 2011) (González Diez, Santos, Campos, & Morell Pérez, 2016) Random Linear Combination (RLC) (Tsoumakas, Spyromitros-Xioufis, Vrekou, & Vlahavas, 2014). En sentido general los basados en transformación anteriormente mencionados emplean diferentes regresores de base y estrategias para combinar las salidas. De manera particular el algoritmo MTS fue evaluado usando el regresor de base RandomForest o Bagging.

La implementación del algoritmo MTS y los antes mencionados se encuentran desarrollados en la herramienta MULAN (Tsoumakas, Spyromitros-Xioufis, Vilcek, & Vlahavas, 2011) el cual es un software de código abierto para el aprendizaje automático cuyo objetivo inicial fue el manejo de problemas de

clasificación multi-etiqueta pero este software ha extendido sus funcionalidades para el manejo de datos con salidas compuestas con la incorporación de estos algoritmos. Además Weka al emplearse como librería es posible utilizar sus algoritmos de aprendizaje automático como base para otras herramientas como es el caso de MULAN que utiliza toda la librería de Weka.

Recientemente, como parte del trabajo que desarrollan investigadores del grupo de Inteligencia Artificial de la UCI se han desarrollado nuevos algoritmos de predicción con salidas compuestas, algoritmos basado en la regla de los k-vecinos más cercanos (González Diez, Santos, Campos, & Morell Pérez, 2016) y sus respectivas variantes

.

MATLAB con relación a Weka trata los valores ausentes de manera diferente en la base de datos, Weka ignora las tuplas de los valores ausentes, en tanto en MATLAB corrige los valores ausentes por el valor medio lo cual no compromete el valor de la función de distancia.

Por otra parte Weka utiliza una normalización min-max de los datos en el algoritmo KNN para el cálculo de la función de distancia lo cual es susceptible en datos alejados de la media aritmética motivo por el cual el algoritmo KNN implementado en MATLAB utiliza una normalización de los datos a media 0 y varianza 1. Estos 2 elementos hacen que los resultados del algoritmo KNN de Weka arrojen resultados diferentes al que está implementado en MATLAB.

El algoritmo MTS utilizando el regresor de base KNN no ha sido implementado; los algoritmos desarrollados en MULAN no emplean el regresor de base KNN mientras que los implementados por el grupo de Inteligencia Artificial de la UCI sí lo contienen.

Tomando en cuenta los elementos anteriormente planteados se afirma que:

- El algoritmo MTS muestra buenos resultados para problemas de predicción con salidas compuestas sin embargo su evaluación haciendo uso del regresor de base KNN no ha sido estudiada con anterioridad.
-
- Los algoritmos que se encuentran desarrollados en MATLAB utilizan como regresor de base el KNN mientras que los desarrollados en Weka no lo utilizan.
-
- Los resultados de implementar el algoritmo MTS con el regresor de base KNN en MATLAB arroja valores más exactos en cuanto al error de predicción en comparación si el mismo fuera realizado en Weka.
-

Por la problemática anteriormente expuesta se plantea como **problema a resolver:** ¿Cómo mejorar el error predictivo en el algoritmo de predicción con salidas compuestas basado en trasformación MTS en la herramienta MATLAB?

A partir del problema a resolver se define como **Objeto de estudio** para esta investigación: Algoritmo MTS, y como **Campo de acción**: Algoritmo MTS en la herramienta MATLAB

.

Objetivo General: Implementar en MATLAB el algoritmo de transformación MTS para problemas de predicción con salidas compuestas

.

Objetivos Específicos:

1. Caracterizar el marco teórico-conceptual del algoritmo MTS y sus variantes.

2. Implementar el algoritmo basado en transformación MTS y sus variantes con el regresor de base KNN en la herramienta MATLAB.

3. Validar la solución implementada mediante el diseño de experimentos para evaluar el error de predicción de estos algoritmos.

Aportes de la investigación

Siguiendo los conceptos de ciencia y tecnología, la ciencia se proyecta en la tesis mediante la investigación realizada sobre los distintos algoritmos de predicción con salidas compuestas analizados, así como la propuesta que se ha desarrollado en la herramienta MATLAB. De este modo se sientan las bases teóricas e innovadoras para el desarrollo de investigaciones futuras, para lograr avances en la economía, la biotecnología, la bioinformática, entre otras áreas de aplicación de los modelos de predicción estudiados. Con los nuevos algoritmos implementados en la herramienta MATLAB, se beneficiará la comunidad científica internacional, debido a que estos estarán a su disposición para que puedan valorarlos, utilizarlos, así como poder compararse con los resultados obtenidos en los experimentos realizados durante la investigación.

Capítulo 1 Fundamentación teórica

1.1 Introducción

En este capítulo se realiza un estudio sobre los problemas de predicción con salidas compuestas. Se describe un estado del arte de los diferentes tipos de algoritmos que se basan en dichos problemas y los regresores de base. Además, se describen las herramientas a utilizar para el desarrollo e implementación de la investigación y las diferentes pruebas estadísticas que se pueden realizar para evaluar y comparar el error predictivo de los algoritmos utilizados.

1.2 Problemas de predicción con salidas compuestas

En los últimos años, en la rama del aprendizaje automático, han aparecido problemas de la vida práctica donde se hace necesario modelar la predicción a través de estructuras complejas en su salida. Estos problemas están relacionados con la necesidad de predecir valores de salidas en forma de: grafos, jerarquías, vectores con valores reales o binarios entre otras. Como término general este tipo de tarea es conocida como problemas de predicción estructurada.

Como caso particular de problemas con salidas estructuradas se encuentran los problemas de predicción con salidas compuestas donde se trata de predecir un vector de salida (la variable objetivo es un vector con valores reales). Por lo tanto, un problema de predicción con salidas compuesta intenta predecir de forma simultánea y con un único modelo todos los atributos del espacio de salida expresados en forma vectorial y queda expresado de la siguiente manera

:

Sean x, y dos vectores aleatorios del espacio de entrada R^ddonde $x = [x_1, \dots x_d]$ y $y = [y_1, \dots y_m]$en el espacio de salida R^m para una instancia determinada. Dado el conjunto $D = \{(x^1, y^1), \dots (x^n, y^n \subset R^d \, X \, R^m)\}$de n instancias de entrenamiento,

el objetivo en un problema de predicción de salidas compuestas es aprender un modelo $h : R^d \rightarrow R^m$ de modo que dada una instancia con entrada conocida x^q es posible predecir, a través de un único modelo, todas las salidas $\hat{y}^q = h(x^q)$ de forma simultánea.

La principal diferencia entre éstos es el número de modelos entrenados: un modelo separado para cada una de las tareas frente a un único modelo entrenado durante todo el problema. El aprendizaje con salidas compuestas tiene como meta predecir las características de las salidas y describir explícitamente su relación con las características descriptivas. Además, se describe implícitamente las relaciones entre características de las salidas.

El aprendizaje con salidas compuestas capta implícitamente las dependencias entre las salidas y los representa en un modelo único generado. A través de este modelo, se determina el efecto de las características descriptivas de todas las salidas, y analizar las relaciones, ya sea lineal o no lineal, entre las salidas (o grupos de salidas). En caso de que las salidas estén relacionadas, se obtiene información sobre estas relaciones.

Varios métodos estándar de aprendizaje han sido extendidos a problemas de predicción con salidas compuestas en los últimos años como ST, C&W, MTS, ERC, ERCC.

1.3 Algoritmos de predicción con salidas compuestas basados en transformación

1.3.1 Ensemble of Regressor Chains (ERC)

El método de Regressor Chains (RC) se basa en la idea de modelos de encadenamiento de salidas simples. El entrenamiento del RC consiste en seleccionar una cadena aleatoria (permutación) del conjunto de variables de salida y luego construir un modelo de regresión separado para cada salida.
Una propiedad notable de RC es que es sensible en el ordenamiento de la cadena seleccionada. El principal problema que surge de la utilización de una

sola cadena al azar, es que las salidas que aparecen antes en una cadena no pueden modelar posibles relaciones estadísticas con las salidas que aparecen más adelante en la cadena. Además el error de predicción es probable que sea propagado y amplificado a lo largo de la cadena cuando hacemos predicciones para una nueva instancia de prueba. Para mitigar estos efectos, se propone un esquema de conjunto llamado Ensemble of Classifier Chains donde una prueba de k (típicamente k=10) modelos Classifier Chains (CC) con cadenas de diferente orden son construido sobre muestras de arranque del conjunto de entrenamiento y las predicciones finales provienen de la votación por mayoría. Este esquema ha demostrado mejorar consistentemente la exactitud de un solo CC en el dominio de clasificación. Se aplica la misma idea (sin muestreo) en RC y se calcula las predicciones finales tomando la media de las estimaciones de k para cada objetivo. El método resultante se llama Ensemble of Regressor Chains.

1.3.2 Multi-target Stacking (MTS)

El entrenamiento en MTS está compuesto por dos etapas fundamentales: en la primera se aprenden m modelos $h_j : X \rightarrow R$ de salidas simples independientes como un Single Target (ST). Sin embargo, en lugar de usar directamente estos modelos para la predicción, MTS tiene una segunda etapa de entrenamiento adicional donde un segundo conjunto de m modelos objetivos $h^{x}_{j} : X \times R^{m-1} \rightarrow R$ por cada salida Y_j. Para predecir las salidas de una instancia conocida x^q se aplica la primera etapa obteniéndose un vector de salida $\tilde{y}^q$, luego al aplicar la segunda etapa en un vector de una entrada $x^{x}_{j}{}^{q}$ transformado se produce el vector final de salida $\hat{\hat{y}}_j{}^{q}$.

1.3.3 Single Target (ST)

En el método ST, un modelo h con salidas compuestas es convertido a m modelos de salidas simples h_j: X→ R donde cada modelo h_j es entrenado en una transformación del conjunto de entrenamiento$D = (x^1, y_j^1), \dots, (x^n, y_j^n)$ para predecir el valor de una variable de salida simple Y_j . De esta manera, las

variables de salida son predichas independientemente y no se explota la relación entre ellas.

1.3.4 Curds and Whey (C&W)

La idea general del algoritmo Curds and Whey es tomar las regresiones de mínimos cuadrados, y luego de modificar los valores previstos de esas regresiones por la contracción de ellos, utilizando las correlaciones canónicas entre las variables de salida y las variables predictoras. Breiman y Friedman en su trabajo propone la contracción simultánea tanto en el espacio de entrada como en el de salida.

El enfoque estándar de determinación de la matriz de coeficientes C es a través del OLS, donde el resultado que se obtiene es equivalente a la regresión de cada una de las variables de salida a través de las variables predictoras de forma separada. Para este enfoque el error predictivo es pobre en presencia de una correlación alta entre las variables predictivas o cuando existen una cantidad significativa de este tipo de variables.

1.3.5 Reduce Rank Regretion (RRR)

RRR es un método de estimación explícita en la regresión multivariante, que tiene en cuenta la restricción de rango reducido en la matriz de coeficientes. Una incidencia directa del RRR sobre la regresión lineal es el número de restricciones lineales sobre la matriz de coeficientes C de la regresión, permitiendo que el número de parámetros efectivos se reduzca y la eficiencia de la estimación aumente.

1.3.6 MTS Corrected (MTSC) y ERC Corrected (ERCC)

Una modificación de los algoritmos MTS y ERC son MTS Corrected (MTSC) y ERC Corrected (ERCC), se plantea que tanto MTS y ERC se basan en la misma idea central de tratamiento de las otras salidas de predicción como variables de entrada adicionales que aumentan el espacio de entrada original. Estas meta variables difieren de las variables de entrada común en el sentido de que

mientras que sus valores reales se encuentran disponibles en el tiempo de formación, se están perdiendo en la predicción.

Con el fin de enfrentar el problema antes mencionado se propone en unas modificaciones mediante el empleo de un procedimiento que se asemeja a la de f-Fold cross-validation con el fin de obtener estimaciones imparciales fuera de la muestra de la meta-variable. El enfoque de validación cruzada evita el problema del enfoque hold-out como todos los ejemplos de entrenamientos son usados en el segundo estado ST de los modelos MTS y el RC en los modelos ERC. Se espera que en comparación con el uso de los valores reales o estimados en la muestra, las estimaciones de validación cruzada se aproximarán a la distribución de las estimaciones obtenidas en el momento de la predicción de forma más precisa y por lo tanto será más útil para el modelo.

1.3.7 Distance Metric Learning for Multi-target Regression (DMLMTP)

El propósito del DMLMTP consiste en aprender una función de distancia para problemas de regresión con salidas múltiples. El aprendizaje de la función de distancia se basa en la heurística de preservar las relaciones de orden, para cada objeto del conjunto de entrenamiento, considerando las distancias de los vectores del espacio de entrada y salida respectivamente.

El algoritmo DMLMTP se inspiran en el problema de aprendizaje de distancia métrica .Este enfoque, tiene la particularidad de que funciona bien en los problemas de regresión a diferencias a los algoritmos de aprendizaje métrico de distancia comunes.

En el clásico aprendizaje supervisado de distancia métrica, el objetivo es aprender un Mahalanobis función de distancia equivalente a transformar el espacio de datos de entrada en mejorar la correlación entre las variables de entrada y salida. Para asegurar que la función de distancia satisfaga las propiedades de ser una métrica, debe cumplirse que M>= 0 para que cualquier problema de aprendizaje métrico a distancia tenga que considerar esta restricción hay dos vías, una M = LTL [18,17] o M = ATW A. En la primera opción

de descomposición, la transformación del espacio de datos de entrada a través de la matriz L mientras que en la segunda descomposición, se aprende la matriz diagonal W para transformar el espacio de entrada. En el caso de que la matriz arbitraria A sea igual a la identidad, el aprendizaje del problema de distancia es equivalente a la distancia de ponderación euclidiana.

Después de realizar un estudio de los diferentes algoritmos de predicción con salidas compuestas, se identifica que los algoritmos MTS, MTSC, ERC y ERCC explotan la interdependencia entre las variables de salida y muestran los mejores errores predictivos, con respecto a los que tratan cada variable de salida de forma independiente. Se decide escoger el algoritmo MTS ya que como se dijo anteriormente este obtuvo los mejores resultados en cinco de las doce bases de datos en los que fue evaluado con el regresor de base Bagging o RandomForest.

1.4 Métodos de regresión

1.4.1. Regresión lineal

En estadística la regresión lineal o ajuste lineal es un modelo matemático usado para aproximar la relación de dependencia entre una variable dependiente Y, las variables independientes Xi y un término aleatorio ε.

Existen diferentes tipos de regresión lineal que se clasifican de acuerdo a sus parámetros. Uno es la regresión lineal simple (solo maneja una variable independiente, por lo que sólo cuenta con dos parámetros), otra es la que permite trabajar con una variable a nivel de intervalo o razón y por último cuando posible analizar la relación entre dos o más variables a través de ecuaciones, la que se denomina regresión múltiple o regresión lineal múltiple. Constantemente en la práctica de la investigación estadística, se encuentran variables que de alguna manera están relacionadas entre sí, por lo que es posible que una de las variables pueda relacionarse matemáticamente en función de otra u otras variables.

1.4.2. Máquina soporte vectorial

Una Máquina de Soporte Vectorial (SVM) aprende la superficie decisión de dos clases distintas de los puntos de entrada. Como un clasificador de una sola clase, la descripción dada por los datos de los vectores de soporte es capaz de formar una frontera de decisión alrededor del dominio de los datos de aprendizaje con muy poco o ningún conocimiento de los datos fuera de esta frontera. Los datos son mapeados por medio de un kernel Gaussiano u otro tipo de kernel a un espacio de características en un espacio dimensional más alto, donde se busca la máxima separación entre clases. Esta función de frontera, cuando es traída de regreso al espacio de entrada, puede separar los datos en todas las clases distintas, cada una formando un agrupamiento.

La teoría de las Máquinas de Soporte Vectorial (SVM por su nombre en inglés Support Vector Machines) es una nueva técnica de clasificación y ha tomado mucha atención en años recientes. La teoría de la SVM está basada en la idea de minimización de riego estructural (SRM). En muchas aplicaciones, las SVM han mostrado tener gran desempeño, más que las máquinas de aprendizaje tradicional como las redes neuronales y han sido introducidas como herramientas poderosas para resolver problemas de clasificación.

Una SVM primero mapea los puntos de entrada a un espacio de características de una dimensión mayor (i.e.: si los puntos de entrada están en R2 entonces son mapeados por la SVM a R3) y encuentra un hyperplano que los separe y maximice el margen m entre las clases en este espacio.

1.4.3 KNN

El algoritmo de aprendizaje automático K-Vecinos más Cercanos (KNN por sus siglas en inglés) está considerado uno de los de mejor desempeño en el ámbito del aprendizaje automático. Al mismo tiempo, el KNN construye un modelo sencillo para resolver problemas de predicción el cual está basado en los objetos del conjunto de datos de entrenamiento. En KNN para un nuevo ejemplo, se asigna al atributo de la clase, el valor medio de las clases de los objetos que se encuentran dentro de la frontera de los K vecinos más cercanos. En general la distancia Euclidiana es empleada para medir la proximidad entre el nuevo

ejemplo y los que conforman el conjunto de datos de entrenamiento. Sin embargo, a pesar del buen desempeño del KNN para problemas de predicción este no ha sido ampliamente utilizado para problemas donde se desee predecir un vector de salida. En la definición básica del KNN-SP se considera una variante simple de modelo de transformación.

1.4.4 Multi-objective Bagging (Clusbag) y Multi-objective Random Forest (MORF)

Clusbag y MORF surgen al aplicar métodos de ensamble Bagging y Random Forest respectivamente al método Multi-objective decision trees (MODTs) Los métodos de ensamble no son más que un conjunto de clasificadores construidos con un algoritmo dado. Cada nuevo ejemplo se clasifica mediante la combinación de las predicciones de cada clasificador desde el conjunto. Éstas se pueden combinar tomando el promedio (para las tareas de regresión) o el voto de la mayoría (para las tareas de clasificación), como se describe por Breiman, o tomando combinaciones más complejas.

Para aplicar Bagging a MODTs, el procedimiento de Predictive Clustering Trees PCTs (Ei) se utiliza como una clasificadora base. Para la aplicación de Random Forests, se sigue el mismo enfoque, cambiando el procedimiento BestTest de tomar un subconjunto de tamaño aleatorio f(x) de todos los atributos posibles.
Con el fin de combinar la salida de las predicciones por las clasificadoras bases, se toma la media de la regresión, y se aplica un voto distribución de probabilidad en lugar de una simple mayoría de votos para la clasificación, según lo sugerido por Bauer y Kohavi. Esta combinación de funciones se generaliza trivialmente para el caso en que existan varias variables de salida. Cada conjunto consta de 100 árboles, que son sin podar. Para la construcción de los bosques al azar, el parámetro f(x) se establece en log2(x) + 1 como en Breiman.

1.4.5. Reptree

Los árboles de clasificación se emplean para asignar sujetos a las clases de una variable dependiente a partir de sus mediciones en uno o más predictores. Modernamente, los árboles de clasificación (AC) constituyen uno de los recursos

instrumentales básicos de la llamada "minería de datos". Producen cortaduras en los regresores para predecir o explicar variables dependientes discretas (usualmente binarias) y constituyen, por la interpretación inmediata de sus resultados y por su condición "no paramétrica", una opción favorable entre otras alternativas como el análisis discriminante, el análisis de clústeres o la regresión logística binaria o politómica.

Los AC son jerarquías de cortes que se construyen a partir de los predictores, de modo que se maximice cierto criterio de asociación con la variable de respuesta. Cada cortadura da lugar a una partición de los sujetos en 2 grupos: avanzando en pasos sucesivos a lo largo del árbol jerárquico de cortaduras se llega a la clasificación final.

El uso de los árboles de clasificación no es frecuente en el ámbito de las probabilidades o del reconocimiento de patrones, pero se han extendido considerablemente en el ámbito del diagnóstico, las ciencias de la computación, la taxonomía y la teoría de la decisión. Los árboles tienen una expresión gráfica que facilita su interpretación.

Después de haber realizado un estudio sobre algunos regresores de base existentes se decide utilizar el regresor de base KNN por ser uno de los de mejor desempeño en el ámbito del aprendizaje automático. Además como se dijo anteriormente, el algoritmo MTS que fue el seleccionado a implementar, no ha sido implementado con este regresor.

1.5. Herramientas para la resolución de problemas de predicción con salidas compuestas.

1.5.1. Matlab

Matlab es un entorno de computación y desarrollo de aplicaciones totalmente integrado, orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos. Cuenta con un lenguaje de programación propio (lenguaje M), disponible para las plataformas Unix, Windows, Mac OS X y GNU/Linux. Matlab

integra análisis numérico, cálculo matricial, proceso de señal y visualización gráfica en un entorno completo donde los problemas y sus soluciones son expresados del mismo modo en que se escribirían tradicionalmente, sin necesidad de hacer uso de la programación tradicional. Dentro de sus características se encuentran: cálculos intensivos desde un punto de vista numérico, gráficos y visualización avanzada, lenguaje de alto nivel basado en vectores, arreglos y matrices; y una colección muy útil de funciones de aplicación.

1.5.2 Weka

Weka es una extensa colección de algoritmos de máquina de aprendizaje para tareas de minería de datos implementados en Java y desarrollado por la universidad de Waikato, Nueva Zelanda. Incluye herramientas para el pre-procesamiento de los datos, clasificación, regresión, agrupamiento, reglas de asociación y visualización. Weka es un software de código abierto publicado bajo la licencia GNU. Los algoritmos pueden ser aplicados directamente a un conjunto de datos o dataset desde la interfaz gráfica del programa (Java Swing), cargarlos desde el shell o utilizar los códigos independientes que se proporcionan mandándolos llamar desde nuestro propio programa en Java, utilizándolos de la forma en que indica la documentación. Es posible agregar códigos para extender la funcionalidad del paquete ya que proporcionan el código fuente completo.

1.5.3 Mulan

Mulan es un software de código abierto para máquinas de aprendizaje cuyo objetivo principal es el trabajo con datos multi-etiquetas. Ofrece los algoritmos que existen sobre clasificación multi-etiqueta y ranking de etiquetas, así como un framework que contiene un compendio de medidas para evaluar la clasificación multietiqueta mediante la validación hold-out y la validación cruzada. Mulan está escrita en Java y se construye encima de Weka para emplear todos los recursos que contiene sobre algoritmos de aprendizaje supervisado aunque esta es independiente. Una característica exclusiva de la librería Mulan es la introducción reciente de un paquete de experimentos que tiene como objetivo

reproducir los resultados de los experimentos en un documento de aprendizaje multi-etiqueta. Además, da soporte a la clasificación multi-etiqueta siendo un aporte significativo debido a la no existencia en la mayoría de las plataformas de aprendizaje actuales.

Luego de haber realizado un estudio sobre alguna de las herramientas disponibles para llevar a cabo la implementación del algoritmo MTS con el regresor KNN, se escoge MATLAB ya que presenta un exquisito trabajo con cálculos intensivos desde un punto de vista numérico, gráficos y visualización avanzada, lenguaje de alto nivel basado en vectores, arreglos y matrices; y una colección muy útil de funciones de aplicación. Además, como se dijo anteriormente, en esta herramienta el regresor de base ya escogido, KNN, muestra los valores de predicción más exactos que en las demás.

1.6 Bases de datos y medidas de evaluación.

En el trabajo (Spyromitros-Xiou, Tsoumakas, Groves, & Vlahavas, 2014) se proponen varios algoritmos de predicción con salidas compuestas los cuales fueron evaluadas con 12 bases de datos estándares que se han establecido en este ámbito.

Estas bases de datos se relacionan en la tabla 1 donde la primera columna indica el nombre de la base de datos, la segunda se refiere al número de observaciones de la base de datos. En esta columna además se indica si los datos están particionados en datos de prueba (test) y datos de entrenamiento (train) o si la base de datos contiene todos los datos los cuales deben ser particionados para los experimentos. Para particionar los datos se utiliza un método de validación cruzada el cual se conoce como K-Fold Cross Validation (lo señalamos como CV). La 3ra y 4ta columna indica la cantidad de atributo en el espacio de entrada y salida respectivamente. La última columna brinda una breve explicación del dominio de aplicación en el cual fueron colectados los datos.

Dataset	Instancias	Atributos de entrada	Atributos de salida	Dominio
water-quality	1060/cv	16	14	Abundancia de 14 plantas y especies animal en ríos de Eslovenia.
scm20d	7463/1503	61	16	Predice los precios de los productos futuros en una competencia.
scm1d	8145/1658	280	16	Predice los precios de los productos futuros en una competencia.
rf2	4108/5017	576	8	Predice el comportamiento de las redes fluviales de ríos durante48horas.
rf1	4108/5017	64	8	Predice el comportamiento de las redes fluviales de ríos durante 48horas.
oes97	343/cv	263	16	Estima el número de empleados a tiempo completo para una ciudad.
oes10	403/cv		16	Estima el número de empleados a tiempo completo para una ciudad.
flare2	1066/cv	10	3	Veces que se producen destellos repentinos en la superficie del sol durante 24 h.
flare1	323/cv	10	3	Veces que se producen destellos repentinos en la superficie del sol durante 24 h.
edm	154/cv	16	2	Generación de electricidad.

atp7d	188/108	411	6	Precio de los boletos de avión para una aerolínea durante intervalos de tiempo.
atp1d	201/136	411	6	Precio de los boletos de avión para una aerolínea durante intervalos de tiempo.

Tabla 1 Bases de datos para problemas de predicción con salidas compuestas

Como medida de evaluación en la mayoría de los trabajos se emplean el promedio del error relativo cuadrático medio (aRRMSE) entre lo predicho por el algoritmo y lo medido en cada base de datos para las variables de salidas. El mismo se determina de la siguiente manera:

$$a\mathrm{RRMSE}(\mathrm{h};\mathrm{D}_{\mathrm{test}}) = \frac{1}{q}\sum_{j=1}^{q}\sqrt{\frac{\sum_{(x,y_j)\in D_{test}}(h(x_j)-(y_j))^2)}{\sum_{(x,y_j)\in D_{test}}(\overline{\mathrm{Y}}_{\mathrm{j}}-\mathrm{y}_{\mathrm{j}})^2}} \quad (7)$$

1.7 Metodologías de evaluación de algoritmos de aprendizaje automático

En esta sección se exponen las diferentes metodologías empleadas para la evaluación y comparación de los resultados en el ámbito del aprendizaje automático.

1.7.1 Validación cruzada

La validación cruzada es un método de evaluación de modelos que se emplea para mejorar el error predictivo. El problema de la evaluación del error es que no se conoce que tan bien el modelo ha aprendido para predecir datos desconocidos. Una forma de solucionar este problema es no emplear todo el conjunto de datos para entrenar el modelo. Una parte del conjunto de datos es

removido del conjunto de entrenamiento y empleado para la prueba, una vez que el modelo haya sido entrenado.

El método hold-out es el más simple de los tipos de validación cruzada. Este consiste en dividir en dos conjuntos complementarios los datos de muestra, realizar el análisis de un subconjunto (denominado datos de entrenamiento o training set), y validar el análisis en el otro subconjunto (denominado datos de prueba o test set), de forma que la función de aproximación sólo se ajusta con el conjunto de datos de entrenamiento y a partir de aquí calcula los valores de salida para el conjunto de datos de prueba (valores que no ha analizado antes). La ventaja de este método es que es muy rápido a la hora de computar. Sin embargo, este método no es demasiado preciso debido a la variación del resultado obtenido para diferentes datos de entrenamiento. La evaluación puede depender en gran medida de cómo es la división entre datos de entrenamiento y de prueba, y por lo tanto puede ser significativamente diferente en función de cómo se realice esta división. Debido a estas carencias aparece el concepto de validación cruzada (Schneider, 1997).

En la validación cruzada de K iteraciones o K-Fold cross-validation los datos de muestra se dividen en K subconjuntos. Uno de los subconjuntos se utiliza como datos de prueba 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 prueba. Finalmente se realiza la media aritmética de los resultados de cada iteración para obtener un único resultado. Este método es muy preciso puesto que evaluamos a partir de K combinaciones de datos de entrenamiento y de prueba, pero aun así tiene una desventaja, y es que, a diferencia del método de retención, es lento desde el punto de vista computacional. En la práctica, la elección del número de iteraciones depende de la medida del conjunto de datos. Lo más común es utilizar la validación cruzada de 10 iteraciones (10-fold cross-validation) (Joanneum, 2005).

Este método consiste al dividir aleatoriamente el conjunto de datos de entrenamiento y el conjunto de datos de prueba. Para cada división la función de aproximación se ajusta a partir de los datos de entrenamiento y calcula los valores de salida para el conjunto de datos de prueba. El resultado final se corresponde a la media aritmética de los valores obtenidos para las diferentes divisiones. La ventaja de este método es que la división de datos entrenamiento-prueba no depende del número de iteraciones. Pero, en cambio, con este método hay algunas muestras que quedan sin evaluar y otras que se evalúan más de una vez, es decir, los subconjuntos de prueba y entrenamiento se pueden solapar (Moore, 2001).

La validación cruzada dejando uno fuera o Leave-one-out cross-validation (LOOCV) implica separar los datos de forma que para cada iteración se tiene una sola muestra para los datos de prueba y todo el resto conformando los datos de entrenamiento. La evaluación viene dada por el error, y en este tipo de validación cruzada el error es muy bajo, pero en cambio, a nivel computacional es muy costoso, puesto que se tienen que realizar un elevado número de iteraciones, tantas como N muestras tengamos y para cada una analizar los datos tanto de entrenamiento como de prueba (Schneider, 1997)

1.7.2 Comparación estadística de múltiples clasificadores

Siguiendo las recomendaciones de (Demšar, 2006), el método estadístico común para esta tarea es ANOVA para análisis paramétrico. Una de las desventajas de ANOVA es que está basada en asunciones, las cuales en el ámbito experimental de los algoritmos de aprendizaje automático, son posiblemente violadas. Primeramente ANOVA asume que las muestras son extraídas de distribuciones normalizadas. La segunda y más importante asunción de ANOVA es la esfericidad. La esfericidad es una propiedad que requiere que las variables aleatorias tengan igual varianza y que debido a la naturaleza de los algoritmos de aprendizaje y las colecciones de datos no se puede dar por hecho. Violaciones de estas asunciones tiene un efecto aún mayor

en las Pruebas post-hoc, por lo que ANOVA no es frecuente su uso para estudios experimentales de aprendizaje automático.

Un equivalente no paramétrico de ANOVA es la Prueba de Friedman (Friedman M. , 1937)la cual jerarquiza los algoritmos por cada colección de datos separadamente y en caso de empate asigna un rango promedio. Tomando r_i^jcomo el rango del algoritmo j de k posibles algoritmos en la i-ésima colección de datos de N colecciones de datos, la Prueba de Friedman compara los rasgos promedios de los algoritmos$R_j = \frac{1}{N}\sum r_i^j$. La hipótesis nula afirma que todos los algoritmos son equivalentes y por eso sus rangos R_j deberían ser iguales. Bajo esta hipótesis nula, la estadística de la Prueba de Friedman se calcula:

$$X^2{}_F = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right]$$

De acuerdo a X_F^2con $k-1$ grados de libertad cuando k y N son lo suficientemente grandes $k > 5$, $N > 10$ Si la hipótesis nula es rechazada se puede proceder con una Prueba de post-hoc. La Prueba de Nemenyi (Nemenyi, 1963) es similar a la de Tukey para ANOVA y es usado cuando los clasificadores son comparados entre sí. El desempeño de cualesquiera dos mediciones modeladas en el experimento es significativamente diferente si la diferencia entre estas excede al menos la Diferencia Crítica CD. Los valores críticos son determinados usando el estadístico del rango Studentized dividida por $\sqrt{2}$. Esta diferencia crítica se calcula de la siguiente manera:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$$

Teóricamente la Prueba de Friedman no paramétrica tiene menos poder que el ANOVA paramétrico cuando las asunciones de ANOVA se cumplen, en caso contrario no necesariamente. Friedman en su trabajo (Friedman M. , 1940) realizó 56 experimentos relacionados con diversos problemas independientes y demostró que ambos coincidían generalmente. Cuando uno encontraba un nivel de significación en $p < 0.01$el otro en al menos $p < 0.05$respectivamente. Solamente en dos casos ANOVA encontró nivel de significación que era insignificante para Friedman, mientras que en caso contrario ocurrió en 4 pruebas.

Se selecciona la prueba de Friedman para determinar si existen diferencias entre los algoritmos evaluados debido a que para utilizar ANOVA se deben cumplir las asunciones, lo cual no se asegura siempre para dominios específicos, además las diferencias entre los resultados de ambas no es significativa. Así como se emplea el post-hoc de Nemenyi para determinar si existen diferencias significativas entre los algoritmos evaluados, debido a que muestra una representación gráfica, lo que facilita el análisis y comprensión de los resultados.

1.8. Conclusiones del capítulo

Luego del análisis del marco teórico de la investigación se puede concluir que:

-Los algoritmos MTS, MTSC, ERC y ERCC son los de mejores resultados en cuanto al error de predicción.

-Se escoge el MTS ya que este obtuvo los mejores resultados en cinco de las doce bases de datos en los que fue evaluado.

- Se emplea para evaluar los modelos predictivos la validación cruzada de *k* iteraciones, teniendo en cuenta que es el más empleado en la evaluación de los algoritmos del estado del arte. Además este método es más preciso que el *hold-out*, puesto que evalúa a partir de *k* combinaciones de datos de entrenamiento

y de prueba, todo el conjunto de datos y computacionalmente es menor el costo que el Leave-one-out.

-Para a realizar la comparación de los algoritmos se utilizará la prueba no paramétrica de Friedman, con el fin de conocer si hay diferencias entre estos.

-Se utiliza el análisis post-hoc de Nemenyi, con el fin de identificar si existen diferencias significativas entre los algoritmos evaluados.

Capítulo 2: Propuesta de implementación del Algoritmo MTS.

2.1 Introducción

En este capítulo se realiza una descripción teórica del algoritmo MTS además del modelo conceptual de la propuesta de solución. También las especificidades de su implementación.

2.2 Descripción teórica del algoritmo MTS e implementación.

El método de Multi-Target Stacking (MTS) fue desarrollado tomando como base el generalized stacked para clasificación multi-etiqueta. El entrenamiento en MTS está compuesto por dos etapas fundamentales: en la primera se aprenden m modelos $h_j : X \rightarrow R$de salidas simples independientes y en la segunda se aprende un segundo conjunto de $m$meta-modelos $h^{x}_j : X \times R^{m-1} \rightarrow R$por cada salida Y_j. Para predecir las salidas de una instancia desconocida x^qse aplica la primera etapa obteniéndose un vector de salida $\tilde{y}^q$, luego al aplicar la segunda etapa en un vector de una entrada $x^{x}_j{}^q$transformado se produce el vector final de salida $\hat{\hat{y}}_j{}^q$.

En MTS cada modelo objetivo h^{*}_{j} está aprendiendo en un conjunto de entrenamiento transformado D^{*}_{j}= $(x^{*1},y_j{}^{1}),\ldots,(x^{*n},y_j{}^{n})$ donde $x_j^{*i} = [x_1^i, \ldots, x_n^i, \hat{y}_1^i, \ldots, \hat{y}_{j-1}^i, \hat{y}_{j+1}^i, \ldots, \hat{y}_m^i]$ son expandidos en vectores de entradas consistente del vector de entrada original de los ejemplos de entrenamiento argumentado por *m*-1 predicciones de las restantes variables de salidas obtenidas por las primeras etapas de los modelos. Intencionalmente se diferencia levemente este método del multi-etiqueta por el punto de no incluir las predicciones de la primera etapa del modelo para la variable de salida Yj en el espacio de entrada de la segunda etapa del modelo para estas variables, desde este se debería añadir información redundante para la segunda etapa de los modelos. Obtener los predictores para una instancia desconocida x^{q}, las primeras etapas de los modelos son primero aplicados y luego se obtiene un vector de salida $\hat{y}^q = [\hat{y}_1^q, \ldots, \hat{y}^m] = [h_1(x^q), \ldots, h_m(x^q)]$. Posteriormente en la segunda etapa de los modelos se aplica una transformación en el vector de salida,

$x_j^{*q} = [x_1^q, \ldots, x_n^q, \hat{y}_1^q, \ldots, \hat{y}_j^q, \ldots, \hat{y}_{j-1}^q, \hat{y}_{j+1}^q, \ldots, \hat{y}_m^q]$ Para producir el de salida final, $\hat{\hat{y}}_j^q = [h_1^*(x^{*q}, \ldots, h_m^*(x^{*q})]$.

Pseudo-código

Data: Instancia desconocida x^q, primero y segundo estado de modelos h_j y $h_j^*, j = 1..m$

Result: Vactor de Salida $\hat{\hat{y}}^q$ del segundo estado de modelos

$\hat{y}^q = \hat{\hat{y}}^q = 0$ // Inicializar el primero y segundo vector de salida ;

Aplicar primer estado de modelos ;

for $j = 1, ..., m$ **do**

  $\hat{y}^q = h_j(x^q)$

$x^{*q} = [x^q, \hat{y}^q]$ // Concatenar x^q y el vector de salida $\hat{y}^q$;

Aplicar segundo estado de modelos ;

for $j = 1, ..., m$ **do**

  $\hat{\hat{y}}^q = h_j^*(x^{*q})$

2.3. Integración del MTS en MATLAB.

Se implementaron tres variantes del algoritmo MTS con el regresor KNN llamados: MTS (k), MTS-Inv (k) y MTS-Gau (k) llamados MTS con función de pesado, con pesado inverso y pesado gaussiano respectivamente para posteriormente comparar esas versiones a ver cuál de estos es el de mejores resultados en cuanto al error de predicción.

En el algoritmo KNN se predice una variable de salida a partir de la media de los k-vecinos más cercanos sobre el conjunto de atributos de entrada. Su generalización a problemas de predicción con salidas compuestas consiste en determinar el valor medio para cada salida de los k-vecinos más cercanos.

Para determinar la proximidad entre cada objeto del conjunto de entrenamiento y los objetos a predecir se emplea una función de distancia que en general se llama la distancia euclidiana para datos numéricos. Esta distancia euclidiana tiene variantes en su fórmula ya que esta se puede emplear en tres variantes que fueron las que se utilizaron para implementar la solución:

$$\tilde{y} = \frac{\sum_{k=1}^{K} w\left(d\left(x_i, x_j\right)\right)\vec{y}}{\sum_{k=1}^{K} w\left(d\left(x_i, x_j\right)\right)}$$

Para la variante euclidiana pesada.

$$w(d(x_i, x_j)) = e^{-d^2}$$

Para la variante de pesado gaussiano.

$$w(d(x_i, x_j)) = \frac{1}{1 + d^2}$$

Para la variante de pesado inverso.

En la implementación se llevaron a cabo 2 clases: la primera se llama demo.m que es la clase principal que realiza la llamada a la clase experiment2.m dentro de la cual contiene métodos como knnregresion.m y knnregresionkernel.m para las dos etapas del algoritmo MTS y un knnregresionkernelMTS para ejecutar la predicción. Además de un método Computed_error para escribir el error de predicción cometido mediante la fórmula del promedio del error cuadrático medio.

2.4. Conclusiones del capítulo

Con la realización de este capítulo se pudo realizar una descripción teórica del algoritmo MTS para su mejor entendimiento además de los detalles de la implementación realizada.

Capítulo 3 Experimento y Resultados.

3.1Introducción

En este capítulo se expone el experimento realizado. Primeramente se describe la medida de evaluación, luego se realiza una descripción de las bases de datos utilizadas para el experimento y se muestra un ejemplo de un caso de estudio empleando las variantes del algoritmo MTS con la base de datos edm.

3.2 Evaluación

Como medida de evaluación se empleó el promedio del error cuadrático medio (aRRMSE) entre lo predicho por el algoritmo y lo medido en cada base de dato para las variables de salidas. La misma se determina de la siguiente manera:

$$a\mathrm{RRMSE}(\mathrm{h};\mathrm{D_{test}}) = \frac{1}{q}\sum_{j=1}^{q}\sqrt{\frac{\sum_{(x,y_j)\in D_{test}}(h(x_j)-(y_j))^2)}{\sum_{(x,y_j)\in D_{test}}(\bar{\mathrm{Y}}_\mathrm{j}-\mathrm{y_j})^2}}$$

3.3 Base de datos

Estas bases de datos se relacionan en la tabla 1.

Descripción de las bases de datos

EDM (Electrical Discharge Machining) es una base de datos que representa un problema de regresión con dos variables de salida y 16 variables de entrada. Cada variable de salida puede tomar 3 valores distintos (-1, 0,1). La tarea es acortar el tiempo de mecanizado mediante la reproducción del comportamiento de un operador humano que controla los valores de dos variables.

SF1, SF2 (Solar Flare) es una base de datos que contiene 3 variables de salida correspondientes a 3 tipos de llamas solares (común, moderada y severa) que

son observadas durante 24 horas. Hay dos versiones de esta base de datos, SF1 contiene los datos del año 1969 y SF2 del año 1978.

WQ (*Water Quality*) es una base de datos que contiene 14 variables de salida que representan las especies de plantas y animales en los ríos de Eslovenia y 16 variables de entrada que se refieren a los parámetros físicos y químicos que miden la calidad del agua.

OES97, OES10 (Occupational Employment Survey) son bases de datos obtenida en los años 1997 (OES97) y 2010(OES10) del Occupational Employment Survey compilada por US Bareu of Labor Statistics. Cada instancia (ciudad) provee un estimado del tiempo completo equivalente a la cantidad de empleados entre un número determinado de empleos para un área metropolitana específica. Existen 334 y 403 ciudades (instancias) en la base de datos de 1997 y mayo de 2010 respectivamente. Las variables de entrada en estas bases de datos son una secuencia aleatoria de un conjunto de empleos (ejemplo: doctor, dentista, mecánico) observados en al menos el 50% de las ciudades (algunas de las categorías no tiene valor para determinadas ciudades). Las variables de salida en los dos años son aleatorias, seleccionadas de todo el conjunto de categorías alrededor del 50% del umbral. Los valores perdidos tanto en las variables de entrada como en las de salida son reemplazados por la media de la muestra.

SCM1d, SCM20d (*The Supply Chain Management)* los conjuntos de datos de gestión de la cadena de suministro se derivan de la Competencia Agente Operador en la cadena de suministro (SCM TAC) del torneo a partir de 2010. Los métodos precisos para pre-procesamiento de datos. Algunos valores de referencia para la precisión de la predicción en este dominio están disponibles en la Predicción Desafío TAC SMC. Estos datos corresponden sólo al "Producto Futuro" tipo de predicción. Cada fila corresponde a un día de observación en el torneo (hay 220 días en cada juego y 18 partidos del torneo en un torneo). Las variables de entrada en este ámbito son los valores de un día de torneo

específico. Además, se incluyen 4 observaciones temporizadas para cada producto observado y el componente (1, 2,4 y 8 días de retardado) para facilitar cierta anticipación de las tendencias en el futuro. Los conjuntos de datos contienen 16 objetivos, cada objetivo se corresponde con el valor medio del día siguiente (SCM1d) o el valor por 20 días en el futuro (SCM20d). Los días sin valores objetivos se excluyen de los conjuntos de datos (es decir, los días con etiquetas que están más allá del final del juego están excluidos).

RF1, RF2 (The river flow) los conjuntos de datos de flujo de los ríos se refieren a la predicción de la red fluvial fluye durante 48 horas en el futuro en lugares específicos. El conjunto de datos contiene los datos de las observaciones de flujo por hora de los 8 sitios en la red del río Mississippi en los Estados Unidos y se obtuvieron del Servicio Meteorológico Nacional de Estados Unidos. Cada fila incluye la observación más reciente para cada uno de los 8 sitios, así como observaciones en tiempo lag de 6, 12, 18, 24, 36, 48 y 60 horas en el pasado. En RF1, cada sitio contribuye 8 variables de atributos para facilitar la predicción. Hay un total de 64 variables más 8 variables objetivo. El conjunto de datos RF2 extiende los datos RF1 mediante la adición de información de pronóstico de precipitación para cada uno de los 8 sitios (lluvia esperada reportado como valores discretos: 0,0, 0,01, 0,25, 1,0 pulgadas). Para cada sitio de observación y de calibre, el pronóstico de precipitación para las ventanas de 6 horas hasta 48 horas en el futuro se añade (6, 12, 18, 24, 30, 36, 42, y 48 horas). Los dos conjuntos de datos tanto contienen más de 1 año de observaciones horarias (>9000 horas) recogidos a partir de septiembre de 2011 hasta septiembre de 2012. El período de formación es el primer 40 % de las observaciones, y el período de prueba es el resto. El dominio es un candidato natural para la regresión multi-objetivo porque hay relaciones físicas claras entre las lecturas en la red fluvial contigua.

ATP1d, ATP7d (The Airline Ticket Price) los conjuntos de datos se refiere a la predicción de los precios de los billetes de avión. Las filas son una secuencia de observaciones en tiempo-ordenado lo largo de varios días. Cada muestra en

esta base de datos representa un conjunto de observaciones de una fecha de observación y salida par específico. Las variables de entrada para cada muestra son valores que pueden ser útiles para la predicción de los precios de los billetes de avión para una fecha de salida específica. Las variables objetivo en estos conjuntos de datos están al día siguiente (ATP1d) precio o el precio mínimo observado en los próximos 7 días (ATP7d) para las preferencias de vuelo 6 de destino (cualquier línea aérea con cualquier número de paradas, cualquier línea aérea sin parar solamente, Delta Airlines, Continental Airlines, Airtrain Airlines y United Airlines). Las variables de entrada incluyen los siguientes tipos: el número de días entre la fecha de la observación y la fecha de salida (1 función), las variables booleanas para el día-de-la-semana de la fecha de observación (7 funciones), la enumeración completa de la siguiendo 4 valores: 1) el precio mínimo, el precio, y el número de citas de 2 media) todas las compañías aéreas y de cada aerolínea citando a más del 50% de los días de observación 3) para no dejar, de una sola parada, y de dos paradas vuelos, 4) para el día actual, el día anterior, y dos días anteriores. El resultado es un conjunto de características de 411 variables. La naturaleza de estos conjuntos de datos es heterogénea con una mezcla de varios tipos de variables que incluyen variables booleanas, los precios, y los recuentos.

3.4 Resultados

En la tabla 2 se muestra el aRRMSE obtenido al evaluar los algoritmos: Multi-Target Stacking (MTS) y sus variantes en las 12 bases de datos anteriormente descritas.

Dataset	MTS(kl)(K)	MTSInv(kl)(K)	MTSGau(kl)(K)
Edm	0.743(3)	0.735(3)	0.736(3)
sf1	0.989(35)	0.995(35)	0.992(35)
sf2	0.973(35)	0.976(35)	0.974(35)
oes10	0.484(5)	0.48(11)	0.479(5)
oes97	0.533(3)	0.531(9)	0.519(3)
Rf1	0.538(3)	0.509(3)	0.519(3)
Rf2	0.737(3)	0.74(5)	0.737(5)

atp1d	0.456(7)	0.498(11)	0.448(7)
atp7d	0.606(5)	0.841(9)	0.613(5)
Scm1d	0.743(9)	0.734(9)	0.736(9)
Scm20d	0.283(11)	0.283(11)	0.283(11)
Wq	0.981(17)	1.04(35)	0.969(31)

Tabla 2 Resultados del aRRMSE obtenidos al evaluar las 12 bases de datos para el algoritmo MTS y sus variantes.

En el análisis de la tabla se puede observar que hay una paridad aunque se resaltan los bajos valores de error predictivo en las bases de datos oes10, oes97, rf1, atp1d y scm20d; además que sale el algoritmo MTS en la variante de pesado gaussiano como el de más reiteraciones de menor error de predicción.

Una vez obtenida la tabla 2 se procede a aplicar el test de Friedman para conocer si existen diferencias entre las variantes implementadas del algoritmo MTS. Los resultados del test se muestran en la tabla 3 y en la figura 1.

Algoritmo	MTS(k)	MTSInv(k)	MTSGau(k)
Ranking	2.12	2.25	1.62

Tabla 3 Estadístico de Friedman (distribuidos según chi-cuadrado: 1.3509; p-value: 0.2797).

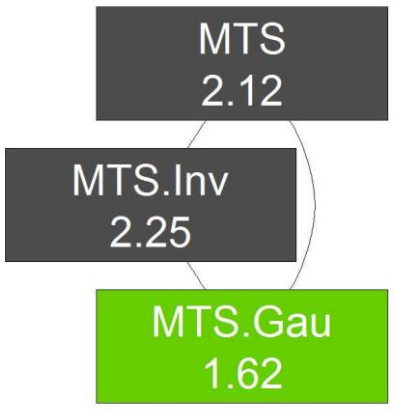

Figura 1 Test de Friedman.

Al aplicarse el test de Friedman para el MTS y sus variantes, los valores obtenidos difieren por tanto se rechaza la hipótesis nula y se procede a aplicar un test post-hoc. Se escoge el test de Nemenyi ya que este compara todos los algoritmos entre sí. Los resultados se encuentran en la figura 2.

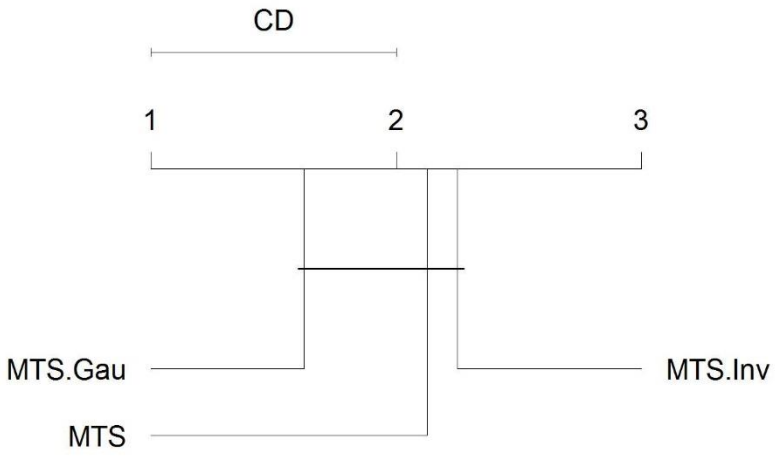

Figura 2 Test de Nemenyi.

En el análisis del test de Nemenyi se puede observar que entre los algoritmos no se aprecian estadísticamente diferencias significativas en cuanto a la precisión de la predicción.

3.5 Caso de Estudio

Para el desarrollo del caso de estudio se emplea como base de datos edm, una base de datos que tiene 154 observaciones para entrenamiento y prueba, 16 atributos de entrada y 2 de salidas.

Primeramente se dividen los datos en test y train.

```
prefix=char(dataset);
    nameTr=horzcat(prefix,'-5-');
    extendsTr='tra.arff';
    extendsTest='tst.arff';
    path=horzcat('data/',prefix, '/');
nfolds=5;
```

Luego se procede a leer el primer estado de modelos.

xTe

-4,71	0,05	0,65	0,26	5,6
-4,65	0,01	1,13	0	5,23
-4,65	0,01	1,13	0	5,15
-4,71	0,09	0,5	0,16	5,72
-4,62	0,02	1,06	0,11	5,41

yTe

0	1
0	1
0	-1
1	0
0	0

xTr

-4,81	0,1	0,37	0,36	5,82
-4,72	0,16	0,4	0,25	5,82
-4,83	0,04	0,33	0,16	5,86
-4,96	0,02	0,04	0,02	6,1
-4,51	0,29	0,74	0,28	5,53

yTr

0	0
0	1
0	0
0	1
1	0

Para ejecutar la predicción se obtiene el vector de variables predictoras de la instancia a predecir, se centraliza por la media correspondiente del conjunto de entrenamiento y finalmente se obtiene un vector con los resultados de la predicción.

```
function [yhat_non,yhat_inv,yhat_gau,yhat_lin] =
knnregresionkernelMTS(w,xTr,yTr,xTe,yTe,Kg,flag,n)

[Iknn, dist] = kNearstNeigbours (xTr, xTr, Kg, w);

yhat_non= predictionMTS (yTr,yTe,xTr,xTe,iknn,dist,'non',flag,n,Kg,w);
yhat_gau= predictionMTS (yTr,yTe,xTr,xTe,iknn,dist,'gau',flag,n,Kg,w);
yhat_inv= predictionMTS (yTr,yTe,xTr,xTe,iknn,dist,'inv',flag,n,Kg,w);

function y= PredictionMTS
(yTr,yTe,xTr,xTe,iknn,dist,kernel,flag,n,Kg,w)

    yhat= prediction (yTr,yTr,iknn,dist,kernel,true);
    y=[];
    for i = 1:n
        x= [];
        y1=yTr;
        y1 (i, :)= [];
        x= [xTr',y1']';
        [iknnext,distext] = kNearstNeigbours (xTr, xTe, Kg, w);
        yq = prediction
(yTr(i,:),yTe(i,:),iknnext,distext,kernel,flag);
        y = [y',yq']';

    end

end
```

Se calcula el aRRMSE y se muestran los resultados.

```
Function [err, det_err]=Computed_error (yhat, yTe)
        sden=sum(bsxfun(@minus,yTe',mean(yTe')).^2);
        If min(sden)==0
            err=-1;
        else
        det_err=sqrt(sum((yhat'-
yTe').^2)./(sum(bsxfun(@minus,yTe',mean(yTe')).^2)));
        err =mean(det_err);
        end
    end

Function writeToFile (file, conts)
    out= fopen(file,'a');
    fprintf(out, '%s', conts);
    fclose(out);
end
```

3.6 Conclusiones del capítulo

Al concluir este capítulo se puede decir que el algoritmo MTS en la variante de pesado gaussiano fue el de mejores resultados en cuanto al error de predicción según el test de Friedman. También el análisis del test de Nemenyi demostró que no existen diferencias significativas entre el algoritmo MTS y sus variantes.

CONCLUSIONES

Debido al estudio realizado en la presenta investigación se puede llegar a las siguientes conclusiones:

- ✓ De la documentación teórica analizada se identifica al algoritmo MTS como uno de los de mejores resultados en cuanto al error predictivo, el cual se implementó con el regresor de base KNN que no había sido implementado anteriormente.

- ✓ Se identificó la prueba no paramétrica de Friedman como las más utilizadas en el diseño de experimentos para validar algoritmos de aprendizaje automático mediante la cual la variante de pesado gaussiano salió como el de mejores resultados en cuanto al error de predicción.

- ✓ Además el análisis del test de Nemenyi demostró que no existen diferencias significativas entre las variantes implementadas.

Debido a lo antes planteado queda demostrado el cumplimiento del objetivo trazado en la presente investigación.

RECOMENDACIONES

Al concluir la presente investigación se proponen, a manera de recomendaciones, una serie de tareas para ampliar y dar continuidad al trabajo realizado:

- ✓ Experimentar la combinación de otros regresores de base existentes para el algoritmo MTS.

- ✓ Llevar todos los algoritmos de problemas de predicción con salidas compuestas a una sola plataforma, ya que están divididos, para tener un paquete completo disponible para su utilización.

- ✓ Realizar estudios sobre la evolución del algoritmo MTS e informarse sobre nuevos descubrimientos de algoritmos de este tipo que puedan surgir.

Bibliografía

1. Bakir, G. H., Hofmann, T., Schölkopf, B., Smola, A. J., Taskar, B., & Vishwanathan, S. V. N. (2007). *Predicting Structured Data (Neural Information Processing)*. The MIT Press.
2. Borchani, H., Varando, G., Bielza, C., & Larrañaga, P. (2015). A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *5*(5), 216–233.
3. Breiman, L., & Friedman, J. H. (1997). Predicting multivariate responses in multiple linear regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *59*(1), 3–54.
4. D Kocev, P. G. S. D. M. D. W. G. R. N. (2009). Using single- and multi-target regression trees and ensembles to model a compound index of vegetation condition. *Ecological Modelling 220*.
5. Džeroski, S., Demšar, D., & Grbović, J. (2000). Predicting chemical parameters of river water quality from bioindicator data. *Applied Intelligence*, *13*(1), 7–17.
6. Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning* (Vol. 1). Springer series in statistics Springer, Berlin.
7. González Diez, H. R., Santos, G., Campos, F., & Morell Pérez, C. (2016, July 13). Evaluación del algoritmo KNN-SP para problemas de predicción con salidas compuestas. *Revista Cubana de Ciencias Informáticas*, *10*(3), 119–129.

8. Izenman, A. J. (1975). Reduced-rank regression for the multivariate linear model. *Journal of Multivariate Analysis*, *5*(2), 248–264.
9. Pugelj, M., & Džeroski, S. (2011). Predicting structured outputs k-nearest neighbours method. In *Discovery Science* (pp. 262–276). Springer.
10. Spyromitros-Xioufis, E., Tsoumakas, G., William, G., & Vlahavas, I. (2014). Drawing Parallels between Multi-Label Classification and Multi-Target Regression. *arXiv Preprint arXiv:1211.6581v2*.
11. Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., & Vlahavas, I. (2011). Mulan: A java library for multi-label learning. *The Journal of Machine Learning Research*, *12*, 2411–2414.
12. Tsoumakas, G., Spyromitros-Xioufis, E., Vrekou, A., & Vlahavas, I. (2014). Multi-Target Regression via Random Linear Target Combinations. *arXiv Preprint arXiv:1404.5065*.
13. Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*(2), 241–259.
14. I. H. Witten and E. Frank, Data Mining: Practical machine learning tools techniques. Morgan Kaufmann, 2005.
15. T. Li, C. Zhang, and S. Zhu, “Empirical studies on multi-label classification.,” in IcTAI, vol. 6, pp. 86–92,2006.
16. R. Statistical Package, “R: A language and environment for statistical computing,” Vienna, Austria: R Foundation for Statistical Computing, 2009.
17. M. U. Guide, “The mathworks,” Inc., Natick, MA, vol. 5, p. 333, 1998.
18. I. H. Witten and E. Frank, “Weka,” Machine Learning Algorithms in Java, pp. 265–320, 2000.
19. J. Struyf, B. Zenko, H. Blockeel, C. Vens, and S. Dzeroski, “Clus: User’s manual,” 2010.

20. E. Spyromitros-Xioufis, G. Tsoumakas, W. Groves, and I. Vlahavas, “Multi-label classification methods for multi-target regression,” arXiv preprint arXiv: 12A.

21. A. J. Izenman, “Reduced-rank regression for the multivariate linear model,” Journal of multivariate analysis, vol. 5, no. 2,1975.

22. L. Breiman and J. H. Friedman, “Predicting multivariate responses in multiple linear regression,” Journal of the Royal Statistical Society.

23. L. Breiman, “Bagging predictors,” Machine learning, vol. 24, no. 2, 1996.

24. L. Breiman, “Random forests,” Machine learning, vol. 45, no. 1, pp. 5–32, 2M.

25. Pugelj and S. Džeroski, “Predicting structured outputs k-nearest neighbours method,” in Discovery Science, pp. 262–276, Springer, 2011.001ethodology), vol. 59, no. 1, pp. 3–54, 1997.

26. E. Bauer and R. Kohavi, “An empirical comparison of voting classification algorithms: Bagging, boosting, and variants,” Machine learning, vol. 36, no. 1-2, pp. 105–139, 1999.

27. B. Durrant, E. FRANK, L. HUNT, G. HOLMES, M. MAYO, B. PFAHRINGER, T. SMITH, and I. WITTEN, “Weka 3: Data mining software in java,” 2013.

28. G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, “Mulan: A java library for multi-label learning,” The Journal of Machine Learning Research, vol. 12, pp. 2411–2414, 2011.

29. J. Schneider, “Cross validation,” A Locally Weighted Learning Tutorial Using Vizier, vol. 1, 1997.

30. F. Joanneum, “Cross-validation explained,” 2005.

31. A. W. Moore, “Cross-validation for detecting and preventing overfitting,” School of Computer Science Carneigie Mellon University, 2001.

32. J. Demšar, “Statistical comparisons of classifiers over multiple data sets,” The Journal of Machine Learning Research, vol. 7, pp. 1–30, 2006.

33. M. Friedman, “The use of ranks to avoid the assumption of normality implicit in the analysis of variance,” Journal of the American Statistical Association, vol. 32, no. 200, pp. 675–701, 1937.

34. M. Friedman, “A comparison of alternative tests of significance for the problem of m rankings,” The Annals of Mathematical Statistics, vol. 11, no. 1, pp. 86–92, 1940.

35. J. Zar, “Biostatistical analysis. The university of chicago press upper saddle river,” New Jersey, 1999.

36. D. J. Sheskin, Handbook of parametric and nonparametric statistical procedures. Crc Press, 2003.

37. R. L. Iman and J. M. Davenport, “Approximations of the critical region of the fbietkan statistic,” Communications in Statistics-Theory and Methods, vol. 9, no. 6, pp. 571–595, 1980.

38. R. A. Fisher, “Statistical methods and scientific inference.,” 1956.

39. P. Nemenyi, “Distribution-free multiple comparisons,” in Biometrics, vol. 18, p. 263, INTERNATIONAL BIOMETRIC SOC 1441 I ST, NW, SUITE 700, WASHINGTON, DC 20005-2210, 1962.

40. J. W. Tukey, “Comparing individual means in the analysis of variance,” Biometrics, pp. 99–114, 1949.

41. O. J. Dunn, “Multiple comparisons among means,” Journal of the American Statistical Association, vol. 56, no. 293, pp. 52–64, 1961.

42. S. Holm, “A simple sequentially rejective multiple test procedure,” Scandinavian journal of statistics, pp. 65–70, 1979.

43. Y. Hochberg, “A sharper bonferroni procedure for multiple tests of significance,” Biometrika, vol. 75, no. 4, pp. 800–802, 1988.

44. A. Asuncion and D. Newman, “Uci machine learning repository,” 2007.

45. S. Džeroski, D. Demšar, and J. Grbovic, “Predicting chemical parameters of river water quality from bioindi-cator data,” Applied Intelligence, vol. 13, no. 1, pp. 7–17, 2000.

46. W. Groves and M. Gini, “Improving prediction in tac scm by integrating multivariate and temporal aspects via pls regression,” in Agent-Mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets, pp. 28–43, Springer, 2013.

47. D. Pardoe and P. Stone, “The 2007 tac scm prediction challenge,” in Agent-Mediated Electronic Commerce and Trading Agent Design and Analysis, pp. 175–189, Springer, 2010.

48. W. Groves and M. Gini, “A regression model for predicting optimal purchase timing for airline tickets,” tech.rep., Technical Report 11-025, University of Minnesota, Minneapolis, MN, 2011.

49. COELLO, S. G., ROLANDO ALFREDO HERNÁNDEZ LEÓN El proceso de investigación científica. Editedby E. UNIVERSITARIA. Edtion ed. Ciudad de La Habana, 2011. ISBN 978-959-16-1307-3.

50. FUENTES, L. R. Desarrollo de paquete de Arquitecturas de Software para Sistemas de Información Geográfica de escritorio. UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS, 2011.

51. GAMMA, E., RICHARD HELM, RALPH JOHNSON, JOHN VLISSIDES Design Patterns: Elements of Reusable Object-Oriented Software. Edited by A.-W.P.C. SERIES. Edtion ed., 1994.

52. GONZÁLEZ, D. G., LILIAN GARCÍA CASTRO. Aplicación informática para caracterizar las ontologías representativas del conocimiento en la Web. Ingeniero en Ciencias Informáticas Universidad de las Ciencias Informáticas, 2011.

53. PRESSMAN, R. Ingeniería del Software. Un enfoque práctico. Editedby M. HILL. Edtion ed. España, 2001. 640 p. ISBN 9788448132149.

54. PRESSMAN, R. Ingeniería del Software: Un Enfoque Práctico. Editedby M.G. HILL. Edtion ed., 2005. ISBN 9701054733