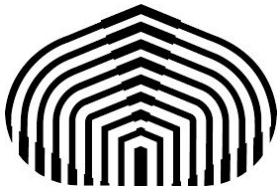


Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
Inteligencia Artificial II

Integrantes:	Erick Silva	11-10906
	Leslie Rodrigues	10-10613
	Georvic Tur	12-11402

Proyecto I: Regresión Lineal Múltiple

Resumen

En primer lugar, se implementó una versión de la regresión lineal múltiple usando el algoritmo de descenso del gradiente para minimizar la función de error cuadrático. Los resultados de dicho algoritmo se puede controlar a través de la tasa de aprendizaje, los datos de entrenamiento y el número máximo de iteraciones. Los coeficientes iniciales son números aleatorios entre -0.3 y 0.3.

Para evaluar dicha implementación se entrenó dicho algoritmo con dos *datasets* de prueba y se generaron gráficas de convergencia con distintos valores de los parámetros para cada *dataset*.

Finalmente, se usó dicha implementación en un *dataset* de ventas inmobiliarias de veinte dimensiones y 1361 instancias. Este *dataset* fue obtenido a partir de un trabajo de limpieza, preprocesamiento y selección aplicado a un *dataset* de 82 atributos y 2930 instancias. Se generó de manera aleatoria un conjunto de entrenamiento, con el 80% de las instancias, y otro de validación, con el 20% de las instancias, ambos disjuntos. Luego se evaluó el modelo usando gráficas de convergencia para distintos valores de los parámetros y las métricas propuestas por [1] en su sección cuarta.

Detalles de Implementación y Experimentación

La implementación del algoritmo de regresión lineal múltiple se hizo con el lenguaje de programación *Python*, usando la librería *NumPy* para trabajar con arreglos de cantidades decimales y disponer de funciones del álgebra lineal, así como la librería *Matplotlib* para construir los gráficos mostrados a continuación. Para instalar dichas librerías se ha provisto un archivo *requirements.txt* a ser usado con el manejador de paquetes *pip*.

La implementación se encuentra dividida en varios módulos con sus respectivas funciones:

- *proyecto.py*
    - *regresion_lineal_multiple:* esta es la función responsable de generar el modelo. Usa una función auxiliar que lo evalúa en un punto y otra que calcula el error cuadrático. Al final devuelve una lista de coeficientes por cada iteración y una lista de sus sendos errores. Se detiene usando como criterios un valor umbral para el error de 10^{-6} y un valor máximo para las iteraciones que por defecto es 1000.
    - *normalizarZ:* esta función es la responsable de normalizar las columnas de la matriz de instancias usando la desviación estándar y la media.
    - evaluar_modelo: esta función genera los cuatro estadísticos propuestos por [1].
    - *error_n*: calcula el error del modelo dados sus coeficientes e instancias.
    - *h:* dados los coeficientes y una instancia del dominio, evalúa el modelo en ella.
- *parseInput.py*: el objetivo de este módulo es limpiar, procesar y filtrar los datos contenidos en el *dataset* de ventas inmobiliarias. Usa a *customValues.txt* para convertir los datos nominales a numéricos.
- *customValues.txt:* este archivo de texto contiene las especificaciones para convertir cada valor de tipo nominal a uno de tipo numérico.

Los módulos de evaluación y experimentación se encuentran divididos de la siguiente manera:

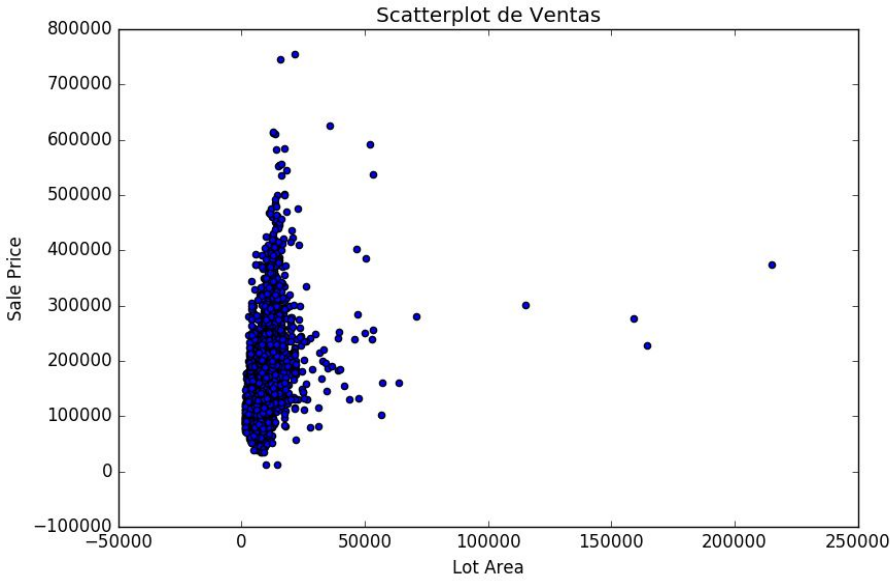
- *pregunta_dos_parte_a.py*
- *pregunta_dos_parte_b.py*
- *pregunta_dos_crimen_parte_a.py*
- *pregunta_dos_crimen_parte_b.py*
- *pregunta_tres.py*

En cuanto a la preparación de los datos, hay que destacar que debido al tamaño pequeño de los *datasets* [3] y [4], su preprocesamiento sólo requirió del uso de un editor de texto y expresiones regulares para eliminar comentarios y espacios en blanco. Todos los *datasets* fueron normalizados con la función mencionada arriba justo antes de entrenar cada modelo. En el caso de [5], fue necesario elaborar un *script* que llamamos *parseInput.py*. Por tanto, el preprocesamiento de [5] consistió en la siguiente secuencia de pasos:

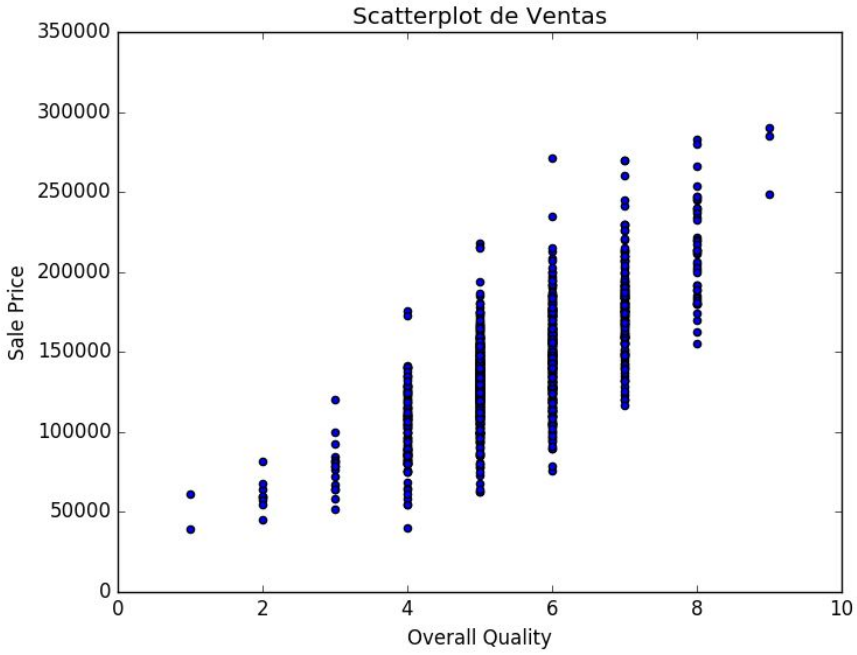
1. Se filtraron las instancias de acuerdo a los criterios de *De Cock* en la sección cuarta de [1].
2. Los valores de tipo nominal fueron sustituidos por valores de tipo numérico. Al hacer esto, se tomó en cuenta que algunas de las dimensiones tienen valores con orden, tal como lo explica [2]. Por esta razón, se asignan valores enteros (pero de tipo *float* en *Python*) centrados en cero.
3. Se seleccionan las dimensiones a usar en la regresión lineal a partir de las 81 dimensiones originales. El criterio usado para seleccionar las dimensiones consistió en hacer gráficas de dispersión de la dimensión *saleprice* contra todos los demás atributos y se seleccionaron aquellos que permitían dividir mejor los precios al hacer una

inspección simple de la gráfica. Un ejemplo de esto se pueden ver en la *gráfica 2*, donde los precios se dividen mejor que en la *gráfica 1*. En la primera se observa que los valores de la dimensión *Lot Area* no permiten discriminar bien a los precios de venta. En cambio, en la segunda los valores de *Overall Quality* sí permiten discriminar en mayor medida a los precios.

4. Se reemplazaron todos los valores faltantes por el promedio de su dimensión. Hay que destacar que según [2] los datos escritos como *NA* no representan datos faltantes, sino ausencia de una característica como podría ser una cerca o calefacción. Los datos faltantes simplemente no aparecen con ningún valor.
5. Se dividieron las instancias aleatoriamente en dos conjuntos disjuntos. El primero, con 80% de las instancias reservado para el entrenamiento y el segundo, con 20% de las instancias, reservado para la validación.



Gráfica 1



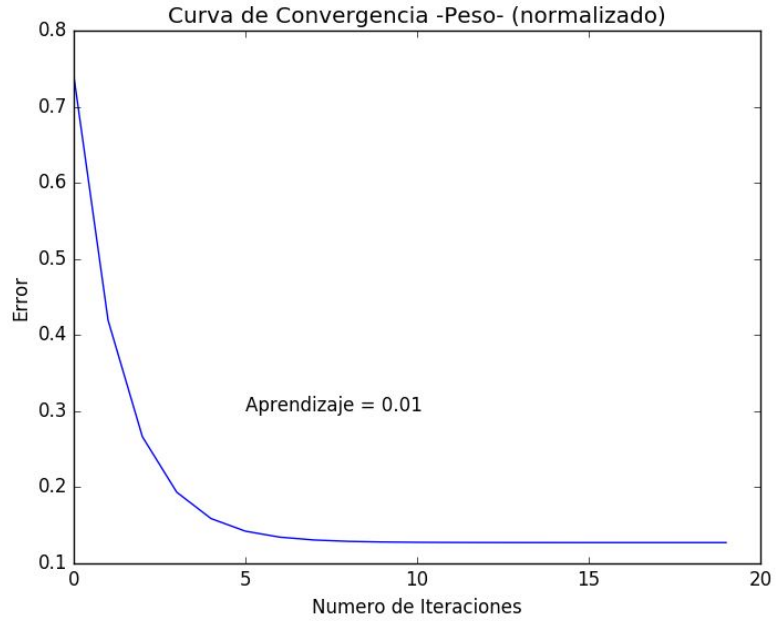
Gráfica 2

Presentación y Discusión de los Resultados

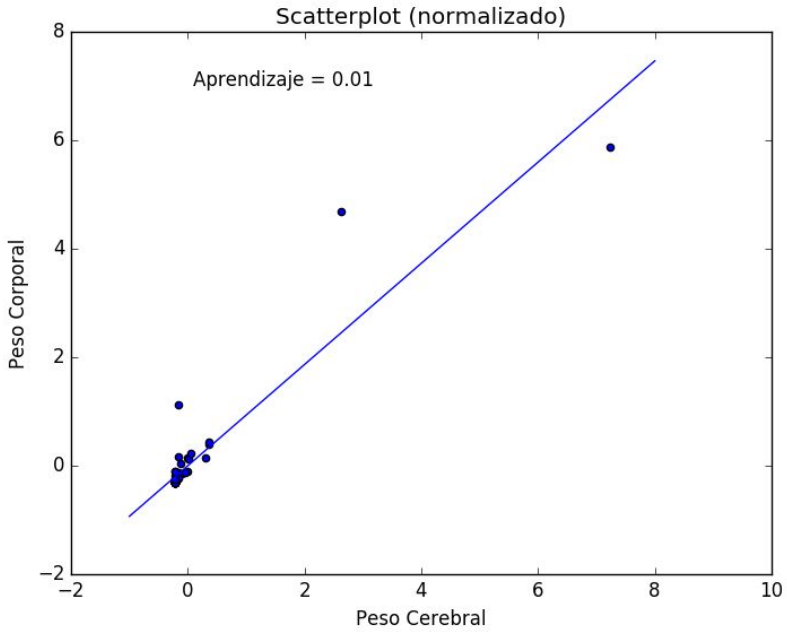
Peso corporal a partir del peso cerebral

Para esta sección se usó el *dataset* [3]. El objetivo es predecir el peso corporal de un individuo a partir de su peso cerebral. Para entrenar el modelo, se usaron 62 instancias con las dimensiones mencionadas. Luego de haber entrenado el modelo hasta lograr un error menor a 10^{-6}, se generó la *gráfica 3* y la *gráfica 4* usando valores iniciales aleatorios, entre -0.3 y 0.3, para los coeficientes y con un aprendizaje que se indica en ellas.

Entre los aspectos interesantes a comentar está la pequeña tasa de aprendizaje usada. El uso de tasas mayores no aseguraban la convergencia del modelo. En este caso, la convergencia se puede evidenciar debido al carácter decreciente de la función de error cuadrático medio por cada iteración en la *gráfica 3*. Otro de los aspectos a resaltar es la influencia de los datos atípicos (por inspección simple) tal como se puede ver en la *gráfica 4.*



Gráfica 3



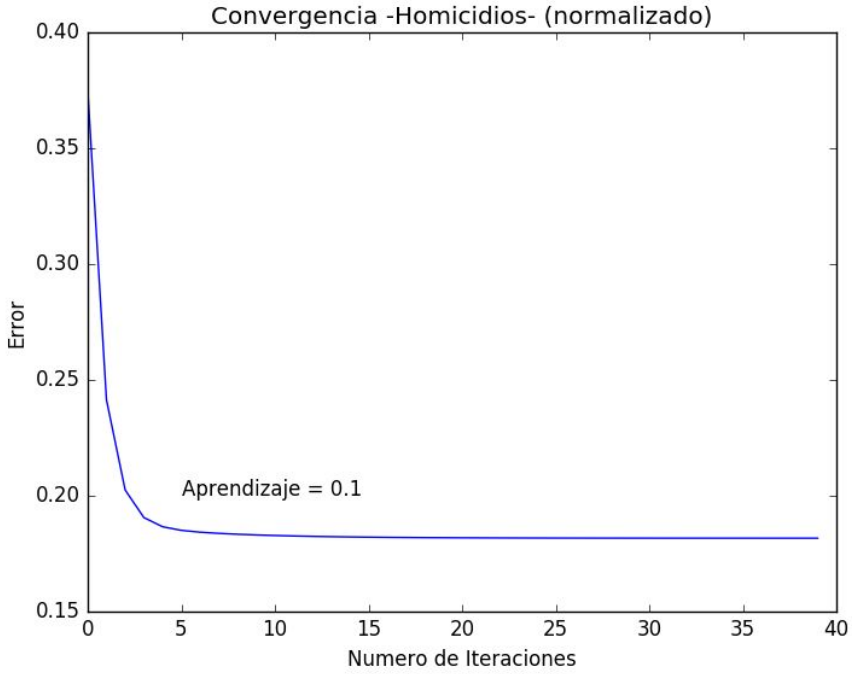
Gráfica 4

Número anual de Homicidios por cada millón de habitantes

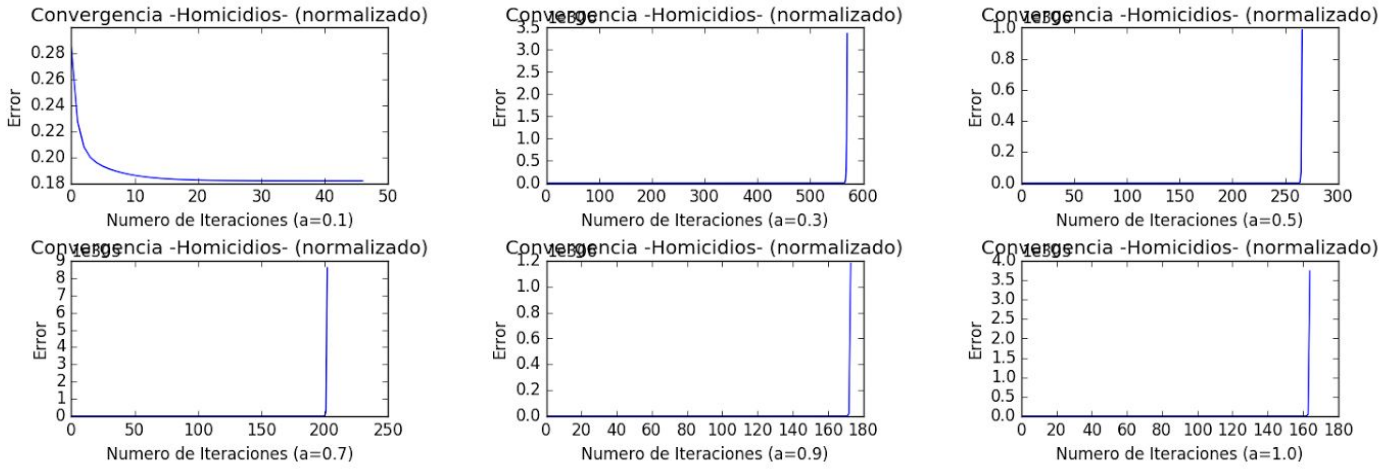
En esta sección se usó el *dataset* [4]. En él se tienen veinte instancias con cuatro dimensiones, siendo así nuestro primer caso de regresión lineal múltiple. El objetivo es predecir el número de homicidios a partir del número de habitantes, porcentaje de la población con ingresos familiares menores a $5000 y porcentaje de la población desempleada. Se entrenó el

modelo hasta lograr un error menor a 10^{-6} o hasta pasar de 1000 iteraciones usando diversos valores del aprendizaje, tal como se muestra en las gráficas cinco y seis.

Al igual que en el caso anterior, si se usaban tasas de aprendizaje más grandes entonces no se podía asegurar la convergencia obtenida en la *gráfica 5.* Esto se puede evidenciar en la *gráfica 6* donde se compara la evolución de la función de error usando seis tasas de aprendizaje. Debido a tasas de aprendizaje muy alta, varias de ellas divergen. Se puede ver que todas salvo por 0.1 van a infinito.



Gráfica 5



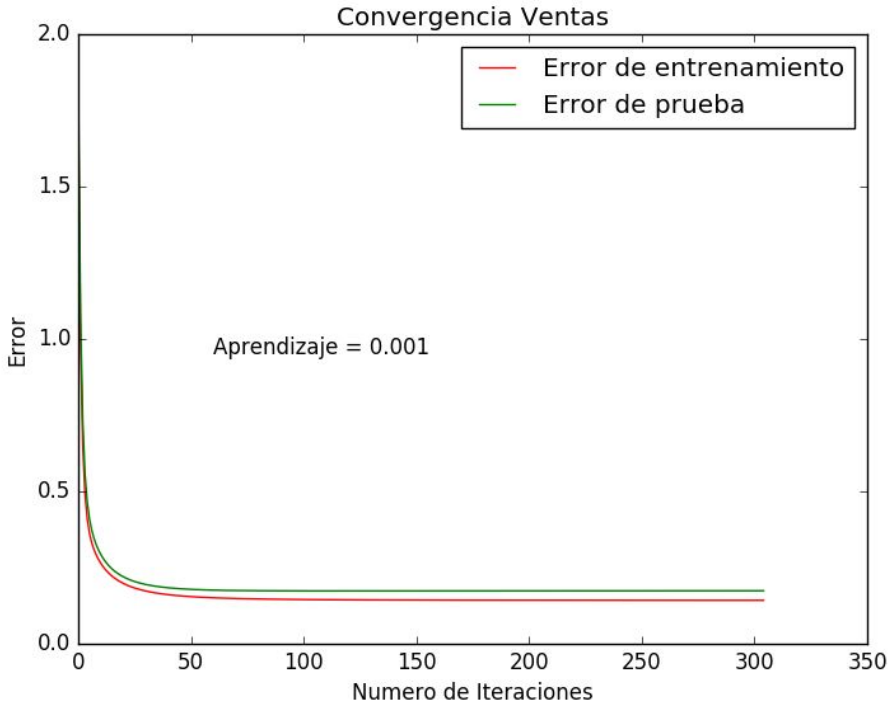
Gráfica 6

Precio de venta de inmuebles

En esta sección se usó el *dataset* [5] que trata de la venta de inmuebles en Ames, Iowa en el periodo de 2006 a 2010. Se encuentran 82 atributos para cada inmueble, principalmente relacionados con el estado actual del inmueble, de estos 82 atributos, hay 23 nominales, 23 ordinales, 14 discretos y 20 continuos, además de 2 que sirven de identificadores. Además se dan en total 2930 instancias.

Después de aplicar los filtros descritos anteriormente se quedaron con 1361 instancias. De las instancias restantes se seleccionó un 20% aleatoriamente como conjunto de validación y se usó el otro 80% para entrenar el clasificador.

A continuación las gráficas de error tanto de validación como entrenamiento:



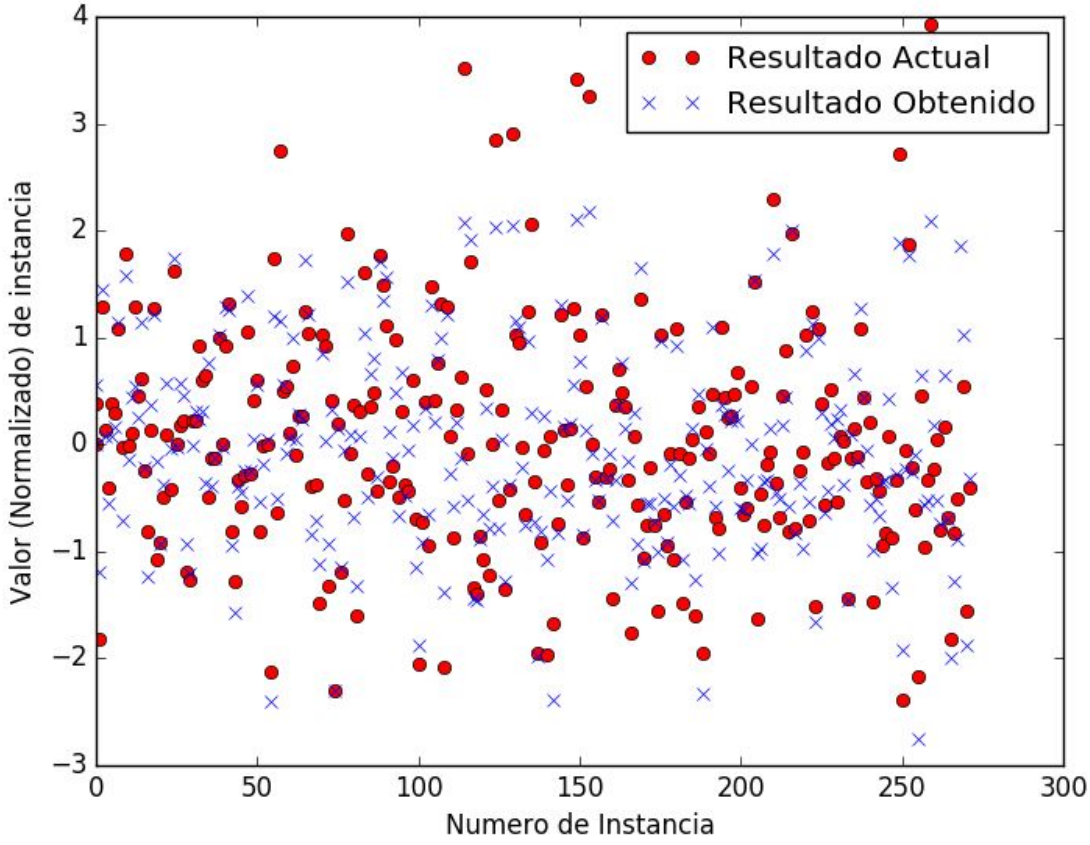
Gráfica 7

En la *gráfica 7* podemos ver que el error del conjunto de prueba decrece pero no tanto como el error de entrenamiento.

Además, se nos pidió que calculáramos 4 métricas para evaluar nuestros resultados:

- La desviación estándar máxima: 1.3877708460054223
- El *bias*: -1.7134829180806731e-05
- La desviación absoluta promedio: 0.28057647019330745
- El error cuadrático medio: 0.14412146285456193

Mientras estos valores pueden parecer bajos, debemos recordar que los precios con que esto se calculo se encuentran normalizados, por lo que un valor muy bajo aquí todavía es alto. Por último, hagamos una comparación visual de los resultados de validación en la *gráfica 8*.



Gráfica 8

En esta podemos ver que aunque algunos resultados a veces son acertados, no son exactos en una gran cantidad de casos y fallan cuando el valor es muy grande, es decir, se tiende a subestimar los valores reales como habíamos visto en el *bias*.

Conclusiones

En el caso de los *datasets* [3] y [4] nos abstenemos de ofrecer alguna conclusión sobre el modelo obtenido por la poca cantidad de instancias. En este sentido, como trabajo futuro se podrían usar cantidades superiores de las mismas. Sin embargo, se pudo demostrar la importancia de poder variar el coeficiente de aprendizaje para asegurar la convergencia del modelo, tal como se aprecia en la *gráfica 6*.

Por su parte, del *dataset* [5] podemos concluir, en primer lugar, que es suficiente con 20 atributos para tener una buena aproximación del precio de venta. Como se explicó anteriormente, dichos atributos fueron seleccionados en base a su capacidad de discriminar el precio de venta bajo una inspección simple de sus gráficas de dispersión con este atributo. La calidad de este modelo se puede evidenciar en la *gráfica 7,* donde se muestran las curvas de convergencia para el entrenamiento y la validación. Como trabajo futuro, se podrían comparar con éste los modelos generados con los demás atributos. En segundo lugar, como se puede apreciar en la *gráfica 8*, nuestro modelo tiende a subestimar los precios en el conjunto de validación. Esto significa que éste es adecuado si se es un usuario de baja tolerancia al riesgo.

Referencias

[1] De Cock, Dean. 2011. "Ames, Iowa: Alternative To The Boston Housing Data As An End Of Semester Regression Project". *Journal Of Statistics Education* 19 (3). https://ww2.amstat.org/publications/jse/v19n3/decock.pdf.

[2] De Cock, Dean. 2017. "Ameshousing.Txt". *Ww2.Amstat.Org*. https://ww2.amstat.org/publications/jse/v19n3/decock/DataDocumentation.txt.

[3] http://people.sc.fsu.edu/~jburkardt/datasets/regression/x01.txt

[4] http://people.sc.fsu.edu/~jburkardt/datasets/regression/x08.txt

[5] http://www.amstat.org/publications/jse/v19n3/decock/AmesHousing.txt