Práctica 1

Luis Fernando Lomelín Ibarra
a01177015@itesm.mx
Tecnológico de Monterrey
Ingeniería en Tecnologías Computacionales
Monterrey, N.L., México

David Alejandro Martinez Tristan
A01610267@itesm.mx
Tecnológico de Monterrey
Ingeniería en Tecnologías Computacionales
Monterrey, N.L., México

RESUMEN

El proposito de esta prática fue ver y trabajar con el método de predicción estadítica de Regreción lineal. Para lograr esto, se proporcionaron dos sets de datos, genero.txt y mtcars.txt, los cuales van a ser utilizados para analizar este modelo. Admás para esta práctica se utilizaron las herramientas estadísticas de R para visualisar y preparar datos y la librería de Scikit learn para programar un modelo de regreción linear en base de los sets de datos proporcionados.

ACM Reference Format:
Luis Fernando Lomelín Ibarra and David Alejandro Martinez Tristan. 2021. Práctica 1. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1. INTRODUCCIÓN

En lo que llevamos del curso hemos empezado primeramente viendo que es el aprendizaje automático. El aprendizaje automático, explicado de una forma breve, es cuando se alimentan datos a un algoritmo para generar reglas. Es decir, se busca entrenar a un algoritmo por medio de datos para que pueda, a base de este conocimiento generado a través de los datos, hacer predicciones y solucionar problemas.

El aprendizaje automático se puede dividir en tres tipos: supervisados, no supervisados, y de refuerzo. Los supervisados son aquellos que usan un set de datos con indicadores que les permiten saber y comparar sus resultados con los resultados de los datos. Los no supervisados son aquellos que no cuentan con indicadores en los datos, es decir, forman su conocimiento solo con los datos proporcionados sin tener un set de indicadores prehecho que le indique si está bien o mal. Finalmente está el aprendizaje de refuerzo, el cual utiliza estados, recompensas y penaltys para que el algoritmo aprenda por su propia cuenta.

También en el curso se vio el tema de minería de datos. La minería de datos es una de las formas en las que se exploran y se extraen conocimientos a través de gran cantidad de datos. La minería de datos y el aprendizaje automático se usan en conjunto para generar conocimiento de utilidad a base de una gran cantidad de datos.

Esta práctica se enfoca en la utilización de ambos, enfocados en la Regresión Lineal. La regresion lineal es una función de estadística que busca encontrar la relación entre una variable dependiente con una independiente. Es decir es un método estadístico que trata de adaptarse y predecir el comportamiento de una variable dado un set de datos. Cuando la regresión cuenta con solo un set de variables independientes se le llama una regreción simple, mientras cuando existen más de una variable se le llama una regreción multivariable.

En este trabajo aplicaremos la regreción lineal para los sets de genero.txt y mtcars.txt.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2. CONCEPTOS PREVIOS

Para llevar a cabo este trabajo utilizamos e investigamos de los siguientes conceptos y herramientas:

- Regreción lineal
- Algebra lineal
- Métodos de Validación
- Estadística
- Python
- R
- Scikit Learn
- Numpy y Cupy
- LaTeX

3. METODOLOGÍA

3.1. Dataset de Género

3.1.1. Visualización de los datos con R

. Antes de empezar con el código empezamos primero cargando y viendo los datos en R. Al principio nos topamos con un problema, el cual era que los datos estaban separados por comas y el comando de read no lo estaba aceptando directamente. Tras investigar la documentación del lenguaje descubrimos que se podía utilizar de la siguiente manera:

```
read.table("genero.txt", sep = ",", header=TRUE)
```

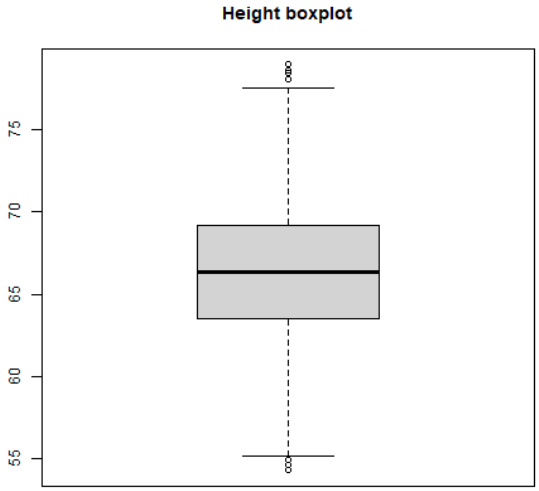
Con la opción de sep, pudimos ya sin problemas cargar los datos a R. Con los datos en la herramienta, ya se podrían explorar los datos con facilidad. La práctica indicaba que solo se iban a utilizar las columnas de *Height* y *Weight*, pero el dataset contaba con tres columnas. Para solo ver los datos relevantes se uso el siguiente comando:

```
subset(gen, select=c(2,3))
```

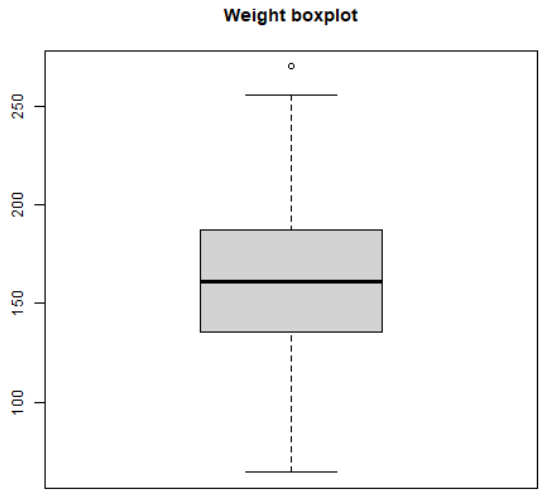
Este comando nos permitio solo tener la información de height y weight, la cual es de interes para la práctica. Con esta información cargada se genero la siguiente tabla:

Height	Weight
Min. :54.26	Min. : 64.7
1st Qu.:63.51	1st Qu.:135.8
Median :66.32	Median :161.2
Mean :66.37	Mean :161.4
3rd Qu.:69.17	3rd Qu.:187.2
Max. :79.00	Max. :270.0

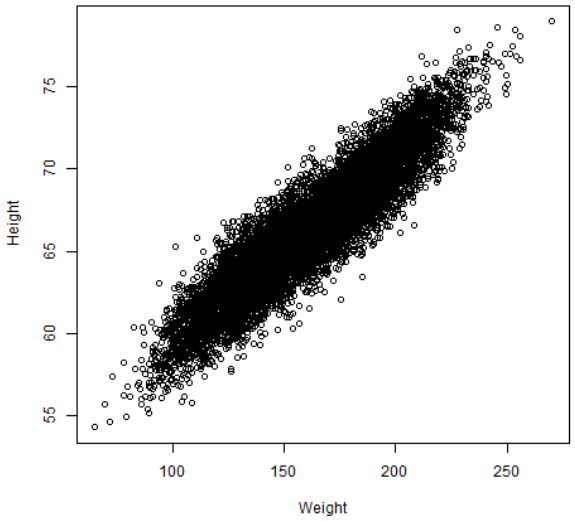
Ademas de esta información decidimos también expresarla en un boxplot para ver que tan dispersos se encuentran los datos ya que esto podría afectar al modelo de regresión.



En el caso de la variable height, los datos se encuentran masomenos bien distribuidos entre los valores de 65 y 70, pero si existen valores muy separados de este rango como son el máximo de 79 y el minimo de 54.26. Esto quiere decir que los valores de height se encuentran en su mayoría relativamente cerca, pero existen valores que drasticamente son diferentes a la mayoría.



Con la variable weight la mayoría de los datos se encuentran entre 135.8 y 187.2, con valores extremos como un minimo de 64.7 y un maximo de 270.0. Simliarmente al caso de la variable de height, los valores de weight se encuentran relativamente bien distribuidos en valores cercanos pero cuenta con valores extremos.



Visualizando los datos con una gráfica de disperción se puede apreciar con mayor detalle que si existe una relación entre los datos y que no se encuentran drasticamente dispersos que es posible utilizar sin problemas la regresión lineal.

Con esta información se puede concluir que si es posible aplicar regresión lineal sin mucho problema, ya que los datos estan masomenos bien distribuidos dentro de un rango razonable, pero existe la probabilidad de que pueda tener un poco de problemas ya que existen valores muy dispersos que puedan afectar a la linea.

Finalmente para facilitar el uso de estos datos en la programación con python se creo un archivo con solo las columnas requiridas y separadas con espacios en blanco.

3.1.2. *Regresión Lineal y Scikit Learn*

. Para poder poner en practica la regreción lineal con el dataset dado, se utilizo la librería de Scikit Learn el cual cuenta con muchas herramientas que permiten con facilidad crear y entrenar modelos de aprendizaje automático. Como nos vamos a enfocar solo en regreción linear, se utilizo la utilería de LinearRegresion.

Dado que el dataset de genero cuenta con 10000 elementos, se decidio utlizar el metodo de partir los datos en 80/20 para entrenar y validar al modelo. Utilizando el 80 % de los datos para entrenar al modelo y el 20 % para probar al modelo. Para lograr la partición de manera alateoría se utilizo la función de train_test_split proporcionada por scikitlearn:

```
x_train, x_test,y_train, y_test=train_test_split(X,y,
     test_size=0.2, train_size=0.8, random_state=1,
     shuffle=True)
```

Teniendo ya divido nuestros datos para entrenar y para probar empezamos entrenando al modelo de la siguiente manera:

```
reg = LinearRegression().fit(x_train,y_train)
```

Finalmente para probar que tan bien entrenado quedo el modelo utlizamos la función de Mean Squared Error para determinar que tan buenas fueron las predicciones. Esto lo logramos con el siguiente código:

```
#Get prediction
predictions = reg.predict(x_test)
#get MSE
mse = mean_squared_error(y_test, predictions)
```

3.1.3. *Gradient Decent*

. Finalmente la práctica nos pedía que consturyeramos nuestro propio gradiente decendiente y que compararamos su desempeño con el modelo de Scikit Learn. Para lograr esto nos basamos en el contenido visto en clase sobre la gradiente decendiente. Primero para reducir el tiempo que nos tardamos en hacer operaciones, decidimos usar la librería de cupy, la cual nos permite utilizar nuestro GPU para los calculos. Con esto en mente vectorizamos las funciones que utilizamos.

Para calucular la linea de regreción utilizamos la siguiente formula:

$$h(\vec{\beta}) = X \cdot \vec{\beta}$$

Nuestra función de costo quedo como se muestra:

$$J(\vec{\beta}) = \frac{1}{n} \sum_{i=1}^{n} (X \cdot \vec{\beta} - \vec{y})^2$$

Y la función gradiente como:

$$\nabla MSE(\vec{\beta}) = \frac{2}{n} X^T \cdot (X \cdot \vec{\beta} - \vec{y})$$

Con estas funciones definidas seguimos el siguiente algoritmo para calcular el gradiente decendiente.

Se declara como un vector de m features β

while no converga **do**

$$\beta = \beta - \alpha \cdot \nabla MSE(\beta)$$

end while

3.2. Dataset de mtcars

3.2.1. *Dataset*

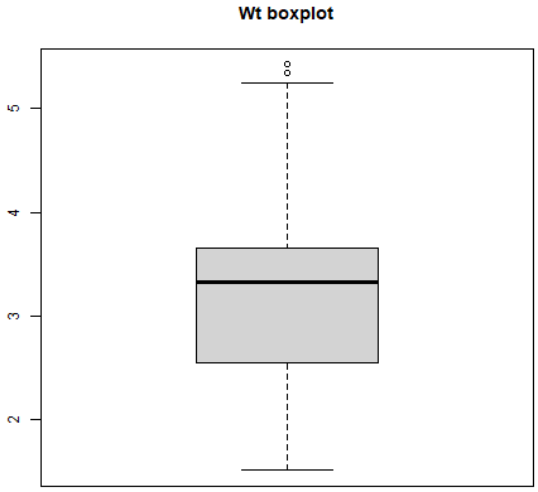
. El dataset MTCARS cuenta con 11 atributos de 33 diferentes vehículos. La intención es predecir los caballos de fuerza **hp** de un modelo desconocido a partir de su desplazamiento del motor **disp** y su peso **wt**.

3.2.2. *Visualización de los datos con R*

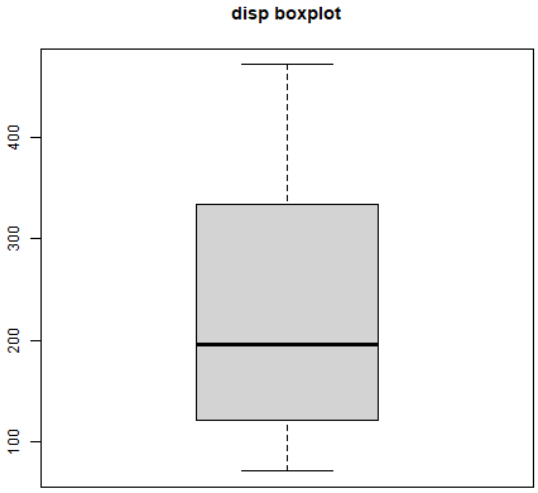
. De similar manera que con el dataset de genero, nos es de interes primero visualizar y ver los datos bien de MTCARS, para ello lo cargamos a la herramienta de R utlizando el mismo comando. Como este dataset esta separado con espcaios no fue necesario utlizar la opción de sep para cargar correctamente los datos.
Con los datos en la herramienta de manera similar selecionamos las columnas de interes a la práctica las cuales eran *wt, disp y hp.* El resumen de estos datos fue el siguiente:

disp"	hp	wt
Min. : 71.1	Min. : 52.0	Min. :1.513
1st Qu.:120.8	1st Qu.: 96.5	1st Qu.:2.581
Median :196.3	Median :123.0	Median :3.325
Mean :230.7	Mean :146.7	Mean :3.217
3rd Qu.:326.0	3rd Qu.:180.0	3rd Qu.:3.610
Max. :472.0	Max. :335.0	Max. :5.424

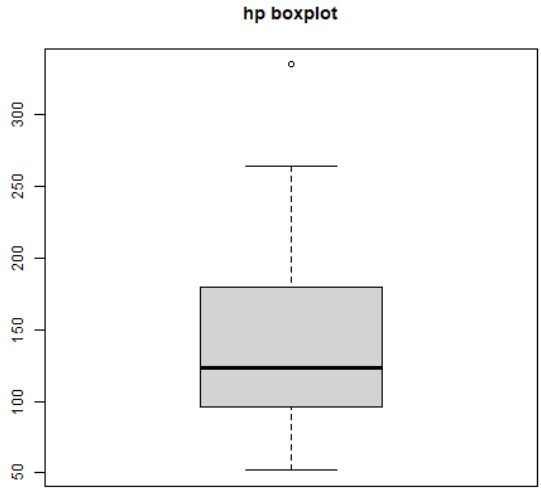
Similarmente analizamos los datos por medio de boxplots y una grafica de dispersión.



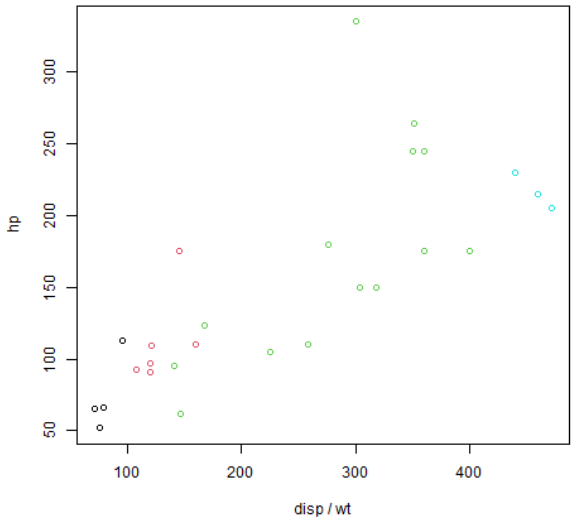
Con la variable wt podemos ver que la mayoría de datos se encuentran entre 2.581 y 3.25 con algunos datos extremos como son el máximo de 5.424 y el minimo de 1.513. Esto quiere decir que los datos de wt se encuentran medio agrupados, lo cual puede ser bueno para nuestro modelo.



En el caso de disp, la mayoría de los datos se encuentran entre 123 y 180 y cuenta con muy pocos valores extremos. De igualmanera pinta bastante bien para nuestro modelo, porque esto indica que al menos los datos de esta variable se encuentran relativamente cerca que se puede hacer una relación clara.



De manera simliar, la vairable hp cuenta con su mayoría de los datos entre 196.3 y 326.0. También cuenta con valores extremos como 472.0 pero no son muchos los casos.



Finalmente para poder ver una mejor relación en los datos en los datos se realizo una gráfica de dispersión. En esta se puede apreciar mejor la relación entre wt, disp y hp. Viendo como estan desplegados y distribuidos los datos se puede concluir que la regresión lineal talvez si sea una buena opción de modelo para estos datos.

3.2.3. *Regresión Lineal y Scikit Learn*

. Para realizar la regresión lineal se extrajeron únicamente las variables que serían consideradas para el modelo y se utilizaron funciones

pre-elaboradas de las librerías de Scikit Learn y NumPy.
Dado que se trató de un dataset pequeño, se aplicó la metodología *n-fold cross validation* para generar el training set y el data set, con el objetivo de generar diferentes modelos y probar la precisión global de las estimaciones:

```
kf = KFold(n_splits=len(X), shuffle=True)
kf.get_n_splits(X)

precision = 0

for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    reg = LinearRegression().fit(X_train, y_train)
    y_pred = reg.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    precision = precision + mse

precision = precision / len(X)
```

Para cada iteración se reportaron los MSEs parciales y por último el MSE final.

3.2.4. *Gradient Decent*

. Realmente se reutilizo la función de gradient decent para este dataset.

4. RESULTADOS

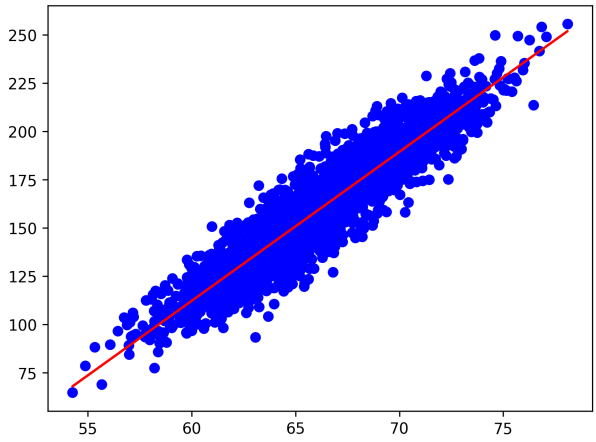
A continuación se presentan los resultados de generar la regresión lineal y aplicar ambos métodos de validación:

4.1. Dataset de gender

Después de construir y entrenar al modelo, se probo con el set de pruebas previamente hecho utilizando las funciones:

```
#Get prediction
predictions = reg.predict(x_test)
#get MSE
mse = mean_squared_error(y_test, predictions)
```

Y se obtubo un MSE de 141.9116147126515. Esto es relativamente bueno, considerando que los datos a pesar de estar medio agrupados si se encontraban bastante separados. Para ponerlo en contexto se genero la siguiente gráfica.



Con los mismos datos de entrenamiento se corrio el gradient decent que programamos y además utilizamos las mismas pruebas. El MSE que regreso fue de 115.1430429833293. Este valor es mucho mejor que el de Scikit learn pero cabe recalcar que el proceso para obtener el resultadp fue significativamente más lento. Por ello se requirio el uso de la librería de Cupy para acelerar los cálculos.

4.2. Dataset de mtcars

Dado que se aplicó la *n-fold validation*, se generaron 33 diferentes modelos con los subsets de train y test que se obtuvieron en cada iteración:

a	b	MSE
-0.4535676377946117	-1.111796390715245	2922.9020266948874
-0.4926750311636729	-1.1467656670110247	22812.26646542839
-0.4862324783788887	-1.1420765520660485	14623.099609693258
-0.4498467914292023	-1.106577073724613	6950.789737444822
-0.4436813227945677	-1.0988052129151304	17338.316757427758
-0.44870908781149627	-1.0060443788936522	11126.49207173443
-0.4535676377946117	-1.111796390715245	2922.9020266948874
-0.4703097565417811	-1.1314169409098154	3366.6246716316537
-0.44247249519745874	-1.096231907756508	20009.775130420523
-0.48330146120741885	-1.1394031819559267	11476.790250003203
-0.4372423254151276	-1.088548875747115	33399.091995825656
-0.4777005399408571	-1.135457880149593	6557.070279732207
-0.4480800743903211	-1.1035915129402945	9493.979800243382
-0.45017018628767336	-1.107434146384528	6513.279190631079
-0.4703097565417811	-1.1314169409098154	3366.6246716316537
-0.4891035349790453	-1.1444907261605393	18064.219406760192
-0.47165492510387264	-1.1327732086786129	4345.822631617231
-0.49318078402540483	-1.145551575842729	23444.61553047343
-0.4835848010639505	-1.1401743070858466	11784.3456358789
-0.4535676377946117	-1.111796390715245	2922.9020266948874
-0.4789570500666701	-1.1362207683850518	7537.910662667879
-0.4833908358087804	-1.1390476574723472	11550.190196125483
-0.47434894335833594	-1.1357460336806289	6691.623036960279
-0.4943322179295022	-1.1477854879741218	25201.292213765348
-0.43194861462335427	-1.081434710859735	50336.70426119002
-0.44959754226293047	-1.005803203180794	11169.119344884588
-0.42781796412643114	-1.0746356897259306	66147.13885247851
-0.43325970741135617	-2.088247012476564	107756.0198839616
-0.4424849737776291	-1.0961420901825218	19990.948230056012
13.79875644355442	-1.996934830930518	9826855.297081808
-0.4293766908075585	-1.0770966226259604	59942.85256205688
-0.4926085956420153	-1.1467137667673497	22718.374132550452

Finalmente se obtuvo una precisión global de: 326541.855636724

5. CONCLUSIONES Y REFLEXIONES

-David

Resultó muy sencillo e intuitivo ejecutar los métodos numéricos requeridos gracias a las librerías disponibles. Se evitó tener que implementar detalles de bajo nivel, lo cual permitió dedicar más tiempo al análisis de los resultados. La presente práctica ayudó a entender mejor los fundamentos del aprendizaje automático desde una aplicación de la vida real, ya que resulta muy diferente aprender la teoría que ponerla en acción.

-Luis

La práctica resulto bastante interesante. Aunque tube que repasar e investigar muchos conceptos de estadística y algebra lineal, fue bastante gratificante ver como el modelo funcionaba y generaba resultados. Antes de empezar la práctica pense que en lo que más iba a batallar era en la codificación en sí y el uso de las librerías pero para mi sorpresa todo salío bastante bien y lo que se me complico más fue la interpretación de los datos y programar el gradiente. Eso sí creo que requiero de más práctica para mejorar mis habilidades en estadística, intepretación de datos y algebra lineal.