Universidad Nacional de Ingeniería

Facultad de Ciencias
Departamento de Ciencia de la Computación

Proyecto formativo

REGRESIÓN LINEAL Y CLUSTERING

Grupo: Equipo 07

INTEGRANTES:

-Espinosa Pari, Franklin	Cód: 20210135D
-Estacio Sanchez, Jose Rodolfo	Cód: 20214027A
-Chavez Chico, Joel Jhotan	Cód: 20210058J9
-Manrique Candela, Esmit jack	Cód: 20210125I

SECCIÓN: 'A'

Profesores:

- Ccoicca Pacasi, Yuri Javier
- Espejo Delzo, Juan Carlos

Fecha de realización: 14/11/2021
Fecha de entrega: 20/12/2021

Índice

RESUMEN

El objetivo de este estudio es conocer el aprendizaje automático o aprendizaje de máquinas, más conocido en el mundo actual como el Machine Learning, que es un subcampo de las ciencias de la computación y una rama de la inteligencia artificial cuya finalidad es desarrollar técnicas que permitan a las computadoras aprender, convirtiéndose en un pilar fundamental para el trato de datos a gran escala.

Finalmente, los campos de aplicación en los cuales se ha enfocado son: la medicina, la construcción, las finanzas, la robótica, la educación entre otras.

Palabras claves:

- Machine Learning
- Ciencias de la computación
- Inteligencia Artificial
- Trato de proliferación de datos

1. Introducción

El análisis simultáneo de información, en conjunto con el procesamiento estadístico suele ser una de las tareas más importantes a nivel global.

Dentro del campo de la computación, existen múltiples métodos para resolver problemas que se han desarrollado a lo largo de su historia.

Por eso en este documento trataremos de ver como Machine Learning se convirtió en un pilar fundamental para el trato de datos a gran escala. Algunos sectores como el informático, salud, corporativo y de transporte.

En el transcurso de las últimas décadas han encontrado la solución a esta ardua tarea de aprendizaje y predicción en una de las cuantas disciplinas procedentes de la inteligencia artificial; esta materia o herramienta informática es el Machine Learning (herramienta que buscan mejorar el análisis de datos).Por otra parte, esta materia promete una gran utilidad en campos como la medicina, robótica, y mecánica. Tanto así que a muchos les preocupa los daños que traerá esta innovadora herramienta consigo. Debido a que, se trata de generar organismos mecánicos capaces de ser más inteligentes, sin que sea necesaria la intervención humana constante, que nos asegura que no seremos mentalmente obsoletos en un futuro al volvernos dependientes de estos sistemas.

2. Objetivos

Objetivos generales:

-Informar acerca de lo que es Machine Learning y cómo puede ser empleado para las codificaciones de algoritmos en C++.

Objetivos específicos:

-Codificar algoritmos de regresión lineal y clustering con la teoría de Machine Learning y sus derivados.

-Implementar y hacer uso de la librería **MLpack** en nuestro compilador.

3. Marco teórico

3.1. Regresión Lineal

La regresión lineal es una técnica de modelado estadístico que se emplea para describir una variable de respuesta continua como una función de una o varias variables predictoras. Puede ayudar a comprender y predecir el comportamiento de sistemas complejos o a analizar datos experimentales, financieros y biológicos.

Las técnicas de regresión lineal permiten crear un modelo lineal. Este modelo describe la relación entre una variable dependiente Y (también conocida como la respuesta) como una función de una o varias variables independientes X_i (denominadas predictores). La ecuación general correspondiente a un modelo de regresión lineal es:

Ecuación general:

$$Y = \beta_0 + \Sigma\beta_i X_i + \epsilon_i \tag{1}$$

donde β representa las estimaciones de parámetros lineales que se deben calcular y ϵ_i representa los términos de error.

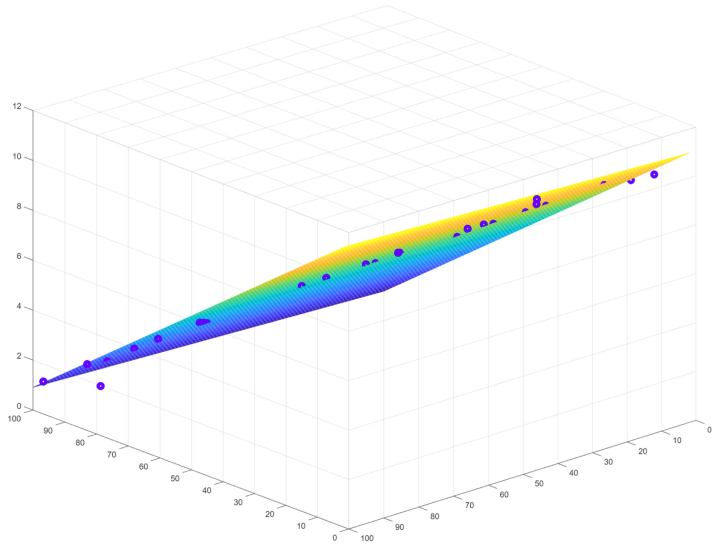

Figura 1: Gráfica de regresión lineal

Podemos llevar acabo dos modelos de análisis distintos en función del número de variables:

- Modelo de regresión lineal simple.
- Modelo de regresión lineal múltiple.

Sin embargo, en esta oportunidad veremos sólo la regresión lineal simple.

3.1.1. Regresión lineal simple

El análisis de regresión lineal simple es el más utilizado y el más sencillo de todos. Se trata de estudiar el efecto de una variable independiente (denominadas predictores) sobre una única variable dependiente (también conocida como la respuesta) de la primera o que al menos a nivel teórico hemos considerado que es dependiente. Empleando esta ecuación de regresión lineal simple se puede realizar una estimación basándose en los datos obtenidos.

Ecuación de regresión lineal simple:

$$Y = \beta_0 + \beta_0 X + \epsilon_0 \tag{2}$$

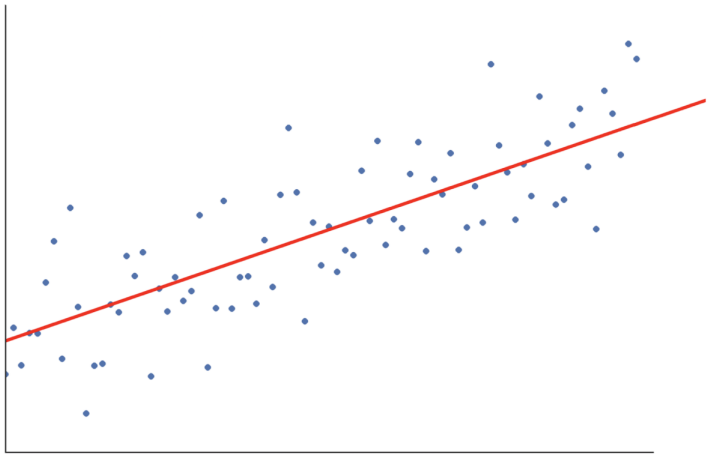

Figura 2: Gráfica de regresión lineal simple

- Donde Y es la variable dependiente, X es la variable independiente, β_0 es la constante o el intercepto de Y, β_1 es la pendiente o el coeficiente de X y ϵ es el término de error.
- El término de error, ϵ, refleja el hecho de que la relación entre Y x no es exacta, sino que está sujeto a algún error. Tenga en cuenta que β_0 y β, uno son parámetros que no pueden ser observados directamente, solo podemos estimarlos usando los valores de Y y x. Igualmente, el término de error, ϵ, no se puede observar directamente.
- El valor predicho de y, escrito como $\hat{\boldsymbol{Y}}$, se define de la siguiente manera:

  Predirector de Y = $\hat{\boldsymbol{Y}}$

$$\hat{\boldsymbol{Y}} = \hat{\boldsymbol{\beta}}_{\mathbf{0}} + \hat{\boldsymbol{\beta}}_{\mathbf{1}} X \tag{3}$$

  Donde $\hat{\boldsymbol{\beta}}_{\boldsymbol{i}}$ es el valor estimado del verdadero parámetro β_i. Como puede ver, el "gorro" indica el estimado del parámetro poblacional basado en los datos de la muestra.

  Residual

- El residual es la diferencia entre el valor verdadero y el valor predicho:

$$\textbf{residual} = Y - \hat{\boldsymbol{Y}} = e \tag{4}$$

- No confunda ϵ y e: ϵ es el término de error no observado en la "verdadera" relación entre Y y x, mientras que e es la diferencia observada entre Y y su valor predicho, $\hat{\boldsymbol{Y}}$, el último basado en la relación estimada entre Y y x.

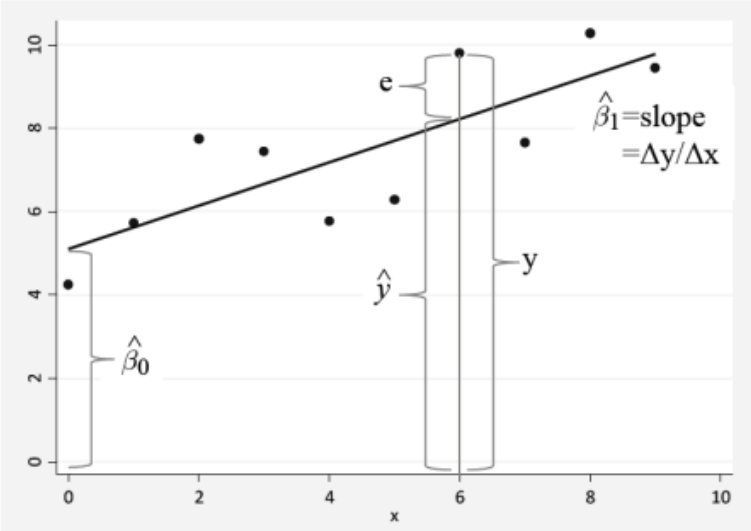

Figura 3: Interpretación gráfica del residual

3.1.2. Función de coste

La pérdida permite medir la diferencia existente entre los datos reales (y) y los datos obtenidos tras realizar la Regresión Lineal (que en adelante llamaremos $\hat{\boldsymbol{Y}}$).
Existen diferentes formas de definir matemáticamente esta pérdida, pero la más usada es a través del error cuadrático medio (ECM), definido de la siguiente manera:

$$ECM = \frac{1}{N} \cdot \Sigma_{i=1}^{N} (y_i - \hat{\boldsymbol{y}}_{\boldsymbol{i}})^2 \tag{5}$$

Donde N es el número total de datos que se tienen originalmente, y_i y $vu * y_i$ son cada uno de los datos originales y los obtenidos a partir de la regresión, y la resta (y_i-$\hat{\boldsymbol{y}}_{\boldsymbol{i}}$)se eleva al cuadrado para que todas las diferencias que están en la sumatoria sean positivas y no se cancelen entre sí.

Esta resta se ilustra en la siguiente figura:

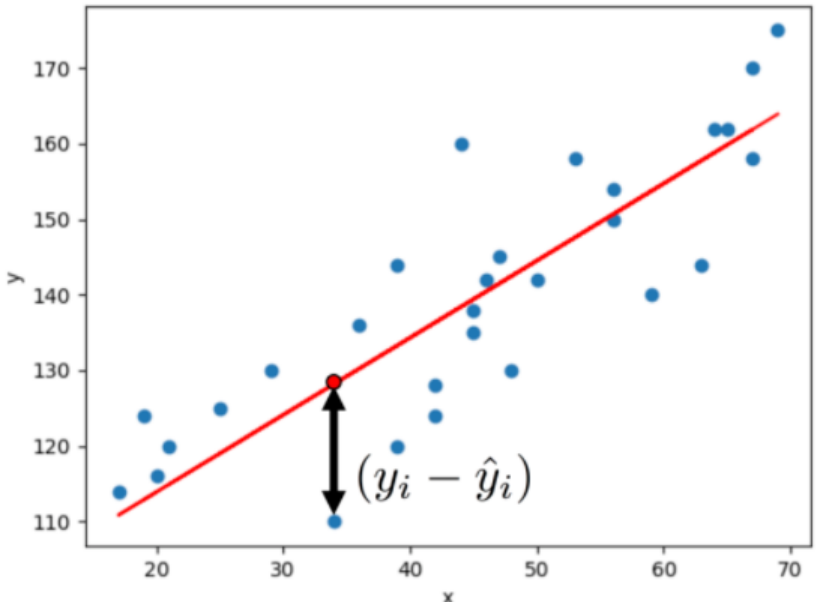

Figura 4: Ejemplo del cálculo del error cuadrático medio para un par de puntos

Así, el ECM mide el error promedio existente entre los datos originales y los datos obtenidos a partir de la Regresión Lineal.

Habiendo definido la pérdida, podemos ahora discutir a qué hace referencia el término "mejor ajuste".

3.2. Gradiante descendente

En este caso el Gradiente Descendente se usa para minimizar el ECM, y será una función de dos variables: w y b. Por tanto, al hacer una gráfica de su comportamiento, encontraremos que para el caso de la Regresión Lineal ésta tiene forma de tazón, como se muestra en la figura de abajo. El mínimo que estamos buscando se encuentra al fondo de dicho tazón:

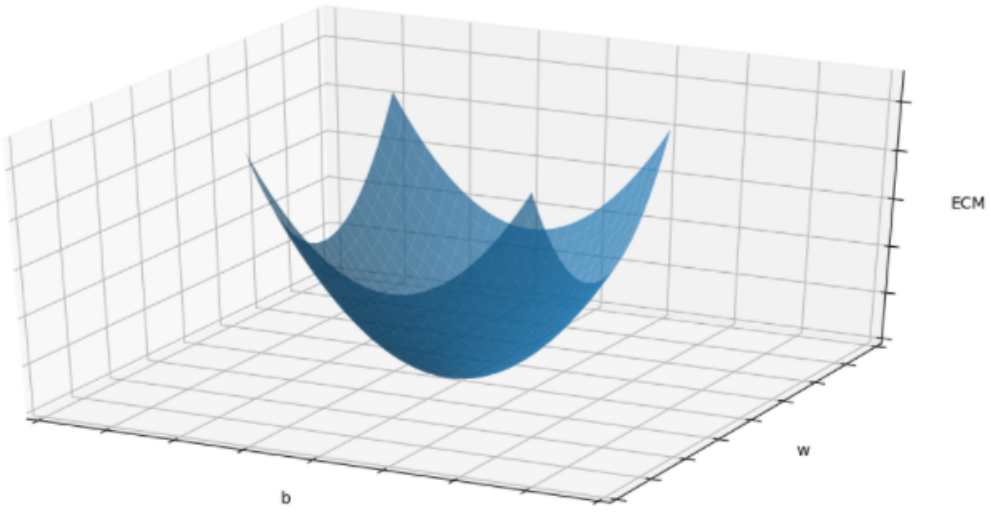

Figura 5: Comportamiento de la pérdida en el caso de la regresión lineal

Así, al usar el Gradiente Descendente para minimizar la pérdida, lo que haremos es de forma iterativa calcular los valores de w y b que hacen que el ECM sea cada vez más pequeño (que nos acerquemos progresivamente al fondo del tazón). Para ello usamos la ecuación general del gradiente descendente:

3.2.1. Algoritmo de la gradiante descendente

$$w \leftarrow w - \alpha dw \tag{6}$$

$$b \leftarrow b - \alpha db \tag{7}$$

Donde α es la tasa de aprendizaje y dw y db hacen referencia a la derivada (gradiente) de la pérdida con respecto a los parámetros w y b:

$$dw = \frac{-2}{N} \cdot \Sigma_{i=1}^{N} x(y_i - wx_i - b)^2 \tag{8}$$

$$db = \frac{-2}{N} \cdot \Sigma_{i=1}^{N} (y_i - wx_i - b)^2 \tag{9}$$

3.3. Regresión Logística

La regresión logística es un algoritmo de clasificación de aprendizaje supervisado que se utiliza para predecir la probabilidad de una variable objetivo. La naturaleza de la variable objetivo o dependiente es dicotómica, lo que significa que solo habría dos clases posibles.

Matemáticamente, un modelo de regresión logística predice P (Y = 1) como una función de X. Es uno de los algoritmos ML más simples que se puede utilizar para varios problemas de clasificación como detección de spam, predicción de diabetes, detección de cáncer, etc.

Modelo de regresión logística binaria: la forma más simple de regresión logística es la regresión logística binaria o binomial en la que la variable objetivo o dependiente puede tener solo 2 tipos posibles, ya sea 1 o 0.

Modelo de regresión logística multinomial: otra forma útil de regresión logística es la regresión logística multinomial en la que la variable objetivo o dependiente puede tener 3 o más tipos desordenados posibles, es decir, los tipos que no tienen significación cuantitativa.

3.3.1. Función hipótesis

La función hipótesis estima la probabilidad de pertenecer a una de las clases.

Antes de comenzar con la función de costo, recuerde que la hipótesis de regresión logística está definida como:

$$h_\theta(x) = g(\theta^T x) \tag{10}$$

Donde la función g es la función sigmoide definida como,

$$g(z) = \frac{1}{1 + e^{-z}} \tag{11}$$

3.3.2. Función de coste para regresión logística

Se busca minimizar esta función de coste del error cuadrático.

$$J(\theta) = \frac{1}{m} \cdot \Sigma_{i=1}^{m}[-y^i \cdot \log\big(h_\theta(x^i)\big) - (1 - y^i) \cdot \log\big(1 - h_\theta(x^i)\big)] \tag{12}$$

3.4. Clustering

Es de las herramientas más utilizadas durante el EDA para ver segmentaciones de los datos y tener una intuición de cómo están estructurados los datos.

Esta técnica no supervisada se basa en identificar grupos en los datos de tal manera que todos los datos del grupo (clúster) son datos con características similares mientras que los datos de los otros grupos son diferentes.

Algoritmo K-Means

K-means es un algoritmo de clasificación no supervisada (clustering) que agrupa objetos en k grupos basándose en sus características. El agrupamiento se realiza minimizando la suma de distancias entre cada objeto y el centroide de su grupo o cluster. Se suele usar la distancia cuadrática.

El algoritmo consta de tres pasos:

1. **Inicialización:** una vez escogido el número de grupos, k, se establecen k centroides en el espacio de los datos, por ejemplo, escogiéndolos aleatoriamente.

2. **Asignación objetos a los centroides:** cada objeto de los datos es asignado a su centroide más cercano.

3. **Actualización centroides:** se actualiza la posición del centroide de cada grupo tomando como nuevo centroide la posición del promedio de los objetos pertenecientes a dicho grupo.

Se repiten los pasos 2 y 3 hasta que los centroides no se mueven, o se mueven por debajo de una distancia umbral en cada paso.

El algoritmo k-means resuelve un **problema de optimización**, siendo la función a optimizar (minimizar) la suma de las distancias cuadráticas de cada objeto al centroide de su cluster.

Los objetos se representan con vectores reales de **d** dimensiones (x_1,x_2,...,x_n) y el algoritmo k-means construye **k** grupos donde se minimiza la suma de distancias de los objetos, dentro de cada grupo $\mathbf{S} = S_1,_2,\ldots,S_n$, a su centroide. El problema se puede formular de la siguiente forma:

$$_S minE(\mu_i =_S minE \Sigma_{i=1}^{k} \Sigma_{x_j \epsilon S_i} ||x_j - \mu_i||^2 \tag{13}$$

donde **S** es el conjunto de datos cuyos elementos son los objetos x_j representados por vectores, donde cada uno de sus elementos representa una característica o atributo. Tendremos **k** grupos o clusters con su correspondiente centroide μ_i .

En cada actualización de los centroides, desde el punto de vista matemático, imponemos la condición necesaria de extremo a la función E(μ_i) que, para la función cuadrática (1) es

$$\frac{\delta E}{\delta \mu_i} = 0 \Longrightarrow \mu_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \Sigma_{x_j \epsilon S_i^{(t)}} X_j \tag{14}$$

y se toma el promedio de los elementos de cada grupo como nuevo centroide.

Las principales ventajas del método k-means son que es un método sencillo y rápido. Pero es necesario decidir el valor de **k** y el resultado final depende de la inicialización de los centroides. En principio no converge al mínimo global sino a un mínimo local.

4. Código en C++

4.1. Algoritmo K-Means con MLpack

Se incluye los siguientes archivos de encabezado propios del mlpack más una librería básica del c++, asimismo se usarán los namespaces para evitar saturar el código más adelante.

```cpp
#include <mlpack/core.hpp>
#include <bits/stdc++.h>// include basic c++ liraries
#include <mlpack/methods/kmeans/kmeans.hpp>// mlpack library for k-means
#include <armadillo>//ml pack is dependent on armadillo

using namespace arma;
using namespace mlpack;
using namespace std;
```

A continuación, se declara algunas variables básicas para establecer el número de clústeres (k) y la cantidad máxima de iteraciones que queremos hacer. ¿Por qué? Porque K-means es un proceso iterativo.

Además, se crea una matriz trainData para almacenar los features y otra llamada centroids para contener los centroides de cada cluster, y un vector fila para guardar el cluster al cual pertenece cada muestra (cada cluster es representado por un número).

```cpp
int k = 3, max_iter = 2;
mat dataset;
mat trainData;
mat centroids;
Row<size_t> clusters;
```

Load Data:

Ahora, trabajaremos en la función main, cargamos nuestro dataset bajo del nombre de "data.csvz lo alojamos en nuestra matriz dataset.

```cpp
    data::Load("C://Users//Joel Chavez//Desktop//proy formativo//data_franklin.csv", dataset);
```

Train Data:

Extraemos las columnas del dataset que nos interesa (los features), y limpiamos la fila 0 que no nos interesa. Luego imprimos en la consola los resultados.

```cpp
    trainData = dataset.cols(0, dataset.n_rows - 1);
    trainData.shed_row(0);
    trainData.print(" -> Train Daata");
```

Build the model 'K-Means':

Una vez cargada la data, necesitamos crear una instancia de la clase KMeans, para luego inmediatamente especificar la cantidad de iteraciones máximas y pasarlo al constructor.

```
23      kmeans::KMeans<> mlpack_kmeans(max_iter);
```

Así que ahora podemos seguir adelante y hacer el clustering. Llamaremos a la función de miembro Cluster de esta clase K-means. Necesitamos pasar los datos, el número de clústeres, y luego también necesitamos pasar el objeto del clúster y el objeto del centroide.

Ahora, esta función de clúster ejecutará K-means en estos datos con un número específico de clústeres, y luego inicializará estos dos objetos: clústeres y centroides

```
24      mlpack_kmeans.Cluster(trainData, k, clusters, centroids);
```

Para ver los resultados por parte de los centroides y clusters respectivos, imprimiremos en la consola los resultados, para ello usamos simplemente el método print.

```
26      centroids.print("\n -> Centroides");
27      clusters.print("\n -> Clusters");
```

4.2. Regresión Lineal Multivariable con MLpack

1. Para poder facilitar el código usaremos la librería MLPack y especialmente los siguientes encabezados, y también se agregarán los "namespace" para una mejor visión del código.

```
1  #include<mlpack/core.hpp>
2  #include<mlpack/methods/linear_regression/linear_regression.hpp>
3
4  using namespace arma;
5  using namespace mlpack;
6  using namespace std;
```

2. Luego, se declara una matriz llamada "dataset", donde lo rellenaremos de los datos dentro del archivo "data.csv".

```
10      mat dataset;
11      data::Load("C://Users//Joel Chavez//Desktop//proy formativo//data.csv", dataset);
```

3. A continuación, entrenamos nuestra maquina ingresándole datos del "dataset". En la matriz "trainData" extraeremos los features y en el vector "trainTarget" extraeremos los precios por área.

```
13      mat trainData = dataset.rows(1, dataset.n_rows - 2);
14      rowvec trainTarget = dataset.row(dataset.n_rows - 1);
```

4. Ahora se crea el modelo de regresión lineal y lo entrenaremos ingresándole los datos del trainData y trainTarget como parámetros.

```
16      regression::LinearRegression regressor;
17
18      regressor.Train(trainData, trainTarget);
```

5. Pero antes usaremos un ejemplo donde se guardarán los features y así predecir su precio de la casa por área, para esto pediremos al usuario que ingrese los 6 features, con los valores cercanos a los del dataset.

```
22      float x[6];
23      cout << "---------FEATURES-------\nX1: Fecha de transicion,\tX2: Anos de la casa \nX3: Distancia a la
24      for (int i = 0; i < 6; i++) {
25          cout << "Ingrese X" << i + 1 << " : ";
26          cin >> x[i];
27      }
28      colvec aux = { x[0], x[1], x[2], x[3], x[4], x[5] };
29      mat points = aux.col(0);
```

6. Finalmente, predeciremos el precio y lo imprimiremos.

```
31      regressor.Predict(points, predictions);
32      predictions.print("Prediction: ");
```

Extra: Obtendremos la función multivariable:

Sea f: X $\rightarrow$ Y, del tipo:

$$Y = f(x_1, x_2, x_3, x_4, x_5, x_6) = A_{x_0} + B_{x_1} + C_{x_2} \ldots\ldots + F_{x_5}$$

```
34      cout << "FUNCIÓN: \n Y = A + Bx1 + Cx2 + .... + Dx6 \n ";
35      regressor.Parameters().print("Parametros: ");
```

5. Resultados

- Después de numerosos intentos por instalar las librerías se logró implementar la librería MLpack, con sus clases y métodos sencillos de manejar, asimismo se instaló

la librería Shogun pero debido a su complejidad solo se usó la primera librería.

- Se logró construir un modelo de regresión lineal con 6 variables para predecir el costo de una vivienda, además se pudo clasificar en 3 grupos gracias al algoritmo K-Means.
-

6. Conclusiones

- De forma somera, pero entendible se brindó la información necesaria de la teoría de Machine Learning.
- Se codificó dos modelos: Regresional lineal y Clustering.
- Se empleó la librería Mlpack en ambos códigos.

7. Referencias bibliográficas

-Regresión Lineal a paso a paso en python (s.f.). Aprende Machine Learninghttps://www.aprendemachinelearning.com/regresion-lineal-en-espanol-con-python/#more-5722

-¿Qué es la regresión lineal? (s.f). Mathworks https://la.mathworks.com/discovery/linear-regression.html

-Análisis de regresión. (s.f.). Delsol https://www.sdelsol.com/glosario/analisis-de-regresion/

-Análisis de Regresión Lineal. (s.f.). http://herzog.economia.unam.mx/profesores/blopez/tallerVI-practica5.pdf

-Sotaquirá M. (16-07-2018). La Regresión Lineal en el Machine Learning. cbcodificandobits. https://www.codificandobits.com/blog/regresion-lineal/

-pardo C. J. (s.f.). Machine Learning Regresión Logística. Visión pòr computador https://carlosjuliopardoblog.wordpress.com/2017/12/31/regresion-logistica/

-Aprendizaje automático: regresión logística. (s.f.). Aprendizaje automático: regresión logística. Tutorialpoint https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_logistic_regression.

htm