

# **Documento de comparación de modelos de regresión lineal**

Mitchell Phillip Bermin Suarez

Facultad de Ciencias exactas e Ingeniería, Universidad Sergio Arboleda

Introducción a HPC

Prof. John Corredor Franco

22 de noviembre de 2022

## Introducción

La búsqueda de la optimización en los ámbitos computacionales siempre ha sido un factor determinante en las decisiones que se toman a cabo para la resolución de problemas. En este documento trataremos los resultados de un ejercicio realizado a datos numéricos entre los lenguajes de programación python y C++. Esto con el fin de analizar sus similitudes, diferencias y ámbitos en los que se destacan los lenguajes.

## Equivalencias entre lenguajes

Durante el desarrollo de ambos modelos de regresión se pudo observar que el manejo de los datos es particularmente similar, puesto que en ambos modelos es necesario la segmentación de las variables y la separación de los datos para las pruebas es notoriamente similar en ambos casos.

```
mlr = LinearRegression()  
X = df[['engine_size', 'Cylinders', 'fuel_cons_comb']]
```

Nota. Segmentación variables Python. CC BY 2.0

```
Eigen::MatrixXd m;  
  
m.conservativeResize(matData.rows(), 4);  
m.col(0) = matData.col(3);  
m.col(1) = matData.col(4);  
m.col(2) = matData.col(9);  
m.col(3) = matData.col(11);
```

Nota. Segmentación variables C++. CC BY 2.0

```
X = df[['engine_size', 'Cylinders', 'fuel_cons_comb']]  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)  
X_train_sm = s.fit_transform(X_train)  
mlr.fit(X_train_sm, y_train)  
X_test_sm = s.transform(X_test)  
y_pred_sm = mlr.predict(X_test_sm)
```

Nota. Separación de los datos Python. CC BY 2.0

```
/* Se dividen en datos de entrenamiento y datos de prueba*/  
Eigen::MatrixXd X_train, y_train, X_test, y_test;  
std::tuple<Eigen::MatrixXd, Eigen::MatrixXd, Eigen::MatrixXd, Eigen::MatrixXd> div_datos = ExData.TrainTestSplit(mat_norm, 0.8);  
  
// Se descomprime la tupla en cuatro conjuntos  
std::tie(X_train, y_train, X_test, y_test) = div_datos;  
  
/* Se crean vectores auxiliares pra prueba y entrenamiento inicializados en 1*/  
Eigen::VectorXd V_train = Eigen::VectorXd::Ones(X_train.rows());  
Eigen::VectorXd V_test = Eigen::VectorXd::Ones(X_test.rows());
```

Nota. Separación de los datos C++. CC BY 2.0

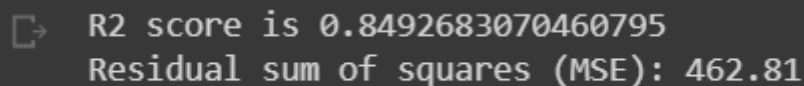
Es prudente mencionar que en el modelo de C++ es necesario mantener un espacio reservado en memoria para realizar los cálculos, esto es debido a la libertad en la memoria que brinda el lenguaje. Se puede concluir que en todo lo referido a manejo de los datos y orden de instrucciones los modelos tienen una estructura similar.

### Diferencias entre modelos

Como se pudo observar en el anterior segmento del documento la principal diferencia entre los modelos es la utilización de librerías externas. En el caso del modelo ejecutado en C++, todos los cálculos necesarios fueron realizados por completo. Además de lo anterior mencionado el modelo presentado en python, dado al uso de librerías, presenta una facilidad para analizar los datos a calcular. Este hecho presenta una flexibilidad en la observación de los datos que el modelos en C++ no presenta. Para revisar ambos de los modelos el lector se puede dirigir a: <https://github.com/mitch222/parcial3>.

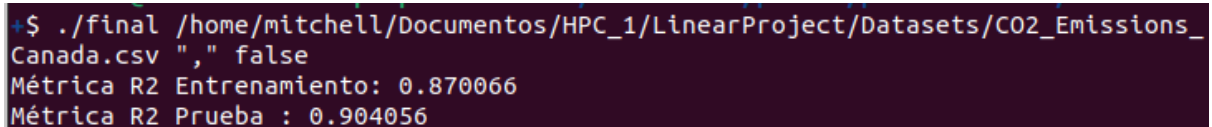
### Resultados

Como he de esperar se ambos modelos presentan una calificación R2 similar. Este resultados tiene una variación dado al manejo estadístico que presentan las librerías usadas (Sklearn en el caso de python y Eigen en el caso C++). La velocidad del análisis fue mayor en el caso del modelo de Python debido al uso de la



```
➜ R2 score is 0.8492683070460795
Residual sum of squares (MSE): 462.81
```

Nota. Análisis de Python. CC BY 2.0



```
➜$ ./final /home/mitchell/Documentos/HPC_1/LinearProject/Datasets/CO2_Emissions_
Canada.csv "," false
Métrica R2 Entrenamiento: 0.870066
Métrica R2 Prueba : 0.904056
```

Nota. Análisis de C++. CC BY 2.0

### Conclusiones

Durante el desarrollo de los modelos se pudo palpar las facilidades que presentan las herramientas actuales (Librerías y frameworks) para el desarrollo no solamente de análisis de datos sino también en la efectividad para el desarrollo de modelación estadística. Sin embargo, para tener un mayor entendimiento de las técnicas tratadas y los modelos utilizados actualmente para análisis de datos, el uso de un lenguaje que no lleve al desarrollador de la mano es conveniente.

## Referencias:

- Badole, M. (2021). Multiple Linear Regression Using Python and Scikit-learn. Retrieved 19 November 2022, from <https://www.analyticsvidhya.com/blog/2021/05/multiple-linear-regression-using-python-and-scikit-learn/>
- Triola, M.F. (2012). Elementary Statistics. International Edition ed. Boston: Pearson Education, Inc.
- Eigen: Slicing and Indexing. (2022). Retrieved 22 November 2022, from [https://eigen.tuxfamily.org/dox/group\\_\\_TutorialSlicingIndexing.html](https://eigen.tuxfamily.org/dox/group__TutorialSlicingIndexing.html)