# Tres técnicas de regresión con aprendizaje de máquina: Regresiones Lineal, Ridge y Lasso

Autor:

Juan Carlos Mogotocoro Sanabria

Trabajo de grado como requisito para optar al título de Matemático

Director:

Andrés Sebastián Ríos Gutiérrez

Candidato a Doctor en Estadística

Universidad Industrial de Santander

Facultad de Ciencias

Escuela de Matemáticas

Bucaramanga

2024

# Dedicatoria

El presente trabajo esta dedicado especialmente a mi familia, por su gran apoyo que me han brindado a lo largo de la carrera. Igualmente, al esfuerzo y la determinación de nunca rendirme. También extiendo mi gratitud a mis amigos, quienes compartieron conmigo durante mi etapa universitaria.

# Agradecimientos

Agradezco a Dios y a mi familia por todo el apoyo recibido durante mi desarrollo académico. En especial a mis padres y abuelos paternos, por su infinito amor, apoyo incondicional y por ser mi fuente de inspiración. A mis hermanas, por su compañía, paciencia y por siempre estar ahí para mí.

A mi director de tesis, por su gran conocimiento, por su guía y su orientación constante en mi sendero académico. Asimismo, agradezco a mis amigos, por las risas, consejos y por celebrar cada logro obtenido.

# Tabla de contenido

# Índice de figuras

# Resumen

**Titulo:** Tres técnicas de regresión con aprendizaje de máquina: Regresiones Lineal, Ridge y Lasso

**Autor:** Juan Carlos Mogotocoro Sanabria

**Palabras claves:** Regresión lineal, regresión Ridge, regresión Lasso, aprendizaje de maquina, regularización, validación cruzada.

**Descripción:** En la actualidad el aprendizaje automático sirve para analizar grandes bases de datos almacenados en los avances tecnológicos y empresas. Normalmente, existen muchas variables predictoras en el modelo de regresión lineal. No obstante, el método de mínimos cuadrados ordinarios hace varias suposiciones sobre los datos lo que genera que no sea ciertas en datos reales. Frecuentemente, ocasiona problemas al ajustar el modelo mediante mínimos cuadrados. La dificultad más común es que el modelo se ajuste demasiado a los datos, esto pasa cuando el estimador es insesgado pero tiene una alta variabilidad.

Para este problema las regresiones Ridge y Lasso son dos técnicas de regularización utilizadas para crear un modelo mejor y más preciso. Las cuales se basan en reducir el número de variables imponiendo una penalización sobre los coeficientes de regresión que obliga a que los coeficientes tiendan a cero o incluso sean cero. Por último, se realiza una implementación con dos conjuntos de datos donde se comparan los tres

modelos y se elige el que mejor tenga ajuste con el criterio del menor error cuadrático medio (ECM).

# Abstract

**Title:** Three regression techniques within machine learning: Linear Regression, Ridge Regression, and Lasso Regression.

**Author:** Juan Carlos Mogotocoro Sanabria

**Keywords:** Linear regression, Ridge regression, Lasso regression, machine learning, regularization, cross-validation.

**Description:** Currently, machine learning is used to analyze large datasets stored in technological advancements. Typically, there are many predictor variables in the linear regression model. However, the ordinary least squares method makes several assumptions about the data that are often not true in real-world datasets. Frequently, this causes issues when fitting the model using least squares. The most common difficulty is that the model overfits the data, which occurs when the estimator is unbiased but has high variability.

For this problem, Ridge and Lasso regressions are two regularization techniques used to create a better and more accurate model. These techniques aim to reduce the number of variables by imposing a penalty on the regression coefficients, forcing them to tend towards zero or even be zero. Finally, an implementation is carried out using two datasets where the three models are compared, and the one with the best fit is chosen based on the criterion of the lowest mean squared error (MSE).

# Introducción

Una de las técnicas más básicas en estadística y aprendizaje automático para analizar datos es la regresión lineal, ya que permite predecir el valor de datos desconocidos utilizando información previamente conocida y relacionada. Este modelo proporciona una fórmula matemática simple cuya interpretación es directa, lo cual facilita la generación de predicciones. La primera evidencia de regresión lineal se registró en 1805 por Legendre, quien utilizó el método de los mínimos cuadrados. Posteriormente, Gauss publicó un estudio que amplió aún más este método e introdujo también el teorema de Gauss-Markov (Draper y Smith, 1998).

Con el transcurso del tiempo, la regresión lineal ha evolucionado en una técnica estadística de fácil aplicación en la computación. Es ampliamente utilizada por empresas y científicos para transformar datos en inteligencia empresarial, realizar análisis preliminares de datos y predecir tendencias futuras. Actualmente, en el Machine Learning se emplea la regresión lineal para resolver problemas, como por ejemplo, elaborar clasificaciones binarias. El Lasso y Ridge son dos técnicas muy conocidas en el ámbito del aprendizaje automático, las cuales se emplean para abordar el problema de la multicolinealidad presente en la regresión lineal.

Las técnicas anteriores actúan como métodos reguladores. Estos métodos fuerzan a que los coeficientes del modelo tiendan a cero, reduciendo varianza, las cuales contribuyen a disminuir el sobre-ajuste y aumentar la precisión del modelo. La regresión Lasso, también denominada regresión de mínimos cuadrados parciales

y selección de variables, incorpora un término penalizador, el cual se utiliza para favorecer que algunos de los coeficientes sean cero. Esto es útil para descubrir cuáles atributos son relevantes y así obtener un modelo que logre una mejor generalización. Mediante esta técnica se castigan los coeficientes de las variables irrelevantes al ser forzados a cero; esto implica que la regresión Lasso es capaz de realizar una selección automática de atributos.

En contraste, el enfoque de regresión Ridge se incorpora mediante la adición de una función de penalización a la función de pérdida de mínimos cuadrados. La función de penalización más común utilizada en la regresión Ridge es la norma L2, la cual se define como la suma de los cuadrados de los coeficientes del modelo. A diferencia del método Lasso, que fuerza los coeficientes a cero, la regresión Ridge reduce gradualmente su magnitud hacia cero. Como resultado, no se realiza una selección automática de las características como lo hace la regresión Lasso; sin embargo, puede disminuir la influencia de variables irrelevantes sin eliminarlas por completo.

El término de penalización en la regresión Ridge se incorpora mediante la adición de una función de penalización a la función de pérdida de mínimos cuadrados. La función de penalización más común utilizada en la regresión Ridge es la norma L2, que se define como la suma de los cuadrados de los coeficientes del modelo. Estas dos técnicas brindan una solución al problema de la multicolinealidad esto ocurre cuando los modelos de regresión lineal tienen las variables independientes correlacionadas entre sí. Esto puede causar problemas en la estimación de los coeficientes del modelo,

ya que estos se vuelven más inestables y difíciles de interpretar, aunque se abordan de maneras ligeramente diferentes.

La elección entre regresión Lasso y Ridge dependerá de las características específicas del conjunto de datos de los objetivos del modelo. Por lo tanto, la regresión Lasso como la regresión Ridge son métodos de regularización útiles para mejorar la precisión de los modelos de regresión lineal cuando hay multicolinealidad presente. Por último, se implementaran las tres regresiones (lineal, Ridge y Lasso) entre dos conjuntos de datos, para así evaluar el rendimiento de cada modelo, esta implementación se realizará usando lenguaje `Python`. En donde se contrastaran las tres regresiones, con el fin de seleccionar la que tenga el mejor ajuste a través de su coeficiente de determinación o su error cuadrático medio.

En Colombia, como en muchos otros países, la salud y la educación representan dos fundamentos necesarios para el desarrollo individual y social. En este sentido, se maneja dos bases de datos 'Recién nacidos Guadalajara de Buga 2016' y 'Saber 11 del año 2020-2', para el primer conjunto de datos se analiza el peso del recién nacido, pues este es un factor importante para el desarrollo del bebé, para ello se tendrá en cuenta el tiempo de gestación y la edad de la madre. Con estos datos disponibles se observa como esas variables influyen en el peso del recién nacido, debido a que se evidencia científicamente que el tiempo de gestación y la edad de la madre. Le influye significativamente en el peso del recién nacido, como señala (Figueiredo, Gomes-Filho, Silva et al., 2018). Para el segundo conjuntos de datos,

el interés se centra en estudiar el percentil global con respecto al puntaje de lectura critica y matemáticas. Para Claessens y Engel (2013), se destaca la importancia de tener excelentes resultados en el puntaje de lectura critica y matemáticas. Puesto que les resulta un requisito fundamental a la hora de comprender numerosas disciplinas académicas. Dado que ambas asignaturas poseen un gran peso a la hora de calcular el percentil global.

Por último, los objetivos que se quieren alcanzar en este trabajo consiste en; establecer los conceptos básicos de las tres regresiones de este estudio, para así tener la estimación de los parámetros y la validación de los modelos. De igual modo, implementar una metodología en la selección de datos de entrenamiento y validación, para con ello estimar los datos de entrenamiento de las dos bases de datos. Finalmente, determinar el modelo que mejor se ajusta para los datos del ICFES y de Nacimientos de Buga.

# 1. Antecedentes

Unos de los métodos de aprendizaje mas antiguos es la regresión lineal, la cual se utiliza para modelar la relación que hay entre una variable dependiente y una o más variables independientes. Este método busca encontrar una recta que se ajuste a los datos dados, minimizando la suma de los cuadrados residuales. Este método fue desarrollada por Gauss en el sigo XIX el cual se basa en el teorema de Gauss - Markov, que establece que la regresión lineal es un estimador insesgado de mínimos cuadrados con una menor varianza.

Por otro lado, la regresión Ridge fue desarrollada por Hoerl y Kennard en 1970 y la regresión Lasso fue desarrollada por Tibshirani en 1996. En la técnica Ridge se utiliza para reducir el sobreajuste en la regresión lineal para ello se introduce un termino de penalización en la función de perdida que aumenta a medida que los coeficientes de la regresión son más grandes. Por otro lado, en la técnica Lasso también se maneja el sobreajuste y se añade una penalización a la función de perdida pero esta coge algunos coeficientes y los hace cero, esto ayuda a seleccionar variables importantes.

# 2. Planteamiento del problema

El problema que se presenta en un modelo de regresión lineal es el sobreajuste y la multicolinealidad. Esto sucede cuando hay muchas variables, esto implica que el modelo aprende de las particularidades del conjunto de datos de entrenamiento pero no generaliza bien a nuevos datos. Frecuentemente, los valores atípicos pueden tener un gran impacto en los coeficientes de la regresión lineal, causando una distorsión del modelo afectando así su precisión. Para esto introducimos las técnicas de regularización que se basan en añadir una penalización a dicha función de coste lo cual produce modelos más simples, con mejores ajustes y predicciones.

Para abordar este problema se introduce las regularizaciones Ridge y Lasso. En la Ridge introduce un sesgo en los coeficientes de la regresión para reducir la varianza del modelo, esto puede ser un problema si el sesgo es muy grande, ya que afecta la precisión del modelo. El rendimiento del modelo depende del hiperparametro lambda el cual se escoge aplicando validación cruzada. Elegir un valor pequeño puede resultar en un sobreajuste, en cambio un valor grande puede aumentar el sesgo. Por otro lado, en la regularización Lasso induce escasez en los coeficientes de la regresión, lo que significa que algunos coeficientes se reducen a cero. Es útil para seleccionar las variables importantes y reducir la complejidad del modelo.

En la regresión Lasso la interpretación de los coeficientes es más sencilla que la regresión lineal, ya que algunos coeficientes son cero. Al igual que la regresión Ridge, en la Lasso se introduce un sesgo en los coeficientes para reducir la varianza.

En la elección de lambda si es muy grande aumenta el sesgo y afecta a la precisión del modelo. Es importante tener en cuenta el sesgo introducido por la regularización Ridge y Lasso al elegir el valor del hiperparámetro. Se recomienda probar las tres regresiones y comparar sus resultados para elegir la mejor opción para cada caso particular.

En este trabajo, se analizara como los tres modelos de regresión se ajustan a dos conjuntos de datos, aplicando los principios teóricos de cada uno de los modelos. Para ello, usaremos el error cuadrático medio (ECM) como criterio para determinar que modelo es el más adecuado.

# 3. Justificación

Normalmente en las bases de datos con demasiados datos se presenta el problema del sobreajuste. Para suplir dicho problema se realiza la regularización adecuada para esto tenemos las regresiones Ridge y Lasso las cuales dependen de la elección del hiperparámetro lambda, ya que si el valor es pequeño resulta un sobreajuste en cambio si es grande aumenta el sesgo. En Ridge se aborda el problema de la multicolinealidad, donde las variables predictoras están muy correlacionadas, para ello se hace una penalización en los coeficientes de la regresión. Esto funciona cuando se trabajan con un número significativo de variables predictoras.

Igual que en la regresión Ridge, Lasso también aborda el problema de la multicolinealidad y realiza una selección de variables. La diferencia de estas técnicas es que Lasso tiene la capacidad de poner los coeficientes de la regresión en cero, lo que hace más valioso en la selección de características y la creación de modelos más interpretativos. En general, estas técnicas tiene soluciones efectivas para una gran variedad de problemas en el aprendizaje automático, pero la elección entre regresión lineal, Ridge y Lasso depende de las características del conjunto de datos y del objetivo del modelo.

# 4. Objetivos

## 4.1. Objetivo general

Analizar estadísticamente y hacer una comparación de tres técnicas de regresión: Lineal, Ridge y Lasso; desde un enfoque de aprendizaje de maquina.

## 4.2. Objetivos específicos

1. Establecer los conceptos básicos, para la estimación de parámetros y para la validación de los modelos lineales: regresión múltiple, regresión Lasso y regresión Rigde.

2. Implementar una metodología de selección de datos de entrenamiento y datos de validación.

3. Estimar utilizando los datos de entrenamiento, los modelos de regresión lineal, Lasso y Ridge, para los datos del ICFES y de Nacimientos de Buga.

4. Determinar con diferentes métricas, el modelo de mejor ajuste entre los modelos de regresión lineal, Lasso y Ridge, para los datos del ICFES y de Nacimientos de Buga.

# 5. Métodos y Marco Teórico

## 5.1. Modelos de regresión

Se quiere desarrollar un modelo de aprendizaje automático para predecir el precio de una casa. Para ello, se necesita recopilar los datos de las casas que ya se han vendido. La variable respuesta seria $Y$ = Precio de la casa y las variables de entrada seria $X_1$ = Tamaño de la casa, $X_2$ = Número de habitaciones, $X_3$ = Ubicación, En este ejemplo, las variables se utilizan para representar la información sobre las casas para predecir su precio. Las variables de entrada proporcionan información sobre las características de las casas, mientras que la variable de salida proporciona la información que se desea predecir, con este ejemplo se introduce la siguiente definición.

Se denota una variable de entrada por el símbolo $X$. Si $X$ es un vector, se puede acceder a sus componentes mediante los subíndices $X_j$. Los resultados cuantitativos se denotarán por $Y$, y los cualitativos por $G$ (de grupo). Utilizamos letras mayúsculas como $X$,$Y$ ó $G$ cuando nos referimos a los aspectos genéricos de una variable. Los valores observados se escriben en minúsculas; así, el i-ésimo valor observado de $X$ se escribe $x_i$ (donde $x_i$ es de nuevo un escalar o vector). Las matrices se representan con letras mayúsculas en negrita; por ejemplo, un conjunto de $N$ vectores $p$ de entrada $x_i$, $i = 1, ..., N$ se representaría la matriz $\mathbf{X}$ mediante $N \times p$. En general, los vectores no estarán en negrita, excepto cuando tengan $N$ componentes; esta convención distingue un vector $p$ de entradas $x_i$ para la $i$-esima

observación del vector $N$, $x_j$ que consiste en todas las observaciones sobre la variable $X_j$ ([4]).

Teniendo en cuenta a Balzarini, Di Rienzo, Tablada et al. [5], la estadística clásica se centra en describir y comprender los datos existentes, se utiliza para hacer inferencias sobre una población a partir de una muestra, para identificar patrones en los datos, y para probar hipótesis. En cambio el objetivo del aprendizaje automático es aprender de los datos para realizar tareas específicas. Se utiliza para desarrollar modelos que pueden predecir valores futuros, clasificar objetos, o tomar decisiones. Por lo anterior, la estadística clásica y la estadística con aprendizaje automático son dos ramas que tienen diferentes objetivos, métodos y datos.

**Definición 1.** *(**Función de perdida**) La función de pérdida es una función que mide la diferencia entre un valor observado y un valor previsto. Se utiliza para evaluar el rendimiento de un modelo estadístico [6].*

*Existen diferentes métodos, pero el más popular con diferencia, es el método de los mínimos cuadrados. En este método, elegimos los coeficientes $\beta$ para minimizar la suma residual de cuadrados*

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - x_i^T \beta)^2. \tag{1}$$

*$RSS(\beta)$ es una función cuadrática de los parámetros y, por tanto, su mínimo siempre existe, pero puede no ser único.*

**Definición 2.** *(**Datos de entrenamiento**) Los datos de entrenamiento son un conjunto de datos que se utilizan para entrenar un modelo estadístico. Estos datos se utilizan para estimar los parámetros del modelo y aprender las relaciones entre las variables [6].*

*Entrenar un modelo es el proceso de ajustar los parámetros de un modelo estadístico a un conjunto de datos de entrenamiento. Este proceso se utiliza para aprender las relaciones entre las variables y para mejorar la precisión del modelo.*

**Definición 3.** *(**Datos de validación**) Los datos de validación son un conjunto de datos que se utilizan para evaluar el rendimiento de un modelo estadístico que ya ha sido entrenado. Estos datos no se utilizan para entrenar el modelo, sino que se utilizan para proporcionar una estimación independiente del rendimiento del modelo en la población real [6].*

## 5.2. Regresión lineal

Los modelos de regresión lineal se emplean para predecir el valor de la variable independiente o para evaluar la relación funcional que tiene los predictores sobre ella y estudiar cuales pueden ser las causas de la variación de $\mathbf{Y}$. En general, el modelo de regresión lineal se sigue de la siguiente ecuación:

$$\mathbf{Y} = \beta_0 + \sum_{j=1}^{p} X_j \beta_j + \varepsilon, \tag{2}$$

donde, $\beta_0$ representa el intercepto y cada $\beta_j$ con $j = 1, ..., p$ representa

los parámetros desconocidos. Además $\varepsilon$ representa el error de observación debido a variables no controladas. En el modelo de regresión lineal, el error tiene una distribución normal $\varepsilon \sim \text{Normal}(0, \sigma^2)$. Esto significa que los errores son igualmente probables de ser positivos o negativos, y que la dispersión de los errores es la misma para todos los valores de la variable independiente [7].

Para realizar el ajuste del modelo lineal consideramos una muestra aleatoria de la variable $\mathbf{Y}$, con $n$ observaciones de la población $(x_1, y_1), ..., (x_n, y_n)$, donde cada $x_i = [x_{i1}, x_{i2}, ..., x_{ip}]^T$ con $i = 1, ..., n$ es un vector de predictores con $p$ características medidas en la cual $p$ es el número de predictores. Luego, el modelo toma la siguiente forma

$$y_i = \beta_0 + x_{i1}\beta_1 + ... + x_{ip}\beta_p + \varepsilon_i, \tag{3}$$

en que se desea estimar el vector de parámetros $\beta = [\beta_0, \beta_1, ..., \beta_p]^T$.

Por lo tanto la forma matricial esta dada por

$$\mathbf{Y} = \mathbf{X}\beta + \varepsilon \tag{4}$$

donde

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix} \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix} \quad \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

siendo $\mathbf{Y}$ el vector de observaciones $n$-dimensional, $\mathbf{X}$ la matriz de tamaño $n \times (p+1)$, donde cada fila representa a un individuo y cada columna a una covariable, siendo la primera columna de unos para incluir al intercepto, $\beta$ es un vector columna $(p+1 \times 1)$ y $\varepsilon$ el vector de errores $n$-dimensional. Con esta forma matricial nos indica que los datos están organizados en columnas, una para cada variable.

**Ejemplo 1.** *Tomando el modelo de regresión lineal* $\boldsymbol{Y} = \boldsymbol{X}\beta + \varepsilon$. *Una empresa de textiles, necesita evaluar la durabilidad de las telas, para ello le permite realizar una muestra pequeña de* 5 *telas de los cuales se mide la durabilidad que se obtiene (*$\boldsymbol{Y}$*) y dos características básicas, elasticidad y solidez del color (*$\boldsymbol{X}$*).*

$$\boldsymbol{Y} = \begin{pmatrix} 7.109 \\ 4.523 \\ 3.291 \\ 5.081 \\ 7.545 \end{pmatrix}, \quad \boldsymbol{X} = \begin{pmatrix} 1 & 2.9 \\ 1 & 3.1 \\ 1 & 3 \\ 1 & 3 \\ 1 & 1.8 \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

*Luego se hallan* $\boldsymbol{X}^T$ *que nos da lo siguiente*

$$\boldsymbol{X}^T = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 2.9 & 3.1 & 3 & 3 & 1.8 \end{pmatrix}$$

*Así la matriz* $\boldsymbol{X}^T\boldsymbol{X}$ *y su inversa esta dada por:*

$$\boldsymbol{X}^T\boldsymbol{X} = \begin{pmatrix} 5 & 13.8 \\ 13.8 & 39.26 \end{pmatrix}$$

$$(\boldsymbol{X}^T\boldsymbol{X})^{-1} = \begin{pmatrix} 6.69 & -2.35 \\ -2.35 & 0.85 \end{pmatrix}$$

*De lo anterior tenemos la siguiente estimación:*

$$\hat{\beta} = \begin{pmatrix} 11.87 \\ -2.30 \end{pmatrix}$$

Para realizar un análisis de regresión se asumen unas hipótesis sobre el modelo:

1. Cada predictor tiene que estar linealmente relacionado con la variable respuesta **Y** mientras los demás predictores se mantienen constantes.

2. El valor esperado de los errores es igual a cero, $E(\varepsilon) = 0$.

3. La varianza de los errores $\sigma^2$ es constante y la covarianza de entre los errores $\varepsilon_i\varepsilon_j$, con $i$ es cero. Esta hipótesis indica que cada elemento de la diagonal principal serán iguales a $\sigma^2$ y que cada elemento fuera de esta serán iguales a cero.

$$\boldsymbol{\varepsilon\varepsilon}^\top = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix} \begin{bmatrix} \varepsilon_1 & \varepsilon_2 & \dots & \varepsilon_n \end{bmatrix} = \begin{bmatrix} \varepsilon_1^2 & \varepsilon_1\varepsilon_2 & \dots & \varepsilon_1\varepsilon_n \\ \varepsilon_2\varepsilon_1 & \varepsilon_2^2 & \dots & \varepsilon_2\varepsilon_n \\ \vdots & \vdots & \dots & \vdots \\ \varepsilon_n\varepsilon_1 & \varepsilon_n\varepsilon_2 & \dots & \varepsilon_n^2 \end{bmatrix}$$

entonces,

$$E\left(\boldsymbol{\varepsilon}\varepsilon^{\top}\right) = E\begin{bmatrix} \varepsilon_1^2 & \varepsilon_1\varepsilon_2 & \dots & \varepsilon_1\varepsilon_n \\ \varepsilon_2\varepsilon_1 & \varepsilon_2^2 & \dots & \varepsilon_2\varepsilon_n \\ \vdots & \vdots & \dots & \vdots \\ \varepsilon_n\varepsilon_1 & \varepsilon_n\varepsilon_2 & \dots & \varepsilon_n^2 \end{bmatrix} =$$

$$\begin{bmatrix} E\left(\varepsilon_1^2\right) & E\left(\varepsilon_1\varepsilon_2\right) & \dots & E\left(\varepsilon_1\varepsilon_n\right) \\ E\left(\varepsilon_2\varepsilon_1\right) & E\left(\varepsilon_2^2\right) & \dots & E\left(\varepsilon_2\varepsilon_n\right) \\ \vdots & \vdots & \dots & \vdots \\ E\left(\varepsilon_n\varepsilon_1\right) & E\left(\varepsilon_n\varepsilon_2\right) & \dots & E\left(\varepsilon_n^2\right) \end{bmatrix}$$

De acuerdo con la hipótesis, para todo $i \neq j$, $E(\varepsilon_i \times \varepsilon_j) = 0$, tenemos que:

$$\begin{aligned} E\left(\boldsymbol{\varepsilon}\varepsilon^{\top}\right) &= \begin{bmatrix} \sigma^2 & 0 & \cdots & 0 \\ 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2 \end{bmatrix} \\ E\left(\boldsymbol{\varepsilon}\varepsilon^{\top}\right) &= \sigma^2 \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \\ E\left(\boldsymbol{\varepsilon}\varepsilon^{\top}\right) &= \sigma^2 \boldsymbol{I}_n \end{aligned} \tag{5}$$

La matriz $E(\varepsilon\varepsilon^T)$ es conocida como la matriz de varianza y covarianzas, en donde $\mathbf{I}_n$ es la matriz identidad $n$-dimensional. Luego la ecuación 5 nos indica la matriz de varianzas y covarianzas de los errores es proporcional a la matriz

identidad $\mathbf{I}_n$.

4. Los errores se observan como variables aleatorias distribuidas de forma normal con media cero y varianza $\sigma^2\mathbf{I}_n$, $\varepsilon \sim N(0, \sigma^2\mathbf{I}_n)$.

**Error Cuadrático Medio (ECM)**

Cuando se estima el vector de parámetros $\hat{\beta}$, el objetivo es predecir nuevos valores para la variable de respuesta y así poder evaluar su nivel de ajuste. En el análisis de regresión, una medida ampliamente utilizada para medir esto es conocida como error cuadrático medio (ECM), que se define como

$$\text{ECM} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i), \tag{6}$$

donde $\hat{y}_i$ es es estimador del modelo y $y_i$ es el dato i-ésimo. De manera equivalente, se calcula el ECM relacionando la varianza y el sesgo de un estimador de la siguiente forma

$$\text{ECM}[\hat{\theta}] = \text{Var}[\hat{\theta}] + [\text{Sesgo}[\hat{\theta}, \theta]]^2 \tag{7}$$

Se observa que como $\hat{\beta}$ es insesgado se tiene que $\text{ECM}[\hat{\beta}] = \text{Var}[\hat{\beta}]$

**Método de mínimo cuadrados**

Hay muchos métodos diferentes, pero el más popular con diferencia es el método de los mínimos cuadrados. En este método, se escoge los coeficientes $\beta$ para

minimizar la suma residual de cuadrados

$$\text{RSS}(\beta) = \sum_{i=1}^{N}(y_i - f(x_i))^2 = \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2. \tag{8}$$

El criterio RSS será razonable si las observaciones $(\mathbf{x}_i, y_i)$ de la muestra son extraídas aleatoriamente e independientes de su población. Las muestras son extraídas aleatoriamente e independientes para garantizar que el criterio RSS sea una estimación precisa del error del modelo en la población real. Si las muestras no fueran aleatorias o independientes, entonces el criterio RSS podría estar sesgado y no proporcionar una estimación precisa del error del modelo [4].

Si esto no se cumpliera, el criterio todavía es válido si los $y_i$ son condicionalmente independientes dados los $\mathbf{x}_i$. Denotemos por $\mathbf{X}$ la matriz $N \times (p+1)$ con cada fila un vector de entrada (con un 1 en la primera posición), y de forma similar se deja que $\mathbf{y}$ sea el vector $N$ de salidas en el conjunto de entrenamiento. Entonces se logra escribir la suma de cuadrados residual en forma matricial

$$\text{RSS}(\beta) = (\mathbf{Y} - \mathbf{X}\beta)^T(\mathbf{Y} - \mathbf{X}\beta), \tag{9}$$

donde $\text{RSS}(\beta)$ es una función cuadrática y por lo tanto siempre tiene mínimo, sin embargo puede no ser único.

Diferenciando con respecto a $\beta$ (29) obtenemos lo siguiente

$$\begin{aligned} \frac{\partial \text{RSS}}{\partial \beta} &= -2\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\beta) \\ \frac{\partial^2 \text{RSS}}{\partial \beta \partial \beta^T} &= 2\mathbf{X}^T\mathbf{X}. \end{aligned} \tag{10}$$

Suponiendo que $\mathbf{X}$ tiene rango de columna completo, y por lo tanto $\mathbf{X}^T\mathbf{X}$ es definida positiva, fijamos la primera derivada a cero

$$\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\beta) = 0 \tag{11}$$

para obtener la solución única

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}. \tag{12}$$

Donde,

$$\boldsymbol{\beta} = \begin{bmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ \vdots \\ \hat{\beta}_p \end{bmatrix} \quad \boldsymbol{X}^T\boldsymbol{X} = \begin{bmatrix} \sum_{i=1}^{n} x_{i1}^2 & \sum_{i=1}^{n} x_{i1}x_{i2} & \cdots & \sum_{i=1}^{n} x_{i1}x_{ip} \\ \sum_{i=1}^{n} x_{i2}x_{i1} & \sum_{i=1}^{n} x_{i2}^2 & \cdots & \sum_{i=1}^{n} x_{i2}x_{np} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} x_{ip}x_{i1} & \sum_{i=1}^{n} x_{np}x_{i2} & \cdots & \sum_{i=1}^{n} x_{ip}^2 \end{bmatrix} \boldsymbol{X}^T\boldsymbol{Y} = \begin{bmatrix} \sum_{i=1}^{n} x_{i1}y_i \\ \sum_{i=1}^{n} x_{i2}y_i \\ \vdots \\ \sum_{i=1}^{n} x_{ip}y_i \end{bmatrix}$$

Cada valor estimado $\beta_i$ indica la variación que experimenta la variable dependiente cuando la variable independiente $\mathbf{x}_i$ varia en una unidad y todas las demás permanecen constantes. Por lo consiguiente, los valores ajustados son

$$\mathbf{Y} = \mathbf{X}\hat{\beta} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}, \tag{13}$$

Para que el estimador $\hat{\beta}$ este bien definido, la matriz $\mathbf{X}^T\mathbf{X}$ debe ser no singular.

**Sumas de cuadrados**

Dado que las observaciones $y_i$ se suponen no correlacionadas con varianza constante $\sigma^2$, y los valores de $\mathbf{x_i}$ son fijos, podemos establecer las siguientes condiciones:

- **Suma de cuadrados totales**

  La suma de cuadrados totales (SCT) es igual a la suma de los cuadrados explicados por la regresión (SCR) más la suma de los errores al cuadrado, es decir:

$$\text{SCT} = \sum_{i=1}^{n}(y_i - \bar{y})^2 = \sum_{i=1}^{n}(\hat{y}_i - \bar{y}) + \sum_{i=1}^{n} e_2^i \tag{14}$$

  donde, los $e_i \in \mathbf{e}$ el cual es el vector de los errores estimados de $\varepsilon$ $(n \times 1)$.

- **Varianza de los errores estimados**

La varianza de los errores estimados ($\varepsilon$) esta definido como:

$$\sigma^2 = \frac{\sum_{i=1}^{n}(\varepsilon_i)^2}{n-k} = \frac{\varepsilon^T\varepsilon}{n-k} \tag{15}$$

donde $e = \hat{y}_i - y_i$, $n$ representa la cantidad de observaciones en la muestra y $k$ es el número de predictores, incluyendo al término constante, que se denota como intercepto. Es decir, específicamente tenemos que $k = p + 1$. La varianza de los errores da información acerca de la precisión de los estimadores, mas no da información acerca de cuan precisos son los estimadores para una muestra en particular. Si se obtiene varianzas pequeñas, sera un indicador que en promedio, los coeficientes son precisos.

**Multicolinealidad**

El problema de multicolinealidad es uno de los mayores inconvenientes que se puede encontrar en un análisis de regresión. Este problema consiste en que al menos una covariable del modelo es combinación lineal de las otras covariables. Esto provoca que no se pueda saber que covariables realmente predicen la variable de respuesta y en consecuencia, no se podría identificar con precisión los estimadores en el modelo de regresión lineal.

Según Montgomery, Peck y Vining en [8] las principales causas del problema

de multicolinealidad son:

- Por la relación casual entre covariables del modelo. Por ejemplo, la edad y la experiencia suelen presentar alta relación ya que ambas evolucionan conjuntamente.
- Cuando se adiciona un termino polinomial al modelo, produciendo un deterioro en la matriz $\mathbf{X}^T\mathbf{X}$, y si el rango de $\mathbf{X}$ es pequeño, al agregar un termino $\mathbf{X}^2$, puede generar multicolinealidad.
- Por la presencia de una mayor cantidad de covariables frente a pocas observaciones disponibles.

Existen dos tipos de multicolinealidad las cuales son:

- **Multicolinealidad perfecta :** Se refiere a la existencia de una relación lineal exacta entre dos o más variables independientes. Por consiguiente, se viola una de las hipótesis del modelo: la matriz $\mathbf{X}$ no es de rango completo por columnas, esto es, $rank(\mathbf{X}) < k$. En consecuencia, el incumplimiento de esta hipótesis no permite invertir la matriz $\mathbf{X}^T\mathbf{X}$, por lo que el sistema normal $\mathbf{X}^T\mathbf{X}\beta = \mathbf{X}^T\mathbf{Y}$ es compatible indeterminado, es decir, no es posible obtener una solución única para $\beta$, ya que existen infinitas soluciones. Por lo tanto no sera posible estimar los coeficientes de las variables, sin embargo, si es posible estimar una combinación lineal de los mismos.
- **Multicolinealidad aproximada:** Se refiere a la existencia de una relación lineal aproximada entre dos o más covariables. En este caso no se viola la hipótesis de que la matriz $\mathbf{X}$ sea de rango completo por columnas ($rank(\mathbf{X}) =$

$k$), por lo que si sera posible invertir $\mathbf{X}^T\mathbf{X}$ y obtener los estimadores por mínimos cuadrados. Sin embargo,el determinante de $\mathbf{X}^T\mathbf{X}$ sera muy próximo a cero, por lo que $(\mathbf{X}^T\mathbf{X})^{-1}$ tendera a tomar valores altos. En consecuencia, la presencia de multicolinealidad no perfecta ocasiona que:

- Las varianzas de los estimadores sean muy grandes.
- Los coeficientes estimados sera muy sensibles ante pequeños cambios en los datos.
- Se tenga un coeficiente de determinación elevado.

Montgomery, Peck y Vining en [8] mencionan varias soluciones para el problema que tenemos de la multicolinealidad, las cuales son:

- Realizar selección de variables: eliminar las variables que son posibles causantes de multicolinealidad.
- Mejorar el diseño muestral extrayendo la información máxima de las variables observadas.

**Métodos de regularización**

Los métodos de regularización son una herramienta en el aprendizaje automático para evitar el sobreajuste y mejorar la capacidad de generalización de los modelos. Estos métodos consiste en ajustar el modelo utilizando todos los $p$ predictores, mientras se reducen a cero los coeficientes estimados con respecto a las estimaciones

de mínimos cuadrados. Estos métodos también se conocen como métodos de contracción y tienen el efecto de disminuir la varianza al establecer algunos coeficientes exactamente iguales a cero, por lo que también son útiles para seleccionar variables [7].

- **Ridge:** Este método aproxima a cero los coeficientes de los predictores pero sin llegar a excluir ningún predictor. Estos coeficientes son similares a los de la regresión lineal, pero con la penalización L2 la cual reduce su magnitud y los hace más robustos al sobreajuste. Esta interpretación de los coeficientes es la relación entre las variables independientes y la variable dependiente. Aunque la interpretación debe hacerse con cuidado, puesto que los coeficientes no son independientes entre si.

- **Lasso:** Este método aproxima a cero los coeficientes, llegando a excluir predictores. Estos coeficientes son más dispersos que los de la regresión lineal, la penalización L1 induce que algunos de los coeficientes sean exactamente cero. La regresión Lasso realiza selección de variables y puede ser útil para identificar las variables más importantes para la predicción.

Estos métodos son especialmente creados para situaciones en las que el número de variables $p$ es similar o incluso mayor al número de observaciones $n$. Además, se aplican los métodos de regularización para obtener estimaciones de los coeficientes de regresión cuando la matriz $\mathbf{X}^T\mathbf{X}$ no tiene inversa. En este contexto, regularizar implica hacer el problema manejable mediante la imposición de restricciones a las soluciones posibles. Para Ridge se tiene la penalización de L2 la cual sanciona la suma de los cuadrados de los coeficientes y Lasso utiliza la penalización L1 que penaliza

la suma de los valores absolutos de los coeficientes. En general, la formulación de los métodos de regularización en el contexto de modelos lineales es muy similar a los mínimos cuadrados y esta dado de la siguiente manera:

$$\min_{\beta_0,\beta}\left\{\sum_{i=1}^{n}(y_i-\beta_0-\sum_{j=1}^{p}x_{ij}\beta_j)^2+\lambda\phi(\beta)\right\} \tag{16}$$

donde $\lambda \geq 0$ y $\phi(\beta)$ es la función creciente de penalización que depende de $\lambda$. Una familia de funciones de penalización bastante utilizada es la correspondiente a la norma $L_q$ la cual esta definida de la siguiente manera.

**Definición 4.** *(Norma $L_q$ ). Una clase útil de normas de vectores es la norma $L_q$, definido por [9].:*

$$\|\boldsymbol{x}\|_q = \left(|x_1|^q + \cdots + |x_n|^q\right)^{1/q}, \quad q \leq 1$$

En esta clase de norma los casos mas importantes son:

- Para $q = 1$ , $\|\boldsymbol{x}\|_1 = |x_1| + \cdots + |x_n|$, denominada norma de la suma.
- Para $q = 2$ $\|\boldsymbol{x}\|_2 = \left(|x_1|^2 + \cdots + |x_n|^2\right)^{1/2} = \left(\boldsymbol{x}^\top\boldsymbol{x}\right)^{1/2}$, denominada norma euclidiana.
- Para $q = \infty$ $\|\boldsymbol{x}\|_\infty = \max_{1\leq i\leq n}|x_i|$, denominada norma del máximo.

De lo anterior tenemos que $\phi(\beta)$ esta dado por:

$$\phi(\beta) = (\|\beta\|_q)^q = \sum_{j=1}^{p}|\beta_j|^q, \quad q > 0 \tag{17}$$

Es crucial tener en cuenta que la magnitud de los coeficientes de regresión depende de la escala en que se mida cada predictor, por lo cual durante el proceso de selección de variables y regularización para el proceso de selección de variables y regularización es importante que todas estén en la misma escala para evitar alguna alteración en los coeficientes estimados. Con esto se asegura así una evaluación más precisa y acorde del modelo.

## 5.3. Regresión Ridge

La técnica de estimación Ridge es una solución al problema de la multicolinealidad. En este caso, se modifica la matriz (9) del modelo agregando una penalización para evitar las consecuencias que resultan del mal estado condicionado. La optimización de la siguiente expresión fue propuesta por Hoerl y Kennard [10], cuya solución es efectivamente el estimador Ridge el cual se toma como función de perdida:

$$L(\beta) = (\mathbf{Y} - \mathbf{X}\beta)^T(\mathbf{Y} - \mathbf{X}\beta) + \lambda(\beta^T\beta) \tag{18}$$

Note que en la parte izquierda se está optimizando para obtener el estimador por mínimos cuadrados $\beta$. Ahora, para esta misma expresión se agrega un término adicional $\lambda(\beta^T\beta)$ que penaliza la norma al cuadrado del vector de parámetros mediante el hiperparámetro positivo $\lambda > 0$, lo que generará la regularización. Donde la función de penalización es la correspondiente a la norma $\mathrm{L}^2$, $\|\beta\|_2^2 = \sum_{j=1}^{p} |\beta_j|^2 = \beta^T\beta$. Cabe destacar que si tomamos $\lambda = 0$, entonces no se tiene ningún efecto alguno sobre este proceso.

A medida que $\lambda \to \infty$, el impacto de la penalización aumenta y las estimaciones del coeficiente de regresión Ridge se acercaran a cero. La selección de un valor adecuado para $\lambda$ es crucial, ya que en entornos de aprendizaje automático esta estimación se realiza normalmente mediante validación cruzada.

**Teorema 5.1.** *Bajo el modelo de regresión lineal, la función* $f(\beta) = (\boldsymbol{Y} - \boldsymbol{X}\beta)^T(\boldsymbol{Y} - \boldsymbol{X}\beta) + \lambda(\beta^T\beta)$ *es minimizada por* $\beta = \hat{\beta}_R = (\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_{p\times p})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$.

*Demostración.* Por definición notemos lo siguiente

$$f(\beta) = (\mathbf{Y} - \mathbf{X}\beta)^T(\mathbf{Y} - \mathbf{X}\beta) + \lambda(\beta^T\beta)$$

$$= (\mathbf{Y}^T - (\mathbf{X}\beta)^T)(\mathbf{Y} - \mathbf{X}\beta) + \lambda(\beta^T\beta)$$

$$= (\mathbf{Y}^T - \beta^T\mathbf{X}^T)(\mathbf{Y} - \mathbf{X}\beta) + \lambda(\beta^T\beta)$$

$$= \mathbf{Y}^T\mathbf{Y} - \mathbf{Y}^T\mathbf{X}\beta - \beta^T\mathbf{X}^T\mathbf{Y} + \beta^T\mathbf{X}^T\mathbf{X}\beta + \lambda(\beta^T\beta)$$

$$= \mathbf{Y}^T\mathbf{Y} - \mathbf{Y}^T\mathbf{X}\beta - \beta^T\mathbf{X}^T\mathbf{Y} + \beta^T\mathbf{X}^T\mathbf{X}\beta + \lambda(\beta^T\beta)$$

Ahora derivando matricialmente (29) con respecto a $\beta$ se tiene que:

$$\frac{\partial}{\partial\beta} f(\beta) = -2\mathbf{X}^T\mathbf{Y} + 2\mathbf{X}^T\mathbf{X}\beta + 2\lambda\beta,$$

la expresión anterior se iguala a cero y se tiene lo siguiente:

$$-2\mathbf{X}^T\mathbf{Y} + 2\mathbf{X}^T\mathbf{X}\beta + 2\lambda\beta = 0$$

$$2(-\mathbf{X}^T\mathbf{Y} + \mathbf{X}^T\mathbf{X}\beta + \lambda\beta) = 0$$

$$(-\mathbf{X}^T\mathbf{Y} + \mathbf{X}^T\mathbf{X}\beta + \lambda\beta) = 0$$

$$\mathbf{X}^T\mathbf{X}\beta + \lambda\beta = \mathbf{X}^T\mathbf{Y}$$

$$(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_{p\times p})\beta = \mathbf{X}^T\mathbf{Y}$$

donde $\mathbf{I}_{p\times p}$ es la matriz identidad de dimensión $p\times p$, note que $\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I}_{p\times p}$ es definida positiva y por la no-multicolinealidad se tiene que $\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I}_{p\times p}$ es invertible. Por lo tanto, se concluye que

$$\beta = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_{p\times p})^{-1}\mathbf{X}^T\mathbf{Y}$$

□

La solución depende de $\lambda$, y el valor óptimo se encuentra con validación cruzada. La ventaja de la regresión Ridge sobre los mínimos cuadrados tiene su origen en la compensación de sesgo-varianza. A medida que $\lambda$ aumenta, la varianza de la regresión Ridge disminuye, pero el sesgo aumenta.

## 5.4. Regresión Lasso

La regresión Ridge tiene una desventaja al no producir una menor varianza, ya que la penalización L2 hace tender los coeficiente de la regresión a cero. Sin

embargo, esto no hará que sean exactamente cero. Esto puede no sea un problema para la precisión de la predicción, pero puede crear un desafió en la interpretación del modelo en un entorno en el que numero de variables $p$ es bastante grande.

Para superar dicha desventaja, Tibshirani en [11] sugirió el uso de un estimador alternativo. Este estimador es conocido como Lasso, el cual a diferencia del estimador Ridge añade una penalización L1 a la suma de los errores al cuadrado. Se toma como función de perdida:

$$L(\beta) = (\mathbf{Y} - \mathbf{X}\beta)^T(\mathbf{Y} - \mathbf{X}\beta) + \lambda||\beta||_1 \tag{19}$$

De forma similar al estimador Ridge, el hiperparametro $\lambda > 0$ se encargará de generar la regularización, adicionalmente si $\lambda = 0$ no habrá penalización y se tiene el estimador de mínimos cuadrados. También de manera general el hiperparametro $\lambda$ en el caso de Lasso se calcula mediante validación cruzada.

Donde $||\beta||_1$ es la penalización correspondiente a la norma L1 se puede reescribir de la siguiente manera $||\beta||_1 = \sum_{j=1}^{p} |\beta_j|$. La norma L1 es una función de distancia que mide la distancia entre un vector y el origen. Se define como la suma de los valores absolutos de los componentes del vector. La cual se utiliza en la regresión Lasso para seleccionar variables. La regresión Lasso minimiza la suma de los residuos cuadrados y la norma L1 de los coeficientes de regresión.

Podemos ver que la función (19) es convexa debido a que es una suma de

funciones convexas, como ocurre con la función (18). No obstante, $||\beta||_1$ no es estrictamente convexa y por consiguiente no se asegura unicidad en el estimador Lasso Efron, Hastie, Johnstone et al. [12]. Un efecto de esto es que en general no existe una solución analítica para Lasso.

**Teorema:** Bajo el modelo de regresión lineal usual y la ortogonalidad de la matriz $\mathbf{X}$ el $i$-esimo estimador Lasso $\hat{\beta}_{L_i}$ viene dado por $\hat{\beta}_{L_i} = sgn(\hat{\beta}_i)(|\hat{\beta}_i| - \gamma)^+$.

**Demostración:** Por definición se obtiene lo siguiente

$$f(\beta) = (\mathbf{Y} - \mathbf{X}\beta)^T(\mathbf{Y} - \mathbf{X}\beta) + \lambda||\beta||_1$$

$$= (\mathbf{Y}^T - (\mathbf{X}\beta)^T)(\mathbf{Y} - \mathbf{X}\beta) + \lambda||\beta||_1$$

$$= (\mathbf{Y}^T - \beta^T\mathbf{X}^T)(\mathbf{Y} - \mathbf{X}\beta) + \lambda||\beta||_1$$

$$= \mathbf{Y}^T\mathbf{Y} - \mathbf{Y}^T\mathbf{X}\beta - \beta^T\mathbf{X}^T\mathbf{Y} + \beta^T\mathbf{X}^T\mathbf{X}\beta + \lambda||\beta||_1$$

$$= \mathbf{Y}^T\mathbf{Y} - \hat{\beta}^T\beta - \beta^T\hat{\beta} + \beta^T\beta + \lambda||\beta||_1$$

$$= \mathbf{Y}^T\mathbf{Y} - 2\hat{\beta}^T\beta + \beta^T\beta + \lambda||\beta||_1$$

Considere los siguientes casos para optimizar la función objetivo, minimizando por separado su $i$-esimo elemento.

Si $\hat{\beta}_i > 0$ entonces $\beta_i \geq 0$. Derivando con respecto a $\beta_i$ se tiene que

$$\frac{\partial}{\partial \beta_i}(y_i^2 - 2\hat{\beta}_i\beta_i + \beta_i^2 + \lambda\beta_i) = -2\hat{\beta}_i + 2\beta_i + \lambda.$$

Igualando a cero se logra lo siguiente:

$$\begin{aligned} -2\hat{\beta}_i + 2\beta_i + \lambda &= 0 \\ \beta_i &= \hat{\beta}_i - \frac{\lambda}{2} \\ \beta_i &= \hat{\beta}_i - \gamma, \quad \gamma = \frac{\lambda}{2} \\ \beta_i &= \hat{\beta}_i - \gamma \\ \beta_i &= sgn(\hat{\beta}_i)(|\hat{\beta}_i| - \gamma)^+ \end{aligned}$$

Ahora análogamente si $\hat{\beta}_i < 0$, entonces $\beta_i < 0$ tenemos lo siguiente

$$\beta_i = (\hat{\beta}_i + \gamma)^- = -(-\hat{\beta} - \gamma)^+ = sgn(\hat{\beta})(|\hat{\beta}| - \gamma)^+$$

□

En el anterior teorema se utiliza la hipótesis que nuestra matriz $\mathbf{X}$ es ortogonal, esto se requiere para asegurar que las estimaciones de $\beta_i$ son independientes, además $+$ denota que se toma solo lo positivo. La ortogonalidad asegura que $(\mathbf{X}^T\mathbf{X})^{-1}$ es diagonal, por lo tanto $Cov(\hat{\beta}_i, \hat{\beta}_j) = 0$ para $i \neq j$, de manera que la estimación de $\beta_i$ no cambiara dependiendo de si $\beta_j$ esta o no en el modelo [11].

Ahora minimizando las funciones (18) (19) resulta equivalente a los dos problemas siguientes

$$\underset{\beta \in \mathbb{R}^p}{\arg\min}((\mathbf{Y}-\mathbf{X}\beta)^T(\mathbf{Y}-\mathbf{X}\beta)), \quad \text{sujeto a } \|\beta\|_2^2 \leq s, \tag{20}$$

y,

$$\underset{\beta \in \mathbb{R}^p}{\arg\min}((\mathbf{Y}-\mathbf{X}\beta)^T(\mathbf{Y}-\mathbf{X}\beta)), \quad \text{sujeto a } \|\beta\|_1 \leq s, \tag{21}$$

respectivamente, donde $||\beta||_2^2 = \beta^T\beta$, $||\beta||_1 = \sum_{j=1}^{p} |\beta_j|$ y arg mín es una abreviatura de argumento del mínimo. Es el valor o vector de valores de un parámetro o vector de parámetros que minimiza una función. En otras palabras, para cada valor de $\lambda$, existe algún $s$ tal que (18) y (20) permitirá obtener las mismas estimaciones de Ridge. Del mismo modo, para cada valor de $\lambda$ existe algún $s$ tal que (19) y (21) permitirá obtener la misma estimación Lasso, este valor $s$ en lenguaje `Python` se toma de 0.001. Note que cuando $p = 2$, entonces (20) indica que el estimador Ridge posee la menor suma de los residuos al cuadrado (RSS) de todos los puntos dentro del circulo definido por $\beta_1^2 + \beta_2^2 \leq s$. De igual forma, el estimador Lasso tiene el menor RSS de todos los puntos dentro del diamante definido por $|\beta_1| + |\beta_2| \leq s$.

Cuando se utiliza la regresión Ridge por lo general se desea encontrar un estimador que posea el menor (RSS), sujeto a la restricción de que existe un limite $s$ para lo grande que puede se $||\beta||_2^2$. Cuando $s$ es muy grande entonces este limite no es muy restrictivo, por lo que el estimador Ridge puede ser grande. De hecho, si $s$ es lo suficientemente grande como para que la solución de mínimos cuadrados este dentro del limite, entonces (20) simplemente obtendrá la solución de mínimos

cuadrados. Por el contrario, si $s$ es pequeño, entonces $||\beta||_2^2$ debe ser pequeña para no sobrepasar el limite. De igual manera para (21) indica que cuando se aplica la regresión Lasso, se busca el estimador tal que RSS sea lo mas pequeño posible, sujeto a $||\beta||_1$ no exceda el limite $s$.

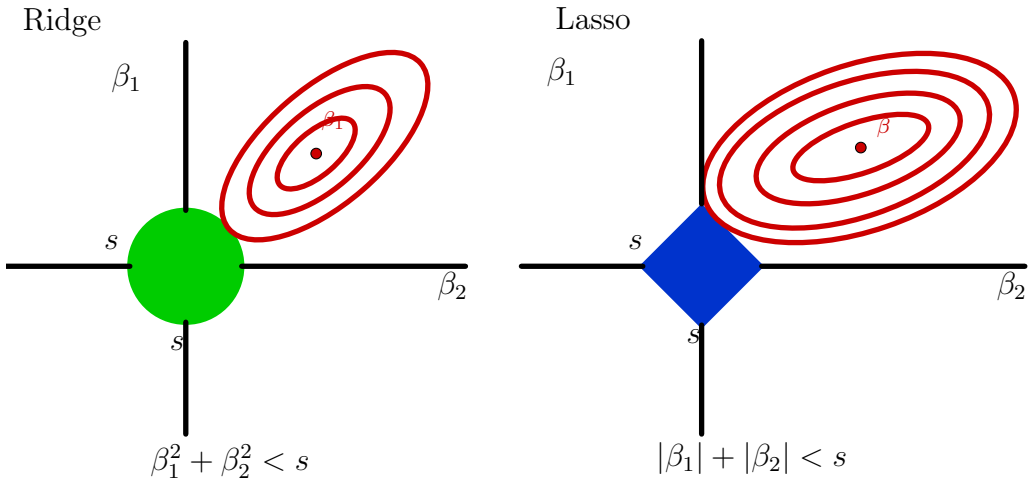


Figura 1: Representación geométrica de los estimadores Ridge y Lasso.
Fuente: Elaboración propia.

Por consiguiente, se puede notar que geométricamente para alguna constante $s$ los coeficientes de Lasso representan el punto de corte del diamante y las elipses. En el primer gráfico de la figura 1 se evidencia que si el punto de corte esta en uno de los vértices del diamante, entonces el coeficiente estimado por Lasso es cero. Por otro lado, se observa que la restricción de Ridge representa las curvas de nivel con forma de circulo, lo que implica que no existe posibilidad de que algún coeficiente corte con la elipse en un vértice, por tal motivo los coeficientes de Ridge no pueden

ser nulos.

La elección del hiperparametro $\lambda$ se realiza en la sección 6.

**Elección del parámetro $\lambda$**

Como puede observarse todas estas técnicas de mínimos cuadrados penalizados dependen de un parámetro de penalización $\lambda$, que controla la importancia dada a la penalización en el proceso de optimización. Cuanto mayor es $\lambda$ mayor es la penalización en los coeficientes de regresión y más son contraídos éstos hacia cero. La elección de éste parámetro involucra un balance entre los componentes de sesgo y varianza del ECM al estimar $\beta$.

Un método más automático, pero intensivo computacionalmente, consiste en estimar $\lambda$ mediante validación cruzada (en general se recomiendan utilizar ambos métodos y comparar resultados). El método de validación cruzada consiste en dividir el modelo en un conjuntos de entrenamiento para ajustar un modelo y un conjunto de prueba para evaluar su capacidad predictiva, mediante el error de predicción u otra medida. La forma en que se aplica la validación cruzada es mediante la división del conjunto de datos disponibles de manera aleatoria en $k$ subconjuntos o pliegues de igual tamaño y mutuamente excluyentes.

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 se evalúa a partir de $K$ combinaciones de datos de entrenamiento y de prueba, pero aun así tiene una desventaja, y es que es lento desde el punto de vista computacional. La validación cruzada con $k = 10$ es una de las más utilizadas, pero hay que tener en cuenta el número de observaciones del que se dispone. El valor del parámetro será el que de el mínimo error. A continuación, se ilustra el algoritmo que se lleva a cabo para obtener ese valor del parámetro, explicado anteriormente:

1. Se divide el conjunto de datos D en K subconjuntos de igual tamaño (partición) $D_1, \ldots, D_K$. Generalmente, K toma los valores 5 o 10.

2. Para cada $k = 1, \ldots, K$

   - Se ajusta el modelo $\hat{y}_k^{(\lambda)}(z)$ con el conjunto de entrenamiento $D - D_k$ (exclusión del k-ésimo pliegue).
   - Se calcula el error por validación cruzada,

$$(\text{CVError})_k^{(\lambda)} = \frac{1}{|D_k|} \sum_{z \in D_k} \left[ y - \hat{y}_k^{(\lambda)}(z) \right]^2$$

3. Criterio: minimizar el error global de validación cruzada:

$$\lambda^* = \arg\min_{\lambda} (\text{ CVError })^{(\lambda)} = \arg\min_{\lambda} \left[ \frac{1}{K} \sum_{k=1}^{K} (\text{ CVError })_k^{(\lambda)} \right]$$

Donde $D$ es el conjunto de datos y $K$ son subconjuntos de igual tamaño estos subconjuntos representa una partición diferente de los datos. Note que $\hat{y}_k^{(\lambda)}$ es la predicción del modelo con parámetro de regularización $\lambda$ calculada en el conjunto de datos $D_K$ excluyendo la observación $z$ en el proceso de validación cruzada. Finalmente, este algoritmo realiza una validación cruzada de $K$ iteraciones así ajustando el modelo con diferentes particiones del conjunto de datos, para ello se calcula los errores de validación cruzada para cada partición y eligiendo el valor ideal de $\lambda$ que minimiza el error global de validación.

El parámetro $\lambda$ es el encargado de la intensidad de la penalización que se aplica a los coeficientes de la regresión, además se le agrega a la matriz para que esta sea invertible. Cuanto mayor sea el valor de $\lambda$, mayor será la penalización y menor será la magnitud de los coeficientes. Esto nos indica que este parámetro interpreta los coeficientes dependiendo del valor que se da según el conjunto de datos.

**Ejemplo 2.** *Se mantiene el mismo problema del ejemplo* 1 *pero ahora se considera el modelo de regresión Ridge con un* $\lambda = 0.5$ *y además*

$$\boldsymbol{Y} = \begin{pmatrix} 7.109 \\ 4.523 \\ 3.291 \\ 5.081 \\ 7.545 \end{pmatrix}, \quad \boldsymbol{X} = \begin{pmatrix} 1 & 2.9 \\ 1 & 3.1 \\ 1 & 3 \\ 1 & 3 \\ 1 & 1.8 \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

*Luego se calcula* $\boldsymbol{X}^T$ *que nos da lo siguiente*

$$\boldsymbol{X}^T = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 2.9 & 3.1 & 3 & 3 & 1.8 \end{pmatrix}$$

*Así la matriz* $\boldsymbol{X}^T\boldsymbol{X}$ *y su inversa esta dada por:*

$$\boldsymbol{X}^T\boldsymbol{X} = \begin{pmatrix} 5 & 13.8 \\ 13.8 & 39.26 \end{pmatrix}$$

*Se tiene que* $\lambda = 0.5$*, luego se obtiene*

$$\lambda \boldsymbol{I}_{p\times p} = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}$$

*De lo anterior, tenemos que*

$$(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_{p\times p}) = \begin{pmatrix} 5.5 & 13.8 \\ 13.8 & 39.7 \end{pmatrix}$$

$$(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_{p\times p})^{-1} = \begin{pmatrix} 1.4 & -0.4 \\ -0.4 & 0.1 \end{pmatrix}$$

$$\boldsymbol{X}^T\boldsymbol{Y} = \begin{pmatrix} 27.54 \\ 73.33 \end{pmatrix}$$

*Por lo tanto, la siguiente estimación:*

$$\hat{\beta} = \begin{pmatrix} 2.95 \\ 0.82 \end{pmatrix}$$

**Ejemplo 3.** *Una carpintería desea realizar un contrato, donde se necesita evaluar el volumen de la madera, para ello se hace una muestra de 5 arboles de los cuales se mide el volumen de madera que se obtiene de cada árbol (**Y**) y tres características básicas del tronco de cada árbol que son el diámetro del tronco medido a dos alturas diferentes (**X**). Para este ejemplo se considera el modelo de regresión Ridge pero se calcula el valor de lambda con el algoritmo anterior de validación cruzada, por lo tanto* $\lambda = 0.76$ *para hallar el lambda se apoya del lenguaje* `python` *y además*

$$\boldsymbol{Y} = \begin{pmatrix} 1.3 \\ -0.5 \\ 2.6 \\ 0.9 \\ 7.545 \end{pmatrix}, \quad \boldsymbol{X} = \begin{pmatrix} 1 & -1 & 2 \\ 1 & 0 & 1 \\ 1 & 2 & -1 \\ 1 & 1 & 0 \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix}$$

*Primero se calcula* $\boldsymbol{X}^T$ *por lo tanto*

$$\boldsymbol{X}^T = \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 2 & 1 \\ 2 & 1 & -1 & 0 \end{pmatrix}$$

*Así la matriz* $\boldsymbol{X}^T\boldsymbol{X}$ *y su inversa esta dada por:*

$$\boldsymbol{X}^T\boldsymbol{X} = \begin{pmatrix} 4 & 2 & 2 \\ 2 & 6 & -4 \\ 2 & -4 & 6 \end{pmatrix}$$

*Note que la matriz* $\boldsymbol{X}^T\boldsymbol{X}$ *no es invertible, se toma el valor de* $\lambda = 0.76$*, luego obtenemos*

$$\lambda \boldsymbol{I}_{p\times p} = \begin{pmatrix} 0.76 & 0 & 0 \\ 0 & 0.76 & 0 \\ 0 & 0 & 0.76 \end{pmatrix}$$

*De lo anterior, se tiene que*

$$(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_{p\times p}) = \begin{pmatrix} 4.76 & 2 & 2 \\ 2 & 6.76 & -4 \\ 2 & -4 & 6.76 \end{pmatrix}$$

$$(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_{p\times p})^{-1} = \begin{pmatrix} 0.53 & -0.38 & -0.38 \\ -0.38 & 0.51 & 0.41 \\ -0.38 & 0.41 & 0.51 \end{pmatrix}$$

$$\boldsymbol{X}^T\boldsymbol{Y} = \begin{pmatrix} 4.3 \\ 4.8 \\ -0.5 \end{pmatrix}$$

*Por lo tanto, la siguiente estimación:*

$$\hat{\beta} = \begin{pmatrix} 0.63 \\ 0.56 \\ 0.07 \end{pmatrix}$$

**Ejemplo 4.** *Consideremos el modelo de regresión Lasso, calculamos el valor de lambda mediante el algoritmo de validación cruzada (dejando uno a fuera), por lo tanto el* $\lambda = 0.037$ *para hallar el lambda nos apoyamos en lenguaje* `python` *y además*

$$\boldsymbol{Y} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}, \quad \boldsymbol{X} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \\ 9 & 10 \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

*Luego hallamos* $\boldsymbol{X}^T$ *que nos da lo siguiente*

$$\boldsymbol{X}^T = \begin{pmatrix} 1 & 3 & 5 & 7 & 9 \\ 2 & 4 & 6 & 8 & 10 \end{pmatrix}$$

*Así la matriz* $\boldsymbol{X}^T\boldsymbol{X}$ *y su inversa esta dada por:*

$$\boldsymbol{X}^T\boldsymbol{X} = \begin{pmatrix} 165 & 190 \\ 190 & 220 \end{pmatrix}$$

*Se toma el valor de* $\lambda = 0.037$*, luego obtenemos*

$$\lambda \boldsymbol{I}_{p\times p} = \begin{pmatrix} 0.037 & 0 \\ 0 & 0.037 \end{pmatrix}$$

*De lo anterior, tenemos que*

$$(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_{p\times p}) = \begin{pmatrix} 165.037 & 190 \\ 190 & 220.037 \end{pmatrix}$$

$$(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_{p\times p})^{-1} = \begin{pmatrix} 1.027 & -0.88 \\ -0.88 & 0.77 \end{pmatrix}$$

$$\boldsymbol{X}^T \boldsymbol{Y} = \begin{pmatrix} 95 \\ 110 \end{pmatrix}$$

*Por lo tanto, la siguiente estimación:*

$$\hat{\beta} = \begin{pmatrix} 0.01 \\ 0.4 \end{pmatrix}$$

Los ejemplos anteriores muestran que en la regresión Ridge contrae las variables en cambio en la regresión Lasso se anulan las variables que no aportan al modelo. Observemos que en el ejemplo 4 y 5 se emplean los mismos datos. No obstante, en el ejemplo 4 se implementa una regresión lineal, mientras que en el ejemplo 5 se recurre a la regresión Ridge. Mirando las estimaciones se nota que las que tienen mejor rendimiento es la de la regresión Ridge. La principal virtud del estimador de Mínimos Cuadrados Ordinarios (MCO) radica en su propiedad de ser insesgado en la mayoría de los contextos, lo que significa que su valor promedio se acerca al valor real del parámetro a estimar a medida que aumenta el tamaño de la muestra. Esto lo convierte en un método confiable para obtener una estimación precisa de la relación entre las variables.

## 5.5. Validación de supuestos

Para validar los supuestos utilizaremos varios test, para la independencia de los residuos utilizamos la prueba de Durbin-Watson la cual mide en que medida están correlacionados entre si los residuos de un modelo de regresión. Cuando los errores en un modelo de regresión están relacionados entre si, es decir, que no ocurren aleatoriamente, esto se le conoce como autocorrelación. El rango de este test es de 0 a 4, y 2 es el punto donde no muestra correlación consigo misma. Por otro lado, si el valor es menor que 2 se dice que existe una relación positiva entre los puntos de datos y si es mayor que 2, indica una relación negativa.

Durbin-Watson se usa para probar la hipótesis nula de que no hay autocorrelación en los residuos de un modelo de regresión. La hipótesis alternativa es que hay autocorrelación presente. Para saber si el valor DW (Durbin-Watson) es importante, se compara con dos números especiales, LI (límite inferior) y LS (límite superior), que se dan para diferentes niveles de importancia y diferentes formas de contar los datos. la hipótesis nula de ausencia de autocorrelación se rechaza a favor de la hipótesis alternativa de autocorrelación positiva si DW es menor que LI. La hipótesis nula de no autocorrelación se rechaza a favor de la hipótesis alternativa de autocorrelación negativa si y sólo si el valor de DW es mayor que el valor de LS.

La prueba de Durbin-Watson identifica el rango de valores de DW que muestran que los errores en un modelo de regresión están correlacionados entre sí, ya sea positiva o negativamente. Esta región está definida por dos valores esenciales: LI

(límite inferior) y LS (límite superior). La región crítica para una autocorrelación positiva se tiene el si el valor de DW cae dentro del rango $0 < \text{DW} < \text{LI}$, se rechaza la hipótesis nula de no autocorrelacion y se concluye que existe autocorrelación positiva en los residuos. Luego para la autocorrelación negativa se tiene que el rango $\text{LS} < \text{DW} < 4$ si el valor de DW cae dentro de este rango, se dice que se rechaza la hipótesis nula de no autocorrelación y se afirma que existe autocorrelación negativa en los residuos. Por último, en la región de no autocorrelación se tiene un rango de $\text{LI} < \text{DW} < \text{LS}$ si esta dentro ese intervalo no hay evidencia suficiente para rechazar la hipótesis nula de no autocorrelación en los residuos. En general, es importante tener en cuenta que la forma de la región crítica depende del nivel de significancia elegido que normalmente se toma del 5 % y el número de grados de libertar. Los valores de LI y LS se pueden encontrar en tablas precalculadas disponibles en libros de estadística [13].

La ecuación del estadístico de prueba de Durbin-Watson es el siguiente

$$d = \frac{\sum_{t=2}^{T} (e_t - e_{t-1})^2}{\sum_{t=1}^{T} e_t^2},$$

donde $e_t$ es el residual asociado a la observación en el tiempo $t$, $T$ es el número de observaciones y $e_{t-1}$ es el residuo en el tiempo $t-1$. Por ultimo se tiene la región crítica.

La prueba de White produce un estadístico de prueba que sigue una distribución

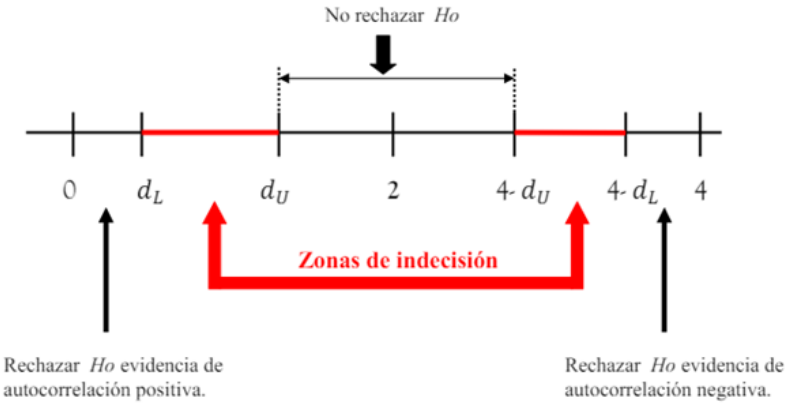


Figura 2: Región crítica Durbin-Watson.
Fuente: Elaboración propia.

de chi-cuadrado. Esta estadística nos ayuda a determinar si los residuos de nuestro modelo exhiben heterocedasticidad. La heterocedasticidad significa que los errores en las predicciones del modelo no son los mismos en todos los valores de las variables independientes. La prueba de White se usa un nuevo modelo que tiene las mismas variables independientes que el modelo original, pero con algunos términos adicionales que involucran los cuadrados y productos de las variables independientes. Luego, comparamos los errores al cuadrado del modelo original con los errores del nuevo modelo y vemos en qué medida coinciden.

Para determinar si el valor del estadístico de prueba es significativo, se compara con el valor crítico de la distribución chi-cuadrado con grados de libertad iguales al número de variables explicativas en la regresión auxiliar. Si el estadístico de prueba es mayor que el valor crítico, podemos rechazar la hipótesis nula de que los residuos tienen la misma varianza en el modelo original y podemos decir que los residuos

varían de diferentes maneras.

En la prueba de Shapiro-Wilks, el estadístico de prueba se denota por SW y se calcula como una medida de la distancia entre la distribución de la muestra y la distribución normal. Para determinar si el valor de SW es significativo, se compara con el valor crítico de la distribución de Shapiro-Wilks para un nivel de significancia predefinido y la longitud de la muestra. Si el valor de SW es menor que el valor crítico, se rechaza la hipótesis nula de normalidad y se concluye que la distribución de la muestra no es normal [14].

El estadístico de prueba de Shapiro-Wilks es el siguiente:

$$SW = \frac{\left(\sum_{i=1}^{n} a_i x_{(i)}\right)^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2},$$

donde $x_{(i)}$ es el número que ocupa la $i$-ésima posicion en la muestra, $\bar{x} = \frac{x_1 + \cdots + x_n}{n}$ es la media muestral.

Las $a_i$ se calculan

$$(a_1, \ldots, a_n) = \frac{m^\top V^{-1}}{(m^\top V^{-1} V^{-1} m)^{1/2}}$$

donde,

$$m = (m_1, \ldots, m_n)^\top,$$

donde, $m_1, \ldots, m_n$ los valores medios del estadístico ordenado y $V$ denota la matriz de covarianzas de ese estadístico de orden.

# 6. Configuración experimental

En esta sección se tratará la validación cruzada, la cual es una técnica fundamental en el aprendizaje automático que ayuda a evaluar el rendimiento de un modelo de manera precisa y confiable. El principal beneficio es evitar el sobreajuste, esto sucede cuando el modelo aprende demasiado de los datos de entrenamiento y no es capaz de generalizar bien los nuevos datos, para esto, la validación cruzada ayuda a identificar el sobreajuste al evaluar el rendimiento del modelo en diferentes subconjuntos de datos. Sumandole a lo anterior, permite ajustar los hiperparámetros del modelo para optimizar su rendimiento, esto ayuda a mejorar la precisión del modelo y su capacidad de generalización. Por último, la validación cruzada proporciona una evaluación más robusta del rendimiento del modelo que la simple división del conjunto de datos en entrenamiento y prueba.

## 6.1. Validación cruzada

La validación cruzada, es una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y prueba. En general, el procedimiento de este criterio consiste en dividir las observaciones de la muestra disponible en dos conjuntos: uno para entrenar el modelo y otro para probarlo. Luego, se ajusta el modelo con la muestra de entrenamiento y finalmente se evalúa utilizando la muestra de prueba.

### Tipos de validación

1. **Validación simple**

Este tipo de validación consiste en dividir aleatoriamente las observaciones disponibles en dos grupos iguales, la muestra de entrenamiento se emplea para ajustar el modelo y la muestra de prueba se usa para evaluar el modelo.

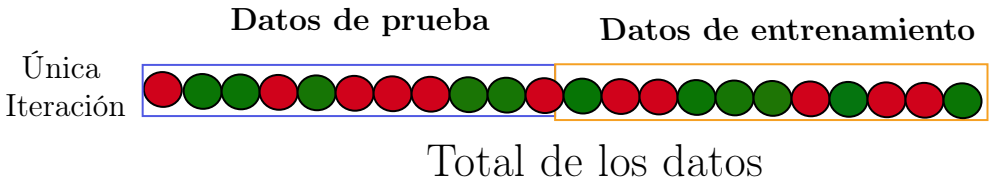


Figura 3: Interpretación de una validación simple.
Fuente: Elaboración propia

Según James, Witten, Hastie et al. en [15], este tipo de validación es la mas simple, pero presenta dos problemas importantes:

- **Problemas de varianza:** La estimación del error de prueba, es altamente variable dependiendo de las observaciones que se incluyan en la muestra de entrenamiento y en la muestra de prueba.
- **Problema de sesgo:** Al dividir las observaciones disponibles como muestra de entrenamiento, se dispone de menos información para el ajuste del modelo y por lo tanto se reduce su capacidad de precisión. En consecuencia, puede presentar un sobreajuste del error de prueba comparado con el

que se obtendría si se emplearan todas las observaciones del conjunto de entrenamiento.

2. **Validación cruzada dejado uno fuera**

La validación cruzada dejando uno fuera, se realiza de forma iterativa, se inicia empleando como muestra de entrenamiento, todas las observaciones disponibles excepto una, que se excluye para emplearla como muestra de prueba. Si se emplea una única observación para calcular el error de prueba, este varia dependiendo de que observación seleccionada. Para evitar este inconveniente, el proceso se repite tantas veces como observaciones disponibles, excluyendo en cada iteración una observación distinta, ajustando el modelo con el resto y calculando el error con dicha observación. El error de prueba es el promedio de todos los $i$ errores calculados.

$$(\text{Validación cruzada - error}) = \frac{1}{R}\sum_{i=1}^{R} \text{ECM}_i$$

donde el $\text{ECM}_i$ corresponde al ECM de la muestra i-ésima, según James, Witten, Hastie et al. en [15], la validación cruzada dejando uno fuera reduce la variabilidad que se presenta en la validación simple. Al final del proceso, todas las observaciones disponibles son utilizadas tanto las de entrenamiento como las de prueba. Sin embargo, este método tiene un alto costo computacional, debido a que, el modelo debe ser reajustado y validado tantas veces como observaciones disponibles. Esto puede representar una dificultad en algunos casos.

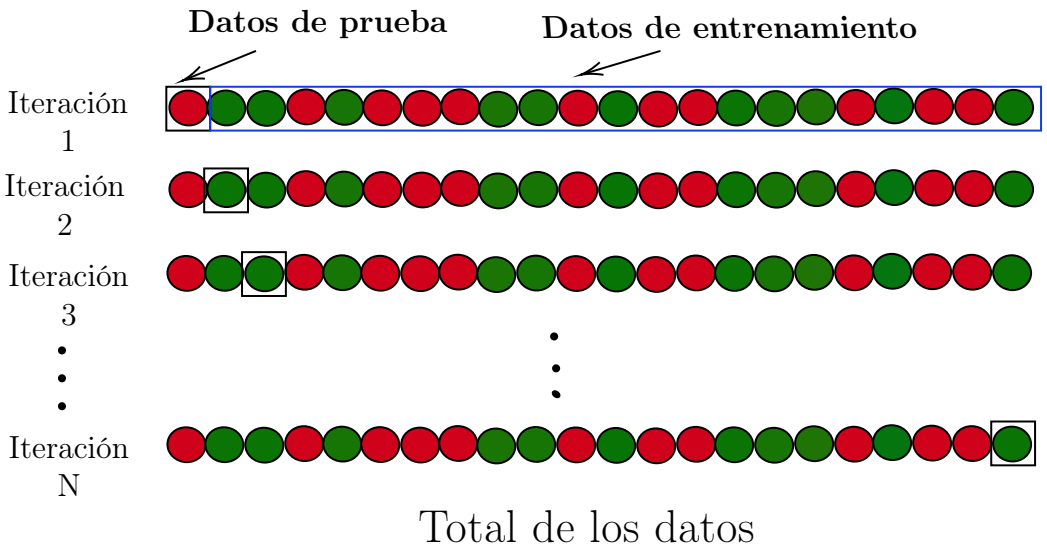


Figura 4: Interpretación de una validación dejando uno fuera.
Fuente: Elaboración propia

3. **Validación cruzada de K-iteraciones**

La validación cruzada de K-iteraciones, es un proceso iterativo, que consiste en dividir los datos de la muestra de forma aleatoria en K grupos de aproximadamente del mismo tamaño, los K-1 grupos se emplean para ajustar el modelo (muestra de entrenamiento) y uno de los grupos se emplean como muestra de prueba, este proceso se repite K-veces utilizando un grupo distinto como muestra de prueba en cada iteración.

El proceso genera K estimaciones del error de prueba cuyo promedio se emplea como estimación final, es decir

$$(\text{Validación cruzada} - Error_{(K)}) = \frac{1}{K}\sum_{i=1}^{K} \text{ECM}_i,$$

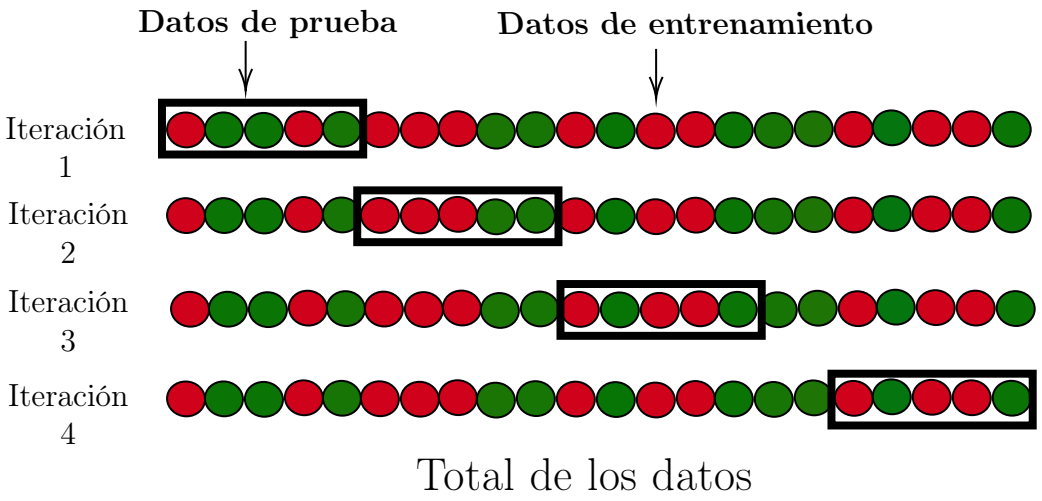


Figura 5: Interpretación de una validación de 4-iteraciones.
Fuente: Elaboración propia

donde el $\text{ECM}_i$ corresponde al ECM de la muestra i-ésima y K la muestra que se toma. Este método posee dos ventajas muy importantes las cuales son:

*a*) **Balance entre sesgo y varianza:** La validación cruzada, *dejando uno fuera*, está en un extremo de sesgo bajo, con alta varianza y al dividir el conjunto de observaciones en menos bloques implicaría un mayor sesgo y menor varianza. El equilibrio no esta en ninguno de los extremos, por lo cual se recomienda dividir el conjunto de observaciones en $K = 5$ o $K = 10$, debido a que se consigue un mejor balance entre sesgo y varianza [16].

*b*) **Computacional:** El numero de iteraciones necesarias vienen determinado por el valor $K$ escogido, en consecuencia la validación cruzada *dejando uno fuera*, tiene un coste computacional muy alto. Por lo general se recomienda que $K \in [5, 10]$ donde $K \in \mathbb{Z}^+$ [16].

## 6.2. Método de validación cruzada para escoger datos de entrenamiento y de predicción

El paquete que se usara en el estudio es sklearn.model con la función `train_test_split`, donde este permite dividir un dataset en dos bloques, generalmente bloques destinados al entrenamiento y validación del modelo. La función añade al bloque de entrenamiento el 75 % de los registros, y al bloque de pruebas el 25 % restante. Se logra controlar estas cifras con los parámetros `train_size` y `test_size`. Estos parámetros pueden ser números enteros, indicando exactamente el número de registros a incluir, o un número real, en cuyo caso indican el porcentaje del total de registros a incluir. Una vez generados los bloques de entrenamiento y pruebas, se consigue extraer la variable objetivo para crear, de esta forma, las estructuras que típicamente exigen los algoritmos para su entrenamiento [17].

- El parámetro `random_state` simplemente fija una semilla para el generador de números aleatorios, lo que permite reproducir la función.
- El parámetro `shuffle` que toma el valor verdadero por defecto especifica si los registros deberán ser desordenados previamente o no.

Si se pasa a la función, no un dataset, sino dos estructuras de la misma longitud (típicamente el bloque de características y la variable objetivo), `train_test_split` va a generar bloques de entrenamiento y pruebas para ambas estructuras, haciendo innecesaria la división posterior entre bloque de características y variable objetivo.

**Validación cruzada: Evaluación del rendimiento**

Cometer el error metodológico de aprender los parámetros de una función de predicción y probarla sobre los mismos datos, resultaría en un modelo que simplemente repetiría las etiquetas de las muestras ya vistas. Aunque esto le daría al modelo una puntuación perfecta, no sería capaz de hacer predicciones útiles sobre datos nuevos. Este fenómeno se conoce como sobreajuste. Para evitarlo, es común reservar parte de los datos disponibles como conjunto prueba `X_test`, `y_test` al realizar experimentos supervisados en aprendizaje automático.

En `scikit-learn` se puede calcular rápidamente una división aleatoria en conjuntos de entrenamiento y prueba con la función de ayuda `train_test_split`. Existe un riesgo de sobreajuste en el conjunto de prueba cuando se evalúan diferentes ajustes llamados ("hiperparámetros") para los estimadores, como por ejemplo, al establecer manualmente el ajuste $C$ en una SVM. Esto ocurre porque los parámetros pueden adaptarse libremente hasta que el estimador alcance su rendimiento óptimo.

De esta manera, el conocimiento del conjunto de pruebas puede filtrarse en el modelo y las métricas de evaluación ya no reflejan el rendimiento general. Para solucionar este problema, una opción es reservar otra parte del conjunto de datos como çonjunto de validación": se entrena con el conjunto de entrenamiento, luego se evalúa con un conjunto aparte llamado validación y cuando parece que los resultados son exitosos, finalmente se realiza la evaluación definitiva en el conjunto

de prueba. Sin embargo, al dividir los datos disponibles en tres conjuntos, se reduce drásticamente el número de muestras que pueden utilizarse para el aprendizaje del modelo, y los resultados pueden depender de una elección aleatoria concreta para el par de conjuntos (entrenamiento, validación).

La solución a este problema es un procedimiento denominado cross-validation (CV). El conjunto de prueba debe seguir utilizándose para la evaluación final, pero el conjunto de validación ya no es necesario cuando se realiza la CV. En el enfoque básico, denominado CV $K$-fold, el conjunto de entrenamiento se divide en k conjuntos más pequeños (más adelante se describen otros enfoques, pero en general siguen los mismos principios). Se sigue el siguiente procedimiento para cada uno de los $K$ "folds":

- Se entrena un modelo utilizando $K-1$ de los folds como datos de entrenamiento;
- El modelo resultante se valida en la parte restante de los datos, es decir, se utiliza como conjunto de prueba para calcular una medida de rendimiento como la precisión.

La validación cruzada $K$-fold es un método de evaluación de modelos estadísticos que consiste en dividir el conjunto de datos en $K$ subconjuntos, también conocidos como pliegues. A continuación, se entrena el modelo en $K-1$ pliegues y se evalúa en el pliegue restante. Este proceso se repite $K$ veces, de modo que cada pliegue se utiliza una vez como conjunto de validación.

Se usa la técnica de validación cruzada $K$-fold para evaluar el desempeño de un modelo en datos que no se han utilizado durante su entrenamiento. El método resulta más preciso, al emplear cada pliegue como conjunto de validación una vez y así obtener estimaciones más fiables del comportamiento del modelo frente a distintos escenarios, lo cual difiere significativamente respecto al uso único e insuficiente dado por un solo grupo probatorio.

El funcionamiento de la validación cruzada $K$-fold se puede analizar de la siguiente manera:

- **División del conjunto de datos :** El primer paso es dividir el conjunto de datos en $K$ subconjuntos de tamaño aproximadamente igual. Esto se puede hacer de forma aleatoria o sistemática.
- **Entrenamiento del modelo :** Para cada iteración, se entrena el modelo en $K - 1$ pliegues. Los pliegues se seleccionan de forma aleatoria o sistemática, pero no se pueden seleccionar los mismos pliegues para todas las iteraciones.
- **Evaluación del modelo :** Para cada iteración, el modelo se evalúa en el pliegue restante. Este pliegue se utiliza como conjunto de validación para estimar el rendimiento del modelo en datos nuevos.
- **Repetición :** El proceso se repite k veces, de modo que cada pliegue se utiliza una vez como conjunto de validación.

Por ejemplo, supongamos que tenemos que tenemos un conjunto de datos con 100 muestras. Si queremos utilizar la validación cruzada $K$-fold con $K = 5$,

dividiremos el conjunto de datos en 5 subconjuntos de 20 muestras cada uno. Para la primera iteración, utilizaremos los pliegues 1, 2, 3 y 4 para entrenar el modelo. El pliegue 5 se utilizará como conjunto de validación. Para la segunda iteración, utilizaremos los pliegues 2, 3, 4 y 5 para entrenar el modelo. El pliegue 1 se utilizará como conjunto de validación. El proceso se repetirá hasta que todos los pliegues hayan sido utilizados como conjunto de validación una vez [16].

La validación cruzada $K$-fold posee varias ventajas una de ellas es proporciona una estimación más precisa del rendimiento del modelo en datos nuevos que utilizar un solo conjunto de prueba, además es menos propensa a sesgos que utilizar un solo conjunto de prueba. Una desventaja que se puede presenta es si se utiliza un valor alto de $K$ es difícil de interpretar, también es mas costosa computacionalmente que utilizar un solo conjunto de prueba.

Como señala Pedregosa, Varoquaux, Gramfort et al. en [17], se tiene que el paquete `sklearn.model_selection` extraido de la biblioteca `scikit-learn` es importante en el desarrollo de modelos de aprendizaje automático. Una función útil dentro del paquete es `train_test_split`, utilizada para dividir un conjunto de datos en dos subconjuntos: uno destinado al entrenamiento del modelo y otro utilizado posteriormente para evaluar su rendimiento. La función `train_test_split` se utiliza para resolver el problema de evaluar modelos de aprendizaje automático. Al separar el conjunto de datos en dos partes, una destinada al entrenamiento y otra a la prueba, se consiguen los siguientes propósitos:

- **Entrenamiento del modelo:** Se ofrece un conjunto de datos para entrenar un modelo de aprendizaje automático. Durante esta etapa, el modelo adquirirá conocimiento sobre patrones y relaciones presentes en los datos.
- **Evaluación del modelo:** La evaluación del rendimiento del modelo en datos no vistos durante el entrenamiento es crucial para determinar su capacidad de generalización y precisa predicción en nuevos datos.
- **Evita el sobreajuste:** La división en conjuntos de entrenamiento y prueba es útil para determinar si el modelo está sobreajustando los datos durante el entrenamiento. Si el rendimiento del modelo no es bueno en el conjunto de prueba, esto puede ser una señal de que se ha producido un sobreajuste.

Ahora observemos un ejemplo con datos ficticios para evidenciar el proceso que hace en lenguaje `Python`:

```
import numpy as np
X = np.random.rand(100, 3)
y = np.random.randint(2, size = 100)
from sklearn.model_selection
import train_test_split
X_train,
X_test,
y_train,
y_test =
train_test_split(X, y, test_size = 0.2, random_state = 42)
print("Conjunto de entrenamiento (X_train):")
print(X_train[:5])
print("\nConjunto de prueba (X_test):")
print(X_test[:5])
print("\nEtiquetas de entrenamiento (y_train):")
```

```
print(y_train[:5])
print("\nEtiquetas_de_prueba_(y_test):")
print(y_test[:5])
```

## 6.3. Selección de variables explicativas

La selección de variables se enfoca en solucionar el problema de construir o elegir un modelo. En general, al incluir más variables en un modelo de regresión se mejora la adaptación a los datos pero aumenta la cantidad de parámetros estimados y disminuye su precisión individual, lo que genera mayor varianza. Por ende, si no existe una selección adecuada puede haber "sobreajuste.[en] la función estimada generando problemas significativos.

Si se utilizan menos variables de lo necesario para el modelo, la variabilidad disminuirá pero habrá un aumento en los sesgos del análisis, dando lugar a una descripción poco precisa. Sin embargo, algunas variables predictoras pueden afectar negativamente a la fiabilidad del modelo si están correlacionadas con otras. El objetivo principal al seleccionar las variables es lograr un ajuste adecuado sin ignorar la simplicidad y encontrar equilibrio entre bondad de ajuste y sencillez.

Para esa selección de variables tenemos algoritmos los cuales nos puede ayudar a elegir las cuales son las siguientes:

- **Método Forward:** Este método inicia con un modelo sin variables explicativas y en cada paso agrega una variable explicativa significativa. Se realiza tantos

pasos como sea posible hasta que ya no se pueden agregar variables significativas.

- **Método Backward:** En este método se inicia con un modelo con todas las variables explicativas y en cada paso se va quitando la variable explicativa menos significativa.

- **Método Stepwise:** Es una serie de procedimientos de selección automática de variables significativas, basados en la inclusión ó exclusión de las mismas en el modelo de una manera secuencial.

## 6.4. Gradiente descendente

El método del gradiente descendente (GD) es un algoritmo de optimización iterativo que se utiliza para encontrar el mínimo de una función diferenciable. La idea es tomar pasos de manera repetida en dirección contraria al gradiente de la función. El gradiente de una función es una medida de la dirección en la que la función crece más rápidamente. Se puede calcular utilizando la regla de la cadena.

La idea del gradiente descendente es ajustar los parámetros de forma iterativa para minimizar una función. Concretamente, se tiene de una función diferenciable convexa $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$, el algoritmo (GD) permite encontrar un $w$ en $\Omega$ tal que $f(w)$ es un mínimo, es decir, GD se utiliza para determinar los elementos del siguiente conjunto:

$$w \in \underset{w \in \Omega}{\operatorname{argmin}} f(w). \tag{22}$$

El método utiliza un conjunto de iteraciones para determinar los valores óptimos de $w$ en la función $f(w)$. Estas iteraciones siguen una regla de actualización definida de la siguiente manera:

$$w_{t+1} = w_t - \eta_t \nabla f\left(w_t\right), \tag{23}$$

que usualmente se inicializa en cero y cada iteración, como se puede observar, se hace en la dirección negativa del gradiente. Recordemos que el gradiente se define como el siguiente vector:

$$\nabla f(w) = \left(\frac{\partial f}{\partial x_1}(w), \ldots, \frac{\partial f}{\partial x_n}(w)\right). \tag{24}$$

En el algoritmo de GD, un hiperparámetro crucial es la tasa de aprendizaje $\eta_t > 0$. Si esta tasa es muy pequeña, se requerirán múltiples iteraciones para alcanzar la convergencia y esto llevará mucho tiempo. Por otro lado, si la tasa resulta ser demasiado alta, existe el riesgo de saltar por encima del mínimo global deseado y terminar en una posición peor que antes. Este comportamiento podría provocar divergencias en los valores obtenidos sin encontrar soluciones satisfactorias debido a constantes incrementos incorrectos.

Para el funcionamiento de este algoritmo se define primero las características esenciales que tiene el gradiente:

- El gradiente es perpendicular a las curvas de nivel de $f$, de manera que para cualquier dirección $v \in \mathbb{R}^n$ ortogonal a $\nabla f(p)$, es una dirección de cambio nulo.

Esto se observa fácilmente al parametrizar la curva $S_k = \{p \in \Omega : f(p) = k\}$ mediante una función $\alpha : I \subset \mathbb{R} \to S_k$ tal que $\alpha(0) = p$, pues al calcular el producto punto de $\nabla f(p)$ con la velocidad de $\alpha$ en $p$ se obtiene que la tasa de cambio es:

$$|df_p(\alpha'(0))| = |\nabla f(p) \cdot \alpha'(0)| = 0,$$

es decir, la tasa de cambio en la dirección de $\alpha'(0)$ es cero.

- El gradiente indica la dirección ascendente de la tasa de máximo cambio de $f$ en el punto $p$. La tasa máxima se calcula como $\|\nabla f(p)\|$. La razón de esto, se aprecia cuando se considera un vector $v \in \mathbb{R}$ tal que $\|v\| = 1$, de manera que para este vector la tasa de cambio es:

$$|df_p(v)| = ||\nabla f(p)|||||v||| \cos\theta| \leq \|\nabla f(p)\|$$

Dicha magnitud es máxima cuando $\theta = 2n\pi$ con $n \in \mathbb{Z}$, es decir, para que $|df_p(v)|$ sea máxima, los vectores $\nabla f(p)$ y $v$ deben ser paralelos, de esta manera, la función $f$ crece más rápidamente en la dirección del vector $\nabla f(p)$ y decrece más rápidamente en la dirección de $-\nabla f(p)$, en efecto, si $v = \frac{\nabla f(p)}{\|\nabla f(p)\|}$, entonces $df_p(v) = \|\nabla f(p)\|$.

Utilizando la iteración definida en (23), podemos generar una sucesión de puntos $\{w_t\}_{t\in[m]_{\mathbb{N}_0}}$, donde cada $w_{t+1}$ es creado a partir del punto anterior, y cumple con la propiedad de que $f(w_{t+1}) < f(w_t)$. Esta sucesión puede ser analizada utilizando el polinomio de Taylor. Si tomamos un punto inicial $w_0$ y expandimos alrededor de

él usando el primer término del polinomio, obtenemos:

$$f(w_1) - f(w_0) \approx \langle w_1 - w_0, \nabla f(w_0) \rangle = -\eta \left\| \nabla f(w_o) \right\|^2$$

Por lo tanto:

$$f(w_1) - f(w_0)) = -\eta \left\| \nabla f(w_o) \right\|^2 + o(\eta)$$

De este modo, al elegir un valor adecuado de $\eta$, se puede asegurar que $f(w_1)$ es menor que $f(w_0)$. Este proceso se puede repetir para las variables $w_{t+1}$ y $w_t$ con el resultado de una mejora en la solución.

El algoritmo del gradiente descendente se define de la siguiente forma:

1. Input: $w_0$, $m$, $\eta$, $\nabla f(w)$
2. for $k = 0$ to $m$ do:
3. $w \leftarrow w - \eta \nabla f(w)$
4. end
5. return: $w$

Note que $w_0$ es la condición inicial del algoritmo, normalmente se elige aleatoriamente o a partir de conocimientos previos sobre el problema. Si no se tiene información previa del problema, una buena practica es iniciar con valores cercanos a cero por lo general se toma un valor de 0.5. Por otro lado $m$ es el número máximo de iteraciones,

es decir, la cantidad de veces que se actualizara el parámetro $w$. Un alto valor de $m$ puede llevar a una mejor convergencia, pero también se requiere más tiempo de cómputo. Su elección depende de la complejidad del problema y de la precisión que se desea, normalmente se suele fijar un número máximo de iteraciones y detener el algoritmo si la mejora en la función objetivo es muy pequeña, valor que se deja predeterminado es de 100 iteraciones. Por otra parte, $\eta$ es la tasa de aprendizaje controla el tamaño de los pasos que se dan en cada actualización de $w$, el valor típico que se toma es 0.001, pero depende del problema que se tenga específicamente. Por ultimo, $\nabla f(w)$ es la función gradiente de $f$. Observe que la regla de actualización se definió por 23. Es importante notar que finalmente $w_{t+1}$ es tal que:

$$w_{t+1} \in \underset{w\in\Omega}{\operatorname{argmin}} \frac{1}{2\eta_t} \left\| w - w_t + \eta_t \nabla f\left(w_t\right) \right\|.$$

Es fácil comprobar que el problema de optimización anterior se puede reescribir de la siguiente manera:

$$\underset{w\in\Omega}{\operatorname{argmin}} \frac{1}{2\eta_t} \left\| w - w_t + \eta_t \nabla f\left(w_t\right) \right\| = \underset{w\in\Omega}{\operatorname{argmin}} \left( f\left(w_t\right) + \left\langle \nabla f\left(w_t\right), w - w_t \right\rangle + \frac{1}{2\eta_t} \left\| w - w_t \right\| \right).$$

Por lo tanto, el $w_{t+1}$ es obtenido para minimizar la linealización de la función $f$ alrededor del punto $w_t$, manteniéndolo lo suficientemente aproximado a este el punto $w_t$.

Es necesario destacar que aunque el algoritmo complete todas las iteraciones, su resultado no siempre será una aproximación precisa del mínimo de la función $f$.

Es por ello que es fundamental establecer un criterio para determinar si el resultado obtenido es apropiado o insuficiente.

```
def minimize(f, grad_f, w0, eta, m):
w = w0
  for _ in range(m):
    w = eta * grad_f(w)

  return w
def f(w):
  return w**2

def grad_f(w): \% Para cualquier funcion
  return 2*w

w0 = 1.0
eta = 0.1
m = 100

minimum = minimize(f, grad_f, w0, eta, m)

print(minimum)
```

Del código anterior se tiene que la función `minimize` toma como entrada la función objetivo $f$, la función del gradiente `grad_f`, el punto inicial `w_0`, el tamaño de la tasa de aprendizaje `eta` y el número de iteraciones `m`. Luego la función inicializa el punto actual `w` al valor de `w_0`. Por otro lado, se realiza un bucle `for` para iterar `m` veces, dentro del bucle se calcula el gradiente de la función en el punto actual `w` usando la función `grad_f`, seguidamente se actualiza el punto actual `w` restando el gradiente multiplicado por el tamaño de la tasa de aprendizaje. Finalmente, la

función devuelve el punto mínimo estimado `w`.

Una posible condición de detención del algoritmo que puede ser considerada por el lector es la interrupción de las iteraciones cuando $||\nabla f(x_t)|| = 0$, aunque esta opción no resulta práctica debido a diferentes factores influyentes, como el comportamiento decimal y una elección adecuada en cuanto a tasa de aprendizaje. En términos comunes, se suele establecer un parámetro tolerable $\epsilon > 0$ junto con alguno de los siguientes criterios se usan para detener las iteraciones:

- **Condición sobre el gradiente:**

$$||\nabla f(x_t)|| < \epsilon$$

  El algoritmo se detiene cuando la norma del gradiente es menor que $\epsilon$, lo que indica que se ha encontrado un mínimo local con una precisión tolerable. Por lo general no existe un valor único para $\epsilon$ que funcione para todos los problemas, su elección depende de la complejidad del problema o los recursos computacionales disponibles. Normalmente, se recomienda elegir un $\epsilon$ lo suficientemente pequeño para garantizar así una buena precisión, pero no tan pequeño como para que el algoritmo sea demasiado lento. Los valores más usuales son 0.001, 0.000001 y 0.000000001 lo cual se puede consultar en [17].

- **Condición sobre las diferencias sucesivas relativas de la función objetivo:**

$$\frac{|f(x_{t+1}) - f(x_t)|}{|f(x_t)|} < \epsilon$$

si el denominador es muy pequeño, es conveniente remplazarlo por $max\{1, |f(x)|\}$.

- **Condición sobre las diferencias sucesivas relativas de la variable independiente:**

$$\frac{||x_{t+1} - x_t||}{||x_t||} < \epsilon$$

si el denominador es muy pequeño, es conveniente remplazarlo por $max\{1, ||x_t||\}$.

La tasa de aprendizaje $\eta$ es determinada por la minimización exacta de:

$$\eta_t \in \underset{\eta>0}{\operatorname{argmin}} f(x_t - \eta \nabla f(x_t)).$$

Principalmente se utiliza para solucionar problemas con carácter cuadrático en los que el cálculo de $\eta$ es económico pero la evaluación del gradiente resulta costosa. Si no ocurre esto, entonces no vale la pena dedicar tiempo y recursos a resolver este subproblema de manera exacta. Este algoritmo es aplicable a funciones de pérdida no lineales y además es eficiente computacionalmente, sin embargo una desventaja de este algoritmo es que puede ser lento para converger en funciones de pérdida complejas.

### Método del gradiente estocástico

El Descenso Gradiente Estocástico (SGD) es un algoritmo de optimización iterativo que se utiliza para encontrar el mínimo de una función diferenciable. La idea es tomar pasos de manera repetida en dirección contraria al gradiente de la función, pero utilizando un solo ejemplo de los datos en cada actualización.

Sea la función objetivo $f : \mathbb{R}^n \to \mathbb{R}$ diferenciable y una función $g(w;\eta)$ que representa un estimador para el gradiente $\nabla f : \mathbb{R}^n \to \mathbb{R}$, la idea de este método es hacer una actualización de $w \in \mathbb{R}^n$ para solucionar el problema de minimización $\min_{w\in\mathbb{R}^n} f(w)$.

Luego lo que propone el método es que la aproximación del argumento que minimiza la función es:

$$w_{k+1} = w_k - \alpha_k g(w_k;\eta_k).$$

Donde $\alpha_k \in [0,1]$ y $g(w_k;\eta_k)$ es el termino que aproxima $\nabla f(w_k)$ con un cierto ruido $\eta_k$ en la $k$-ésima actualización. En el método del gradiente estocástico, la función de pérdida se reduce a medida que el algoritmo converge. Esto se debe a que el algoritmo actualiza los parámetros en la dirección del gradiente negativo de la función de pérdida. Como el gradiente negativo de una función decreciente siempre apunta hacia abajo, el valor de la función también debe disminuir.

Sin embargo, hay algunos casos en los que el método del gradiente estocástico puede no converger. Por ejemplo, si la función de pérdida tiene un mínimo local, el algoritmo puede quedarse atascado en ese mínimo, incluso si el mínimo no es el mínimo global. Para garantizar que el método del gradiente estocástico se reduce, se pueden utilizar algunas técnicas, como:

- **Utilizar una tasa de aprendizaje decreciente :** Esto ayuda a evitar que el algoritmo se quede atascado en un mínimo local.

- **Utilizar un método de regularización :** Esto puede ayudar a evitar que el algoritmo se sobreajuste a los datos.
- **Utilizar un método de inicialización adecuado :** Esto puede ayudar a que el algoritmo comience en una buena región de la función de pérdida.

En el caso de la regresión ridge y lasso, la tasa de aprendizaje decreciente es una técnica eficaz para garantizar que el algoritmo se reduce. También se puede utilizar un método de regularización, como la regularización ridge o lasso. La implementación de SGD en `Python` se puede realizar utilizando la biblioteca `scikit-learn`. La clase `SGDClassifier` implementa un algoritmo de SGD para problemas de clasificación, mientras que la clase `SGDRegressor` implementa un algoritmo de SGD para problemas de regresión. Los parámetros que son más importantes en los algoritmos de SGD son los siguientes:

- `learning_rate` : La tasa de aprendizaje controla el tamaño del paso que se da en cada actualización. Una tasa de aprendizaje demasiado alta puede hacer que el algoritmo no converja, mientras que una tasa de aprendizaje demasiado baja puede hacer que el algoritmo tarde mucho en converger.
- `n_iter` : El número de iteraciones es el número de veces que se recorre el conjunto de datos de entrenamiento. Un mayor número de iteraciones generalmente conduce a una mejor precisión, pero también requiere más tiempo de entrenamiento.
- `loss` : La función de pérdida define la medida de la inadecuación del modelo a los datos. Los valores que se utilizan más son `log_loss` y para la clasificación tenemos `squared_error` para la regresión.

- `penalty` : La regularización se utiliza para evitar que el modelo se ajuste demasiado a los datos de entrenamiento.

Observemos un ejemplo de este algoritmo utilizando `Python`:

```
from sklearn.linear_model import SGDClassifier, SGDRegressor
clf = SGDClassifier(loss="log\_loss", n\_iter=100)
clf.fit(X_train, y_train)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

Los algoritmos de SGD son una herramienta poderosa que se puede utilizar para resolver una variedad de problemas de aprendizaje automático. Son eficientes computacionalmente, robustos a la presencia de ruido y fáciles de implementar. Sin embargo, pueden ser menos precisos que otros algoritmos de optimización y pueden ser más propensos a quedar atrapados en mínimos locales.

La clase `SGDClassifier` implementa una rutina de aprendizaje SGD de primer orden. El algoritmo itera sobre los ejemplos de entrenamiento y para cada ejemplo actualiza los parámetros del modelo de acuerdo con la regla de actualización dada por

$$w \leftarrow w - \eta \left[ \alpha \frac{\partial R(w)}{\partial w} + \frac{\partial L(w^T x_i + b, y_i)}{\partial w} \right] \tag{25}$$

dónde $\eta$ es la tasa de aprendizaje que controla el tamaño del paso en el espacio de parámetros, esta tasa de aprendizaje se toma de manera de cual converger para Ridge se toma 0.01, 0.001, 0.0001 y para Lasso se tiene 0.1, 0.01, 0.05 el cual se toma a conveniencia para que tenga una excelente convergencia. También se puede utilizar una tasa de aprendizaje adaptativamente decreciente. En este caso, la tasa

de aprendizaje se reduce automáticamente a medida que el algoritmo converge. Esto puede ayudar a evitar que el algoritmo se quede atascado en un mínimo local. La interceptación $b$ se actualiza de manera similar pero sin regularización. La tasa de aprendizaje $\eta$ puede ser constante ó decrecer gradualmente. Para la clasificación, el programa de tasa de aprendizaje predeterminado ( `learning_rate='optimal'`) viene dado por:

$$\eta^{(t)} = \frac{1}{\alpha(t_0 + t)},$$

dónde $t$ es el paso de tiempo, $\eta^{(0)} = 0.0001$, $t_0$ se determina en base a una heurística propuesta por Léon Bottou de modo que las actualizaciones iniciales esperadas sean comparables con el tamaño esperado de los pesos esto suponiendo que la norma de las muestras de entrenamiento es aproximadamente uno [18].

# 7. Datasets

Para la realización de esta implementación se usarán dos bases de datos 'Recién nacidos Guadalajara de Buga 2016' y 'Saber 11 del año 2020-2' . Las bases de datos se adquiere mediante (https://www.datos.gov.co) donde estos datos son de libre acceso para el público en general. La primera base de datos esta conformada por las variables: 'Genero', 'Peso', 'Talla', 'Tiempo de gestación', 'consultas prenatales', 'APGAR 1', 'Edad de la madre', 'Hijos nacidos', 'Numero de embarazos' y 'Edad del padre'. Para la segunda base de datos de 'Saber 11 del año 2020-2' se tienen las siguientes variable: 'Genero', 'Puntaje lectura critica', 'Percentil lectura critica', 'Desempeño lectura critica', 'Puntaje matemáticas', 'Percentil matemáticas', 'Desempeño matemáticas', 'Puntaje ciencias naturales', 'Percentil ciencias naturales', 'Desempeño ciencias naturales', 'Puntaje sociales ciudadanas', 'Percentil sociales ciudadanas', 'Desempeño sociales ciudadanas', 'Puntaje ingles', 'Percentil ingles', 'Desempeño ingles','Puntaje global', 'Percentil global' y 'Estudiante generación E'.

En las dos bases de datos se encuentran variables de todas las categorías, sin embargo solo tomamos los datos de categoría numérica, lo que implica que estas variables son de naturaleza cuantitativa discreta. En cuanto a las demás que no son numéricas se consideran variable cualitativa. Ahora, para realizar esta implementación primero se limpia las bases de datos para así detectar datos faltantes o datos inconsistentes. La técnica que se emplea para los datos faltantes es la imputación, esta consiste en observar el comportamiento de los datos vecinos para hacer una estimación del valor del dato faltante. El método que se utiliza es la imputación por la media aritmética, la

| Variables de la base de datos Recién Nacidos Guadalajara de Buga 2016 | | |
|---|---|---|
| Variable | Tipo | Significado |
| Género | Cualitativa | Identificación del recién nacido. |
| Peso | Cuantitativa discreta | El peso del recién nacido dada en centímetros. |
| Talla | Cuantitativa discreta | La altura del recién nacido dada en centímetros. |
| Tiempo de gestación | Cuantitativa discreta | El tiempo que el recién nacido duro en el vientre de la madre dado en semanas. |
| Consultas prenatales | Cuantitativa discreta | La cantidad que la madre fue a las consultas médicas. |

cual consiste en tomar todos los valores conocidos en la variable donde están los datos faltantes, se calculan la media y se remplazan en los datos faltantes. La desventaja de esta imputación es que al remplazar muchos datos faltantes con el mismo valor se cambia la distribución de los datos. Después de la limpieza de datos se procede a realiza el estudio con las tres regresiones en estudio la lineal, Ridge y Lasso.

## 7.1. Dataset Nacimientos Buga 2016

Ahora se presenta la tabla de las 10 variables de la base de datos Recién Nacidos Guadalajara de Buga 2016, donde se explica cada una de ellas:

| Variables de la base de datos Recién Nacidos Guadalajara de Buga 2016 | | |
|---|---|---|
| Variable | Tipo | Significado |
| APGAR 1 | Cuantitativa discreta | Examen rápido que se realiza al primer y quinto minuto después del nacimiento del bebé. |
| Edad de la Madre | Cuantitativa discreta | La edad de la madre medida en años. |
| Hijos nacidos | Cuantitativa discreta | Cantidad de hijos de la madre. |
| Número de embarazos | Cuantitativa discreta | La cantidad de embarazos que la madre ha tenido. |
| Edad del padre | Cuantitativa discreta | La edad del padre medida en años. |

En la primera base de datos se predice la variable 'Peso' con respecto la 'Edad de la madre' y 'Tiempo de gestación'. Es importante analizar el peso del recién nacido, dado que es un factor fundamental para su salud y desarrollo. Los bebés con un peso bajo al nacer tienen un mayor riesgo de complicaciones, como la muerte súbita del lactante, las infecciones y los problemas respiratorios. El tiempo de gestación y la edad de la madre son dos factores que influyen en el peso del recién nacido. El tiempo de gestación normal es de 38 a 42 semanas. Los bebés que nacen antes de las 37 semanas se consideran prematuros, mientras que los que nacen después de las 42 semanas se consideran postmaduros.

Figueiredo, Gomes-Filho, Silva et al. [2], menciona que los bebés prematuros suelen tener un peso menor al de los bebés nacidos a término. Esto se debe a que no han tenido suficiente tiempo para crecer y desarrollarse en el útero. Los bebés postmaduros también pueden tener un peso menor al de los bebés nacidos a término. Esto se debe a que la placenta puede envejecer y dejar de proporcionar suficientes nutrientes al bebé. Las madres más jóvenes suelen tener bebés con un peso mayor que las madres de mayor edad, esto se debe a que las madres más jóvenes tienen una mejor salud general y una dieta más equilibrada. Además, sus cuerpos están más preparados para el embarazo y el parto.

En el primer conjunto de datos contamos con 10 variables, para realizar el modelo se selecciona unas variables explicativas. Se comienza tomando todas las variables dentro el modelo y se va quitando las variables menos significativas como es

el caso de las variables 'Genero', 'Edad del padre', 'APGAR 1', 'consultas prenatales', 'Hijos nacidos', 'Numero de embarazos' estas variables no aportan en el modelo, ya que su correlación es muy baja. Por otro lado, las variables 'Talla', 'Tiempo de gestación', 'Edad de la madre' son significativas, sin embargo la variable 'Talla' no afecta nada al modelo y por lo anterior se toman las variables 'Tiempo de gestación' y 'Edad de la madre' como las variables explicativas, puesto que son más vitales estas variables para el contexto de nuestro modelo en estudio.

## 7.2. Dataset ICFES

A continuación, se ilustra una tabla con las 19 variables que se tiene en el segundo conjunto de datos, junto con una breve explicacion de cada una:

| Variables de la base de datos Saber 11 del año 2020 − 2 | | |
|---|---|---|
| Variable | Tipo | Significado |
| Género | Cualitativa | Identificación del estudiante. |
| Puntaje lectura critica | Cuantitativa discreta | Cantidad de puntos obtenidos en la prueba de lectura crítica. |
| Percentil lectura critica | Cuantitativa discreta | Valor que indica el porcentaje del estudiante que presento la prueba de lectura crítica. |
| Desempeño lectura critica | Cuantitativa discreta | Rendimiento del estudiante en la prueba de lectura crítica. |
| Puntaje matemáticas | Cuantitativa discreta | Cantidad de puntos obtenidos en la prueba de matemáticas. |
| Percentil matemáticas | Cuantitativa discreta | Valor que indica el porcentaje del estudiante que presento la prueba de matemáticas. |

| Variables de la base de datos Saber 11 del año 2020 − 2 | | |
|---|---|---|
| Variable | Tipo | Significado |
| Desempeño matemáticas | Cuantitativa discreta | Rendimiento del estudiante en la prueba de matemáticas. |
| Puntaje ciencias. naturales | Cuantitativa discreta | Cantidad de puntos obtenidos en la prueba de ciencias naturales. |
| Percentil ciencias naturales | Cuantitativa discreta | Valor que indica el porcentaje del estudiante que presento la prueba de ciencias naturales. |
| Desempeño ciencias naturales | Cuantitativa discreta | Rendimiento del estudiante en la prueba de ciencia naturales. |
| Puntaje sociales ciudadanas | Cuantitativa discreta | Cantidad de puntos obtenidos en la prueba de sociales ciudadanas. |
| Percentil sociales ciudadanas | Cuantitativa discreta | Valor que indica el porcentaje del estudiante que presento la prueba de sociales ciudadanas. |

| Variables de la base de datos Saber 11 del año 2020 − 2 | | |
|---|---|---|
| Variable | Tipo | Significado |
| Desempeño sociales ciudadanas | Cuantitativa discreta | Rendimiento del estudiante en la prueba de sociales ciudadanas. |
| Puntaje inglés | Cuantitativa discreta | Cantidad de puntos Obtenidos en la prueba de inglés. |
| Percentil inglés | Cuantitativa discreta | Valor que indica el porcentaje del estudiante que presento la prueba de inglés. |
| Desempeño inglés | Cuantitativa discreta | Rendimiento del estudiante en la prueba de inglés. |
| Puntaje global | Cuantitativa discreta | Valor total de puntos en todo el examen. |
| Percentil global | Cuantitativa discreta | Valor que indica el porcentaje del estudiante que presento el examen. |
| Estudiante Generación E | Cualitativa | Si el estudiante alcanza el beneficio. |

En la segunda base de datos se predice el 'Percentil global' con respecto al 'Puntaje lectura critica' y 'Puntaje matemáticas'. El percentil en el ICFES es una medida que indica el lugar que ocupa un estudiante en relación con el resto de los estudiantes que presentaron la prueba. El percentil se calcula asignando a cada estudiante un puntaje en cada una de las cinco áreas de conocimiento: lectura crítica, matemáticas, ciencias naturales, ciencias sociales e inglés. Los puntajes de lectura crítica y matemáticas son los más importantes para el percentil porque estas dos áreas de conocimiento son fundamentales para el éxito académico y profesional.

La capacidad de realizar una lectura crítica implica el entendimiento y análisis minucioso del texto en cuestión, siendo crucial para alcanzar un nivel óptimo durante la educación superior. Resulta muy fundamental que los estudiantes tengan la habilidad necesaria para comprender textos complejos y variados dentro de numerosas disciplinas académicas. Por otra parte, las matemáticas son consideradas como herramientas centrales del razonamiento lógico iniciando procesos resolutivos ante problemas diversos.

Todas estas destrezas se convierten imprescindibles a efecto trascender con éxito diversas áreas laborales que van desde ingeniería hasta negocios o economía entre otras múltiples opciones profesionales disponibles hoy en día. En general, se tienen que los estudiantes que obtienen buenos puntajes en lectura crítica y matemáticas también obtienen buenos puntajes en las demás pruebas. Por lo tanto, el ICFES otorga un mayor peso a los puntajes de lectura crítica y matemáticas al calcular el percentil [3].

En el conjunto de datos del ICFES, hay un total de 19 variables. Para construir el modelo en cuestión, se eligen algunas variables explicativas específicas. Se comienza incluyendo todas las variables en el modelo y luego se van eliminando aquellas que resultan menos significativas como es el caso de las variables 'Genero', 'Estudiantes Generación E' estas variables no tienen un impacto significativo en el modelo para predecir el percentil global. Por otro lado, las variables que se tiene puntaje, desempeño y percentil de cada una de las áreas que se evalúa en el ICFES se tienen en cuenta, sin embargo como se desea predecir el percentil global para ello se necesitara solo las variables de puntaje es decir las de desempeño y percentil no son significativas para el modelo. Por lo dicho anterior de las áreas de lectura crítica y matemáticas se tiene que las variables que mejor se ajustan al modelo son puntaje de lectura critica y puntaje de matemáticas las cuales son las más principales para nuestro estudio.

## 7.3. Validación de supuestos

En el primer conjunto de datos de los recién nacidos de Buga del año 2016. Se quiere determinar el peso del recién nacido con respecto a las semanas de gestación y la edad de la madre, para ello se hará una validación de supuestos y se utilizara solo los datos de validación. Uno de los supuestos es la homocedasticidad lo que significa que la variación de los residuos es constante. Como se observa en la gráfica 6 y 7 los residuos están dispersos y no identifican ningún patrón obvio, es decir, tenemos que la varianza es constante, es decir, Ho :$Var(\varepsilon_i) = \sigma^2$ para todo $i = 1, ..., n$ , Ha :$Var(\varepsilon_j) \neq Var(\varepsilon_i)$ para algunos $i, j$.

Se plantea una prueba de hipótesis usando el test de White, donde la hipótesis nula Ho propone la presencia de homocedasticidad, mientras que la hipótesis alternativa Ha sugiere la presencia de heterocedasticidad. Primero se aplica un análisis de datos para así obtener el peso estimado y los residuos. El test requiere hacer un modelo auxiliar para los residuos al cuadrado y el peso estimando, con este modelo tenemos un coeficiente de determinación del modelo auxiliar de 0.0021012. Luego, se calcula el estadígrafo que es la multiplicación de las observaciones por el coeficiente de determinación auxiliar el cual es 2.2020.

Para finalizar, se determina el estadístico de prueba, que es bajo una distribución chi-cuadrado con un nivel de confianza de 0.95 obteniendo así un valor de 3.8414 con esto se acepta Ho, ya que la distribución de chi-cuadrado es mayor al estadístico de prueba. Por lo tanto no se rechaza la homocedasticidad de los residuos del modelo mediante la prueba de White el resultado prueba que no hay evidencia de heterocedasticidad. Lo anterior se observa que en las figuras 6 y 7, donde se aprecia los residuos no tienen ningún patrón, lo cual respalda la conclusión del test de White.

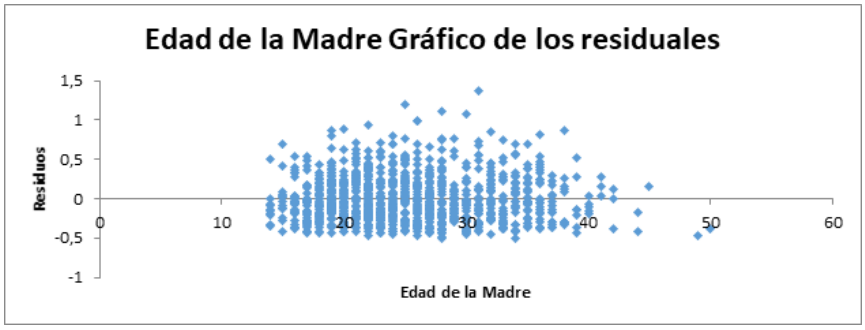


Figura 6: Gráfica de los residuales.
Fuente: Elaboración propia.

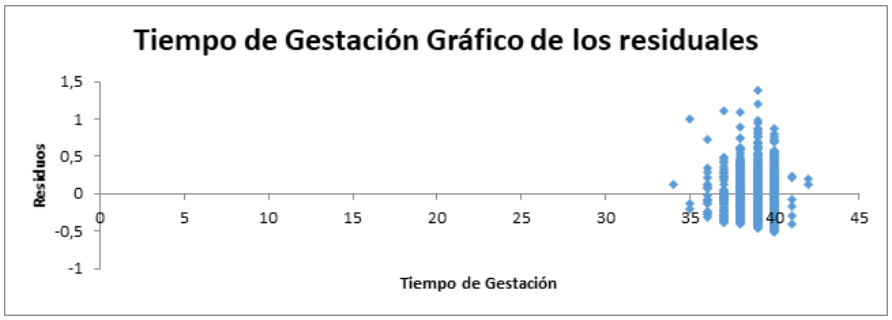


Figura 7: Gráfica de los residuales.
Fuente: Elaboración propia.

Ahora observaremos el supuesto de normalidad, es decir, los residuos siguen la distribución de probabilidad normal. En la figura 8 se observa que es de manera ascendente y presenta una leve normalidad, ya que en la base de datos que manejamos son muchos datos.

Para lo anterior se usa el test de Shapiro Wilks donde Ho es que los datos tienen una distribución normal y Ha es que los datos no tienen una distribución normal. Aplicando el test para evaluar la normalidad de la distribución de las variables peso, tiempo de gestación y edad de la madre. los coeficientes obtenidos son 0.9498,

0.8816 y 0.9683, respectivamente. Comparando estos valores con un valor de referencia de 0.9, se observa que las tres variables presentan valores SW inferiores al umbral, lo que sugiere que sus distribuciones no se ajustan bien a una distribución normal.

Además, se calcularon los valores p para cada variable, obteniendo 0.03, 0.001 y 0.01, respectivamente. Por lo tanto como todos los valores de p son menores que el nivel de significancia $\alpha = 0.05$, se rechaza la hipótesis nula de normalidad.

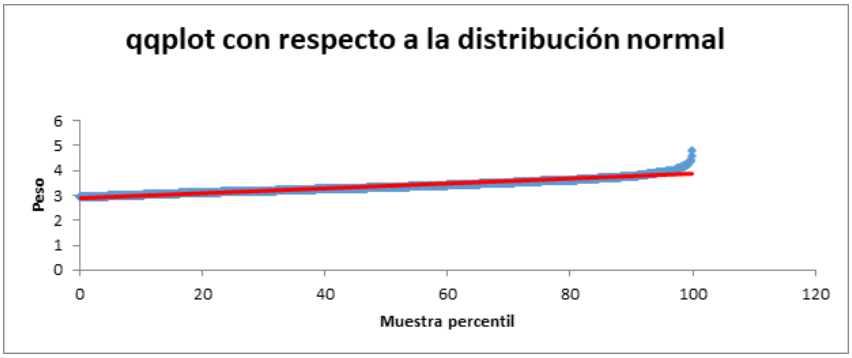


Figura 8: Gráfica qqplot con respecto a la distribución normal.
Fuente: Elaboración propia.

Por último, se observa si los residuos son independientes. Para ello la independencia se da si no se observa un patrón de puntos ascendentes o descendentes. Como se examina en la figura 9 se nota un patrón descendiente leve, luego se concluye que no son independientes los residuos, por otro lado se tiene en cuenta que son demasiados datos que se manejan. Sin embargo, como este trabajo se enfoca más en el aprendizaje automático y en ajustar el mejor modelo a los datos se tiene que la validación de supuestos no es perfecta.

Para ello, se plantea una prueba de hipótesis usando el test de Durbin Watson, donde Ho es que no existe autocorrelación serial y Ha si existe autocorelacción serial. Aplicando el test primero se hace un análisis de datos obteniendo así los residuos, luego se calcula los residuos al cuadrado y los residuos resegados en un periodo. Seguidamente, se calcula el DW (Durbin Watson) que nos da como resultado de 0.08973 en un modelo con bastantes datos, no se puede rechazar la hipótesis nula de independencia de los residuos al nivel de significancia de 5 %. Sin embargo, este valor es inferior a 2 indica que existe cierta evidencia de autocorrelación.

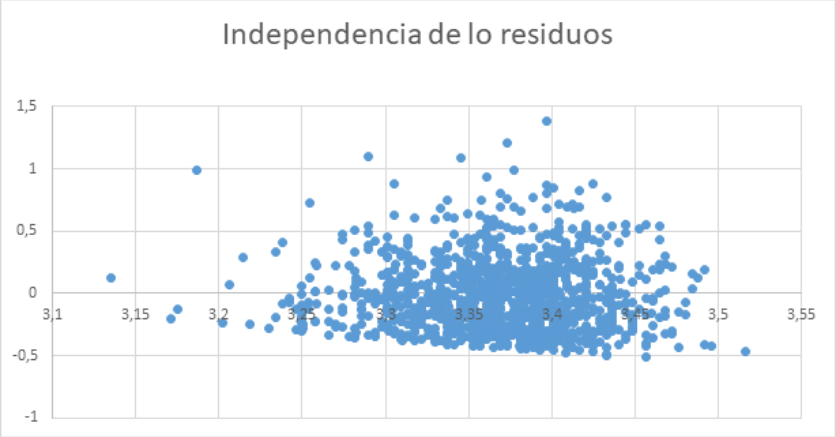


Figura 9: Gráfica de independencia de los residuos.
Fuente: Elaboración propia.

En la segunda base de datos de la prueba saber 11 del año $2020-2$. Se estudia el percentil global de la prueba con respecto al puntaje de lectura crítica y matemáticas. En el supuesto de homocedasticida tenemos que en las figuras 10 y 11 se nota que no tiene ningún patrón, por lo tanto se llega que la varianza es constante, es decir, Ho :$Var(\varepsilon_i) = \sigma^2$ para todo $i = 1, ..., n$ , Ha :$Var(\varepsilon_j) \neq Var(\varepsilon_i)$ para algunos $i, j$.

Se plantea una prueba de hipótesis usando el test de White, donde la hipótesis nula Ho propone la presencia de homocedasticidad, mientras que la hipótesis alternativa Ha sugiere la presencia de heterocedasticidad. Primero se aplica un análisis de datos para así obtener el peso estimado y los residuos. El test requiere hacer un modelo auxiliar para los residuos al cuadrado y el peso estimando, con este modelo tenemos un coeficiente de determinación del modelo auxiliar de 0.0004968.

Luego, se calcula el estadístico de prueba que es la multiplicación de las observaciones por el coeficiente de determinación auxiliar el cual es 11.0037. Para finalizar, se hace la distribución chi-cuadrado con un nivel de confianza de 0.95 obteniendo así un valor de 3.8414, comparando este valor con el valor critico de la distribución chi-cuadrado se observa que el estadístico de prueba del test de white es mayor que el valor critico.

Por lo tanto, se rechaza la hipótesis nula de ausencia de heterocedasticidad y se concluye que existe evidencia de heterocedasticidad en el modelo. Sin embargo, en las figuras 10 y 11 se nota que no posee ningún patrón pero con el test se muestra evidencia de heterocedasticidad, esto sucede por la cantidad de datos que se manejan en esta base de datos.

Ahora observaremos otro supuesto que es si posee normalidad, es decir, los residuos siguen la distribución de probabilidad normal. En la figura 12 se observa que es de manera ascendente.

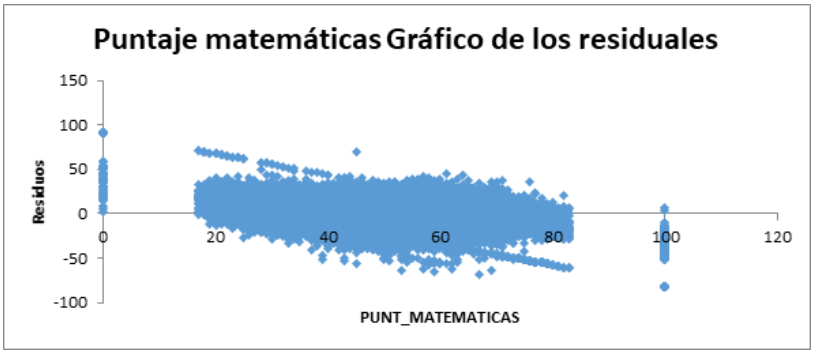


Figura 10: Gráfica de los residuales.
Fuente: Elaboración propia.

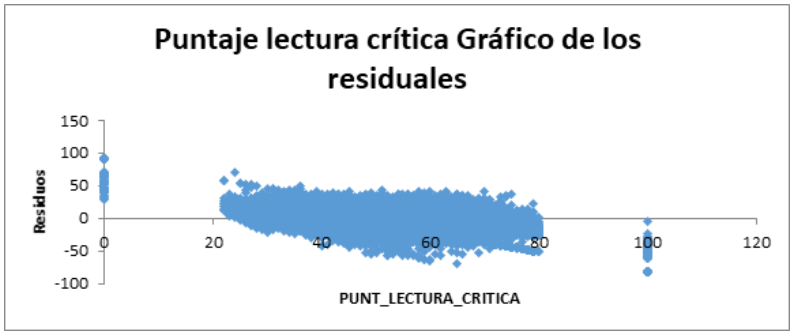


Figura 11: Gráfica de los residuales.
Fuente: Elaboración propia.

Para lo anterior se usa el test de Shapiro Wilks, donde Ho es que los datos tienen una distribución normal y Ha es que los datos no tienen una distribución normal. Aplicando el test para evaluar la normalidad de la distribución de las variables percentil global, puntaje de matemáticas y puntaje de lectura crítica. Los coeficientes obtenidos son 0.9608, 0.9914 y 0.9922 respectivamente. Comparando estos valores con un valor de referencia de 0.9, se observa que los coeficientes son superior al umbral, lo que indica que su distribución se ajusta mejor a una distribución normal. Sin

embargo, los valores de p son 0.0002, 0.001 y 0.0001 respectivamente, lo cual rechaza la hipótesis nula de normalidad.

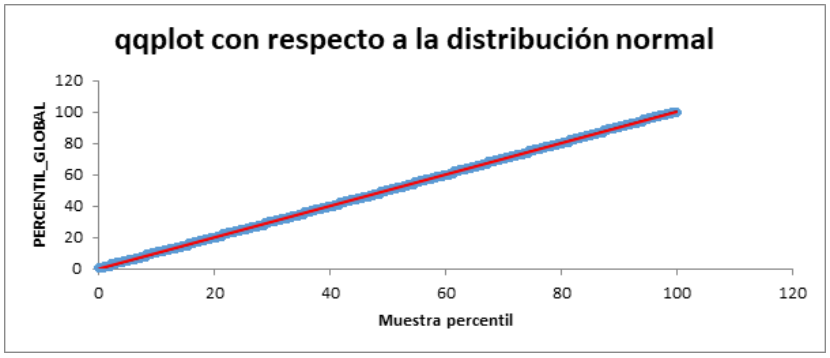


Figura 12: Gráfica qqplot con respecto a la distribución normal.
Fuente: Elaboración propia.

Por último, se observa si los residuos son independientes. Para ello la independencia se da si no se observa un patrón de puntos ascendentes o descendentes. Como se examina en la figura 13 se nota un patrón descendiente, luego se concluye que no son independientes los residuos, ya que no se puede observar bien por ser demasiados datos que se manejan en este conjunto de datos. Sin embargo, como este trabajo se enfoca más en el aprendizaje automático y en ajustar el mejor modelo a los datos se tiene que la validación de supuestos no es perfecta.

Para ello, se plantea una prueba de hipótesis usando el test de Durbin Watson, donde Ho es que no existe autocorrelación serial y Ha si existe autocorelacción serial. Aplicando el test primero se hace un análisis de datos obteniendo así los residuos, luego se calcula los residuos al cuadrado y los residuos resegados en un periodo. Seguidamente, se calcula el DW (Durbin Watson) el cual nos dio como resultado de

2.045 en un modelo con bastantes datos, no se puede rechazar la hipótesis nula de independencia de los residuos al nivel de significancia de 5 %. Sin embargo, este valor es muy cerca a 2 lo que significa que no hay suficiente evidencia para determinar si existe o no autocorrelación en los residuales.

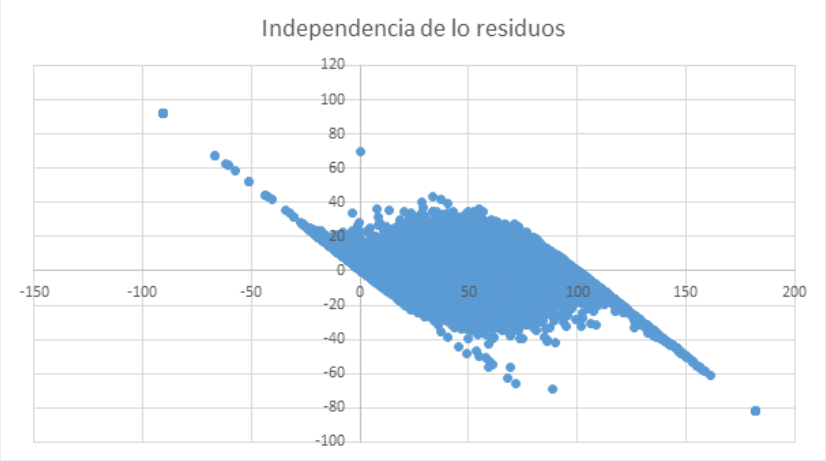


Figura 13: Gráfica de independencia de los residuos.
Fuente: Elaboración propia.

Cabe recalcar que nuestro trabajo se enfoca más en tener un buen ajusto de los tres modelos en estudio el lineal, Ridge y Lasso dependiendo del que tenga menor error y así utilizar el mejor modelo con aprendizaje de maquina.

# 8. Resultados

## 8.1. Estadísticas vitales Buga 2016: Nacimientos

En la base de datos 'Recién nacidos Guadalajara de Buga 2016', se tienen las siguientes variables: 'Genero' es una variable cualitativa y es la identificación del sexo

de los recién nacidos; 'Peso' es una variable cuantitativa discreta, la cual describe el peso del recién nacido expresado en kilos; 'Talla' es una variable cuantitativa discreta que representa la altura del bebé dada en centímetros; 'Tiempo de gestación' es una variable cuantitativa discreta, en la que se expresan las semanas de gestación. Además, esta base posee otras variables tales como: 'Consultas prenatales' es una variable cuantitativa discreta, a esta se le atribuye la cantidad de consultas médicas a las que la madre asistió; 'APGAR 1' es una variable cuantitativa discreta, manifiesta el examen rápido que se le realiza al bebé durante el primer y quinto minuto después del nacimiento; 'Edad de la madre' es una variable cuantitativa discreta medida en años; 'Hijos nacidos' es una variable cuantitativa discreta que declara la cantidad de hijos de la madre; 'Número de embarazos' es una variable cuantitativa discreta, tal como lo describe su nombre, es la cantidad de embarazos que la madre ha tenido; por último, 'Edad del padre' es una variable cuantitativa discreta medida en años.

Estos datos se tomaron de la Alcaldía Municipal de Guadalajara de Buga, entidad encargada de su suministro y propietaria de estos datos de salud y protección social. (https://www.datos.gov.co/d/u5y2-ufx9)

### 8.1.1. Regresión Lineal

Los datos que se emplean corresponden a la de recién nacidos de Buga del año 2016. Nos centraremos en tres variables las cuales son el peso del recién nacido (kg), edad de la madre y tiempo de gestación (semanas). Se quiere determinar si el peso de un recién nacido aumenta a medida que lo hace tanto el tiempo de gestación y la edad

de la madre durante el embarazo. Primero se observa los datos de entrenamiento y de validación donde se utiliza la función `train_test_split` el criterio corresponde a seleccionar el 30 % de los datos para prueba de manera aleatoria. Ahora se entrena el modelo con los datos de entrenamiento escogidos. Por último, se utiliza el modelo de regresión lineal. En este caso dado por:

$$\text{Peso} = \beta_1 \text{Tiempo de gestación} + \beta_2 \text{Edad de la madre} + \varepsilon,$$

donde el error tiene una distribución normal $\varepsilon \sim \text{Normal}(0, \sigma^2)$. Para seleccionar los datos de entrenamiento y de validación con lenguaje `Python` quedaría de la siguiente manera:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test, z_train, z_test= train_test_split(tabla1["Peso"],
tabla1["Tiempo de Gestaci n"],
tabla1["Edad de la Madre"],
test_size = 0.3)
print(x_train, y_train, z_train)
yz_train = pd.concat([y_train, z_train], axis=1)
print(yz_train)
yz_test = pd.concat([y_test, z_test], axis=1)
```

Después de escoger los datos de entrenamiento y validación, se procede a entrenar el modelo con los datos de entrenamiento con la regresión lineal.

```
import numpy as np
from sklearn.linear_model import LinearRegression
modelo = LinearRegression()
modelo.fit(yz_train, x_train)
```

Luego, se puede determinar los parámetros bajo el modelo de regresión lineal simple. También se determinar las medidas de ajuste sobre el parámetro.

```
print("El intercepto del modelo es: ", modelo.intercept_)
print("Coeficiente:", list(zip(yz_train, modelo.coef_.flatten(), )))
print("Coeficiente de determinaci n R^2:", modelo.score(yz_test, x_test))
from sklearn.metrics import mean_squared_error
predicciones = modelo.predict(yz_test)
print(predicciones)
rmse = mean_squared_error(y_true  = x_test, y_pred  = predicciones, squared = True)
print("El cuadrado medio del error es: ", rmse)
```

De lo anterior obtenemos que el intercepto del modelo es 2.2751, el Coeficiente de determinación, $R^2$, es 0.0473, el error cuadrático medio (ECM) es 0.0382, el coeficiente del tiempo de gestación es 0.034093373 y el coeficiente de la edad de la madre es $-0.000608924266$. Por último, presentamos en la Figura 14 el gráfico de dispersión y el plano de regresión correspondiente, donde se muestra los datos de entrenamiento (azul) y los de prueba (púrpura).

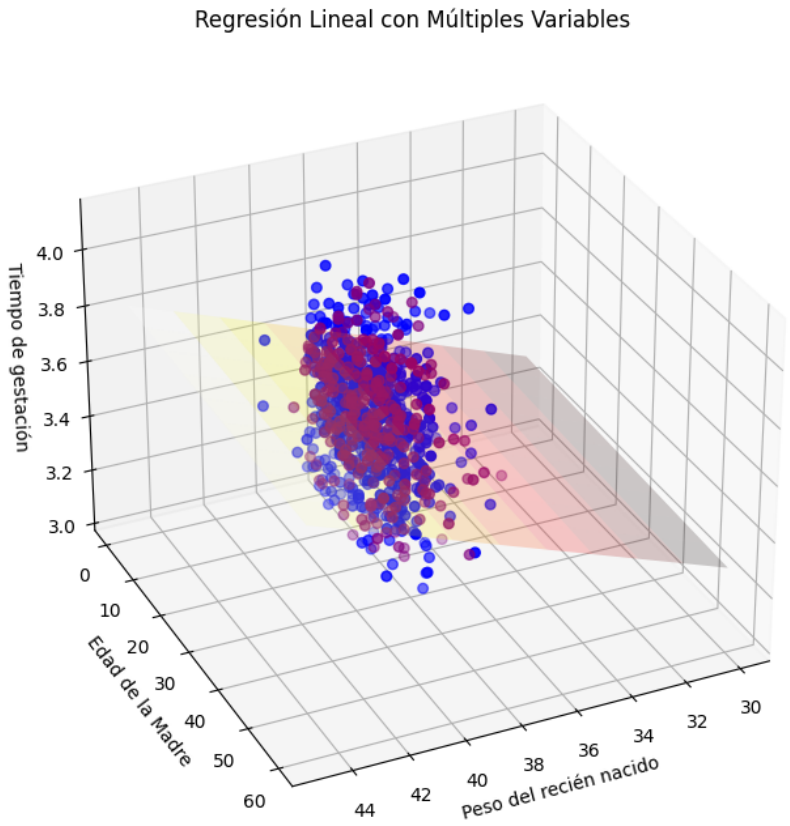


Figura 14: Gráfica de dispersión en el plano de regresión.
Fuente: Elaboración propia.

### 8.1.2. Regresión Ridge

El modelo de regresión Ridge esta dado por:

$$\text{Peso} = \beta_1 \text{Tiempo de gestación} + \beta_2 \text{Edad de la madre} + \lambda \|\beta\|_2^2 + \varepsilon.$$

Para esta regresión se toma en cuenta que $\lambda$ determina el peso del la penalización, en donde se utilizara validación cruzada para encontrar $\lambda$ que cumpla $0 < \lambda < \infty$ con esto se determina cual valor tiene la menor varianza. Primero, se crea un vector de lambda, que tiene 100 entradas que van de 10 a $-2$.

```
print(np.linspace(10,-2,100)[:10])
lambdas = 10**np.linspace(10,-2,100)*0.5
print(lambdas[:10])
print(lambdas[-10:])
```

Luego se hace la división de los datos de entrenamiento y los de validación utilizando el 30 % de los datos para prueba.

```
from sklearn.model_selection import train_test_split
x_train, x_test,y_train, y_test,z_train, z_test= train_test_split(tabla1["Peso"],
tabla1["Tiempo de Gestaci n"],
tabla1["Edad de la Madre"],
test_size = 0.3)
```

Ahora se entrena los datos para esto de define los modelos, es decir, se llena la matriz con coeficientes asociados a cada variable independiente y a cada valor de lambda. Con ayuda de `ridge.set_params(alpha = k )` se toma los valores de lambda, luego tomamos `ridge.fit(yz_train, x_train)` el cual ajusta el modelo requerido y por ultimo se utiliza `coefs.append(ridge.coef_)` generando los coeficientes correspondientes.

```
from sklearn.linear_model import Ridge, RidgeCV
ridge = Ridge( )
coefs = []
for k in lambdas:
  ridge.set_params(alpha = k )
  ridge.fit(yz_train, x_train)
  coefs.append(ridge.coef_)
print(np.shape(coefs))
coefs[0]
```

Luego, se determina los parámetros bajo el modelo de regresión Ridge y los valores de los coeficientes.

```
print("El intercepto del modelo es: ", ridge.intercept_)
print("Coeficiente:", list(zip(yz_train, ridge.coef_.flatten(), )))
ridgecv = RidgeCV(alphas = lambdas, scoring = "neg_mean_squared_error")
ridgecv.fit(yz_train, x_train)
print("El valor de lambda encontrado con cross validation es: ",
round(ridgecv.alpha_,3))
mod_ridgeCV = Ridge(alpha = ridgecv.alpha_, )
mod_ridgeCV.fit(yz_train, x_train)
print(pd.Series(mod_ridgeCV.coef_, index = X.columns))
ECMcv = round(mean_squared_error(x_test, x_pred),3)
print(ECMcv)
```

En la linea `from sklearn.linear_model import Ridge, RidgeCV` importa las clases Ridge y RidgeCV para realizar la regresión Ridge; `ridge = Ridge( )` crea un objeto Ridge sin especificar el parámetro de regularización lambda; `coefs = []` establece una lista vacía para almacenar los coeficientes de los modelos con diferentes lambdas; `for k in lambdas` itera sobre una lista de valores de lambda para probar; `ridge.set_params(alpha = k )` establece el valor de lambda actual para el modelo Ridge; `ridge.fit(yz_train, x_train)` ajusta el modelo a los datos de entrenamiento; `coefs.append(ridge.coef_)` Almacena los coeficientes del modelo en la lista `coefs`; `print(np.shape(coefs))` imprime la forma de la lista `coefs`, que contiene los coeficientes de diferentes modelos; `coefs[0]` Muestra los coeficientes del primer modelo (con el primer valor de lambda).

Para determinar los parámetros del modelo se tiene las siguientes lineas de codigo `ridgecv = RidgeCV(alphas = lambdas, scoring = "neg_mean_squared_error")` crea un objeto RidgeCV para realizar validación cruzada y encontrar el mejor valor

de lambda, donde `alphas = lambdas` es el argumento que define una lista de valores de lambda que se probarán durante la validación cruzada. Cada valor de lambda representa la fuerza que se aplica en la regularización en el modelo Ridge como se presenta, `scoring = "neg_mean_squared_error"` este argumento define la métrica de evaluación que se utiliza para comparar los diferentes modelos durante la validación cruzada. En este caso, se utiliza el error cuadrático medio negativo; `ridgecv.fit(yz_train, x_train)` realiza la validación cruzada para encontrar el lambda óptimo; `mod_ridgeCV = Ridge(alpha = ridgecv.alpha_, )` crea un nuevo modelo Ridge con el lambda óptimo; `mod_ridgeCV.fit(yz_train, x_train)` ajusta el nuevo modelo a los datos de entrenamiento el cual es; `ECMcv = round(mean_squared_error(x_test, x_pred),3)` calcula el error cuadrático medio (ECM) en el conjunto de prueba, por último, se imprime todos los resultados obtenidos. El código utiliza el método de gradiente descendente para minimizar la función de costo regularizada en la regresión Ridge. El algoritmo de gradiente descendente se implementa dentro de la clase `Ridge` de la biblioteca `scikit-learn`. En las demás implementaciones se utiliza la misma analogía en el código en función del modelo en estudio.

De lo anterior se tiene que el intercepto del modelo es 2.0879, el valor de lambda por validación cruzada es 0.03834, el coeficiente de la edad de la madre es 0.03834 y el de tiempo de gestación es $-0.002$, lo que significa que cuando el coeficiente es positivo tiene influencia en el modelo. Sin embargo, el valor obtenido es muy pequeño por ello no aporta suficiente información. Por otro lado, el otro coeficiente es negativo por consiguiente tiene efecto negativo en el modelo. Por último,

se calcula el error cuadrático medio dando como resultado 0.031, así mismo se ilustra la gráfica 15 de dispersión en el plano de regresión correspondiente, donde se muestra los datos de entrenamiento (azul) y los de prueba (púrpura).

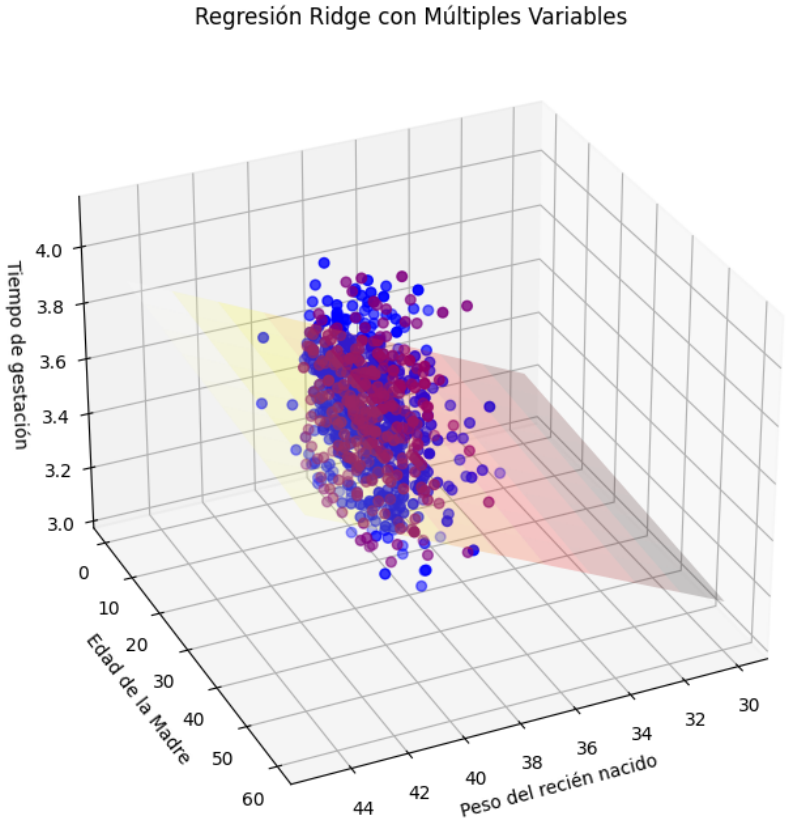


Figura 15: Gráfica de dispersión en el plano de regresión.
Fuente: Elaboración propia.

### 8.1.3. Regresión Lasso

El modelo de regresión Lasso esta dado por:

$$\text{Peso} = \beta_1 \text{Tiempo de gestación} + \beta_2 \text{Edad de la madre} + \lambda\|\beta\|_1 + \varepsilon.$$

Los mismos lambdas obtenidos en la regresión Ridge se mantienen, lo único que cambia es el entrenamiento con Lasso. Después de esto, los datos se dividen entre conjuntos de entrenamiento y validación utilizando un porcentaje del 30 % para pruebas.

```
from sklearn.model_selection import train_test_split
x_train, x_test,y_train, y_test,z_train, z_test= train_test_split(tabla1["Peso"],
tabla1["Tiempo de Gestaci n"], tabla1["Edad de la Madre"],
test_size = 0.3)
```

Ahora se evalúa el modelo de regresión Lasso y se determina los parámetros, valores de los coeficientes.

```
from sklearn.linear_model import Lasso, LassoCV
lasso = Lasso(max_iter = 10000)
coefs = []
for k in lambdas:
    lasso.set_params(alpha = k)
    lasso.fit(scale(yz_train), x_train)
    coefs.append(lasso.coef_)
np.shape(coefs)
print("El intercepto del modelo es: ", lasso.intercept_)
print("Coeficiente:", list(zip(yz_train, lasso.coef_.flatten(), )))
lassocv = LassoCV(alphas = None, cv = 10, max_iter = 100000)
lassocv.fit(yz_train, x_train)
print(pd.Series(lassocv.coef_, index = X.columns))
x_pred = lasso.predict(yz_test)
```

```
x_pred
ECMcv = round(mean_squared_error(x_test, x_pred),3)
print(ECMcv)
```

Por lo tanto, se tiene que el intercepto del modelo es 3.5961, el coeficiente de la edad de la madre es 0.032 y el de tiempo de gestación es $-0.001$, en esta regresión la variable tiempo de gestación afecta de manera negativa al modelo y se hace casi cero. Con Lasso se calcula un error cuadrático medio de 0.038, luego se ilustra la gráfica 16 de dispersión en el plano de regresión correspondiente, donde se muestra los datos de entrenamiento (azul) y los de prueba (púrpura).

En conclusión de las tres regresiones se tiene lo siguiente:

| | Regresión Lineal | Regresión Ridge | Regresión Lasso |
|---|---|---|---|
| Intercepto | 2.2751 | 2.0879 | 3.5961 |
| $R^2$ | 0.047 | 0.025 | 0.045 |
| ECM | 0.038 | 0.031 | 0.038 |

Para esta primera base de datos de los recién nacidos, se puede decir que el modelo que mejor se ajusta para estos datos es Ridge con un error cuadrático medio de 0.031, debido a que es el menor entre las regresiones lineal y Lasso. En cambio el modelo lineal y Lasso tienen el mismo ECM de 0.038, pero no se recomienda para esta base por ser mas alto que el de Ridge con este modelo se reduce el sobreajuste sin perder precisión. Sin embargo, un método que podemos mirar como alternativo es la regresión kernel.

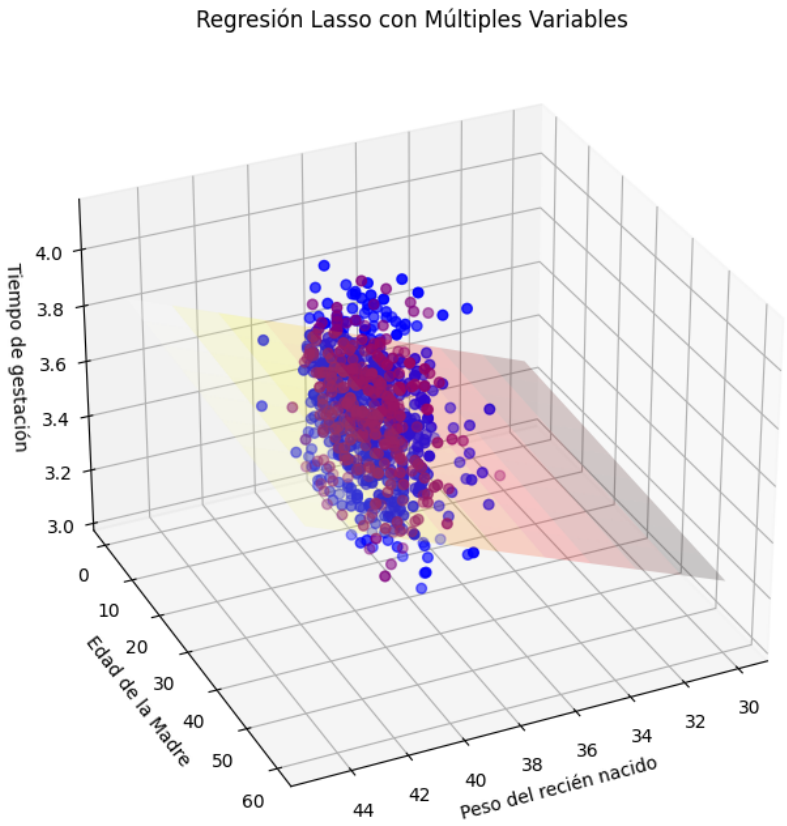


Figura 16: Gráfica de dispersión en el plano de regresión.
Fuente: Elaboración propia.

## 8.2. Datos del ICFES

En la base de datos de 'Saber 11 del año 2020-2' poseemos las siguientes variables: 'Genero' la cual es una variable cualitativa y es la identificación del estudiante; 'Puntaje de lectura critica' es una variable cuantitativa discreta y es la cantidad de puntos obtenidos en la prueba de lectura crítica; 'Percentil lectura critica' es una variable cuantitativa discreta y es el valor que indica el porcentaje del estudiante en la prueba de la sección correspondiente; 'Desempeño lectura critica' es una variable

cuantitativa discreta el cual indica el rendimiento del estudiante en lectura crítica; 'Puntaje matemáticas' es una variable cuantitativa discreta y es la cantidad de puntos obtenido en la prueba de matemáticas; 'Percentil matemáticas' es una variable cuantitativa discreta y es el porcentaje que el estudiante obtuvo en matemáticas; 'Desempeño matemáticas' es una variable cuantitativa discreta y es el rendimiento en la prueba de matemáticas; 'Puntaje ciencias naturales' es una variable cuantitativa discreta y es la cantidad de puntos obtenidos en la prueba de ciencias naturales; 'Percentil ciencias naturales' es una variable cuantitativa discreta y es el porcentaje que el estudiante obtuvo en ciencias naturales; 'Desempeño ciencias naturales' es una variable cuantitativa discreta y es el rendimiento del estudiante en la prueba de ciencias naturales; 'Puntaje sociales ciudadanas' es una variable cuantitativa discreta y es la cantidad de puntos obtenidos en la prueba de sociales ciudadanas; 'Percentil sociales ciudadanas' es una variable cuantitativa discreta y es el porcentaje que el estudiante obtuvo en sociales ciudadanas; 'Desempeño sociales ciudadanas' es una variable cuantitativa discreta y es el rendimiento del estudiante en la prueba de sociales ciudadanas; 'Puntaje inglés' es una variable cuantitativa discreta y es la cantidad de puntos obtenidos en la prueba de inglés; 'Percentil inglés' es una variable cuantitativa discreta y es el porcentaje que el estudiante obtuvo en inglés; 'Desempeño inglés' es una variable cuantitativa discreta y es el rendimiento del estudiante en la prueba de inglés; 'Puntaje global' es una variable cuantitativa discreta y es el valor total de los puntos obtenidos en todo el examen; 'Percentil global' es una variable cuantitativa discreta y es el valor que indica el porcentaje del estudiante que presentó el examen; 'Estudiante Generación E' es una variable

cuantitativa discreta y es un beneficio si el estudiante alcanza un puntaje óptimo.

Estos datos se recolectaron de ICFES DATOS ABIERTOS que son los propietarios y el Instituto Colombiano para la Evaluación de la Educación - ICFES es la entidad que lo suministra. (http://www.icfes.gov.co/normatividad)

### 8.2.1. Regresión Lineal

Para la segunda implementación los datos que se manejan son los de la prueba saber 11 del año $2020-2$. El análisis se centra en tres variables las cuales son el percentil global, puntaje de lectura critica y el puntaje de matemáticas. Se quiere determinar si el percentil global de un estudiante mejora con respecto a los puntajes de lectura crítica y matemáticas. Primero, se observa los datos de entrenamiento y de validación luego se utiliza la función `train_test_split` donde el criterio corresponde a seleccionar el $30\,\%$ de los datos para prueba de manera aleatoria. Seguidamente se entrena el modelo con los datos de entrenamiento escogidos. Finalmente, se utiliza el modelo de regresión lineal. En este caso dado por:

$$\text{Percentil global} = \beta_1 \text{Puntaje lectura critica} + \beta_2 \text{Puntaje de matemáticas} + \varepsilon,$$

donde el error tiene una distribución normal $\varepsilon \sim \text{Normal}(0, \sigma^2)$. Para seleccionar los datos de entrenamiento y de validación con lenguaje `Python` quedaría de la siguiente manera:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test, z_train, z_test
= train_test_split(tabla1["PERCENTIL_GLOBAL"],
```

```
tabla1["PUNT_LECTURA_CRITICA"],
tabla1["PUNT_MATEMATICAS"],
test_size = 0.3)
print(x_train, y_train, z_train)
yz_train = pd.concat([y_train, z_train], axis=1)
print(yz_train)
yz_test = pd.concat([y_test, z_test], axis=1)
```

Se procede a entrenar el modelo con los datos de entrenamiento con el modelo de regresión lineal. Luego se determina los parámetros bajo el modelo y las medidas de ajuste sobre el parámetro.

```
import numpy as np
from sklearn.linear_model import LinearRegression
modelo = LinearRegression()
modelo.fit(yz_train, x_train)
print("El intercepto del modelo es:", modelo.intercept_)
print("Coeficiente:", list(zip(yz_train, modelo.coef_.flatten(), )))
print("Coeficiente de determinaci n R^2:", modelo.score(yz_test, x_test))
from sklearn.metrics import mean_squared_error
predicciones = modelo.predict(yz_test)
print(predicciones)
rmse = mean_squared_error(y_true  = x_test, y_pred  = predicciones, squared = True)
print("El cuadrado medio del error es:", rmse)
```

De lo anterior se tiene que el intercepto del modelo es $-91.46$, el coeficiente de determinación $R^2$ es 0.88, el coeficiente del puntaje de lectura crítica es 1.53260788, el coeficiente del puntaje de matemáticas es 1.20687343 y el error cuadrático medio es 92.81. Por último, se hace el gráfico 17 de dispersión en el plano de regresión correspondiente, donde se muestra los datos de entrenamiento (azul) y los de prueba (púrpura).

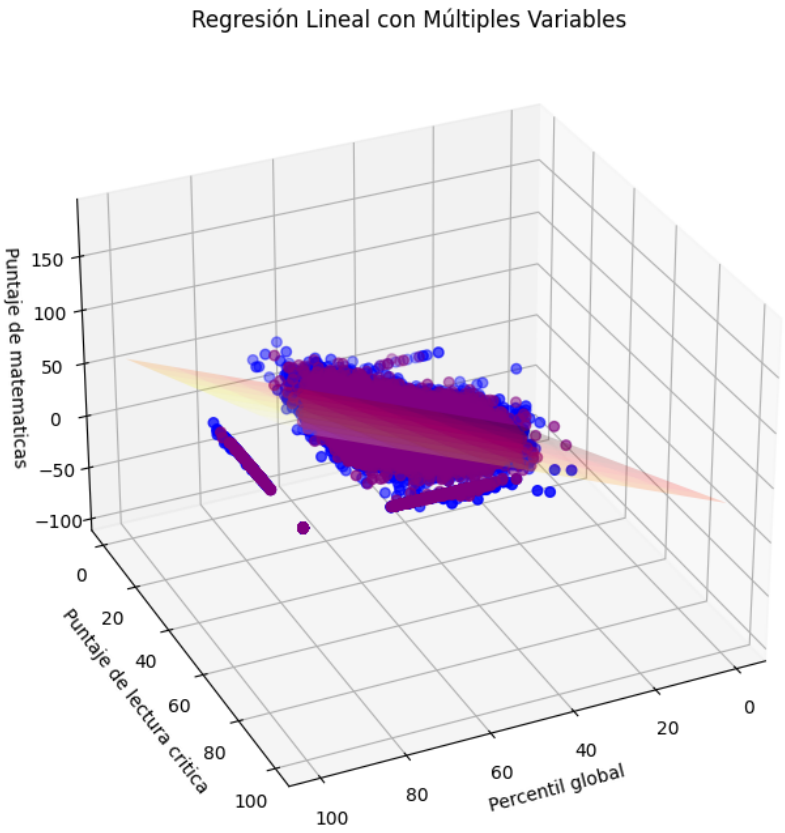


Figura 17: Gráfica de dispersión en el plano de regresión.
Fuente: Elaboración propia.

### 8.2.2. Regresión Ridge

El modelo de regresión Ridge esta dado por:

$$\text{Percentil global} = \beta_1 \text{Puntaje lectura critica} + \beta_2 \text{Puntaje de matemáticas} + \lambda \|\beta\|_2^2 + \varepsilon.$$

Se debe considerar que el valor de $\lambda$ determina la magnitud de penalización. Para encontrar un valor adecuado, se utilizará validación cruzada y se establecerá una restricción en $0 < \lambda < \infty$. De esta manera, se puede identificar cuál es el valor

con menor varianza. Para comenzar este proceso, luego se crea un vector conteniendo 100 valores lambda que van desde 10 hasta $-2$.

```
print(np.linspace(10,-2,100)[:10])
lambdas = 10**np.linspace(10,-2,100)*0.5
print(lambdas[:10])
print(lambdas[-10:])
```

Después de ello, se procede a dividir los datos en dos grupos: el conjunto de entrenamiento y el conjunto de validación. Para este último, se utiliza una proporción del 30 % del total de los datos disponibles para realizar pruebas.

```
from sklearn.model_selection import train_test_split
x_train, x_test,y_train, y_test,z_train, z_test= train_test_split
(tabla1["PERCENTIL_GLOBAL"],
tabla1["PUNT_LECTURA_CRITICA"], tabla1["PUNT_MATEMATICAS"],
test_size = 0.3)
```

Los datos son sometidos a entrenamiento para definir los modelos. Esto implica el llenado de una matriz con coeficientes asociados tanto a cada variable independiente como también a cada valor de lambda presente. A través del uso de `ridge.set_params(alpha = k )`, se seleccionan estos valores antes mencionados y mediante la ejecución exitosa del comando `ridge.fit(yz_train, x_train)`, se lleva ajustar adecuadamente el modelo deseado.

Finalmente, los correspondientes coeficientes quedan generados tras hacer uso de la función `coefs.append(ridge.coef_)`.

```
from sklearn.linear_model import Ridge, RidgeCV
ridge = Ridge( )
coefs = []
```

```
for k in lambdas:
  ridge.set_params(alpha = k )
  ridge.fit(yz_train, x_train)
  coefs.append(ridge.coef_)
print(np.shape(coefs))
coefs[0]
```

Luego, se determina los parámetros bajo el modelo de regresión Ridge y los valores de los coeficientes.

```
print("El intercepto del modelo es:", ridge.intercept_)
print("Coeficiente:", list(zip(yz_train, ridge.coef_.flatten(), )))
ridgecv = RidgeCV(alphas = lambdas, scoring = "neg_mean_squared_error")
ridgecv.fit(yz_train, x_train)
print("El valor de lambda encontrado con cross validation es:",
round(ridgecv.alpha_,3))
mod_ridgeCV = Ridge(alpha = ridgecv.alpha_, )
mod_ridgeCV.fit(yz_train, x_train)
print(pd.Series(mod_ridgeCV.coef_, index = X.columns))
ECMcv = round(mean_squared_error(x_test, x_pred),3)
print(ECMcv)
```

Según los resultados obtenidos, se tiene que el intercepto del modelo es $-91.5383$, el valor de lambda determinada por validación cruzada es 352.74. Los coeficientes asociados al puntaje de lectura crítica y al puntaje de matemáticas son 1.5326 y 1.2080 respectivamente, lo que significa que estos coeficientes afectan al modelo positivamente. Por último, se calcula el error cuadrático medio el cual es 93.337 y el $R^2$ es 0.88, así mismo presentamos la gráfica 18 de dispersión en el plano de regresión correspondiente, donde se muestra los datos de entrenamiento (azul) y los de prueba (púrpura).

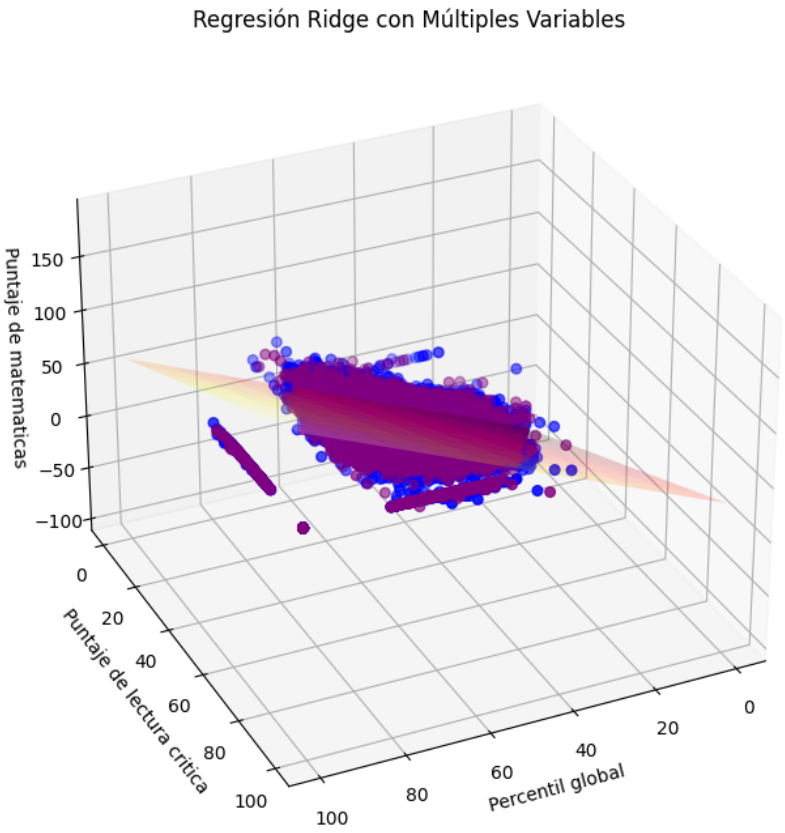


Figura 18: Gráfica de dispersión en el plano de regresión.
Fuente: Elaboración propia.

### 8.2.3. Regresión Lasso

El modelo de regresión Lasso esta dado por:

$$\text{Percentil global} = \beta_1 \text{Puntaje lectura critica} + \beta_2 \text{Puntaje de matemáticas} + \lambda \|\beta\|_1 + \varepsilon.$$

Se mantienen las mismas lambdas obtenidas en la regresión Ridge, con el único cambio del entrenamiento con Lasso. A continuación, los datos se dividen en conjuntos de entrenamiento y validación utilizando un porcentaje 30 % a efectos de prueba.

```
from sklearn.model_selection import train_test_split
x_train, x_test,y_train, y_test,z_train, z_test= train_test_split
(tabla1["PERCENTIL_GLOBAL"],
tabla1["PUNT_LECTURA_CRITICA"], tabla1["PUNT_MATEMATICAS"],
test_size = 0.3)
```

Ahora se evalúa el modelo de regresión Lasso y se determina los parámetros, valores de los coeficientes.

```
from sklearn.linear_model import Lasso, LassoCV
lasso = Lasso(max_iter = 10000)
coefs = []
for k in lambdas:
    lasso.set_params(alpha = k)
    lasso.fit(scale(yz_train), x_train)
    coefs.append(lasso.coef_)
np.shape(coefs)
print("El intercepto del modelo es:", lasso.intercept_)
print("Coeficiente:", list(zip(yz_train, lasso.coef_.flatten(), )))
lassocv = LassoCV(alphas = None, cv = 10, max_iter = 100000)
lassocv.fit(yz_train, x_train)
print(pd.Series(lassocv.coef_, index = X.columns))
x_pred = lasso.predict(yz_test)
x_pred
ECMcv = round(mean_squared_error(x_test, x_pred),3)
print(ECMcv)
```

Por lo tanto, se obtiene que el intercepto del modelo es 49.8501, el valor de lambda por validación cruzada es 0.288, el coeficiente del puntaje de lectura critica es 1.5321 y el puntaje de matemáticas es 1.2059, en esta regresión tenemos que los coeficientes afectan de manera positiva para el modelo. Luego se calcula un error cuadrático medio es 93.746 y el $R^2$ es 0.88. Finalmente se presenta la gráfica 19 de

dispersión en el plano de regresión correspondiente, donde se muestra los datos de entrenamiento (azul) y los de prueba (púrpura).

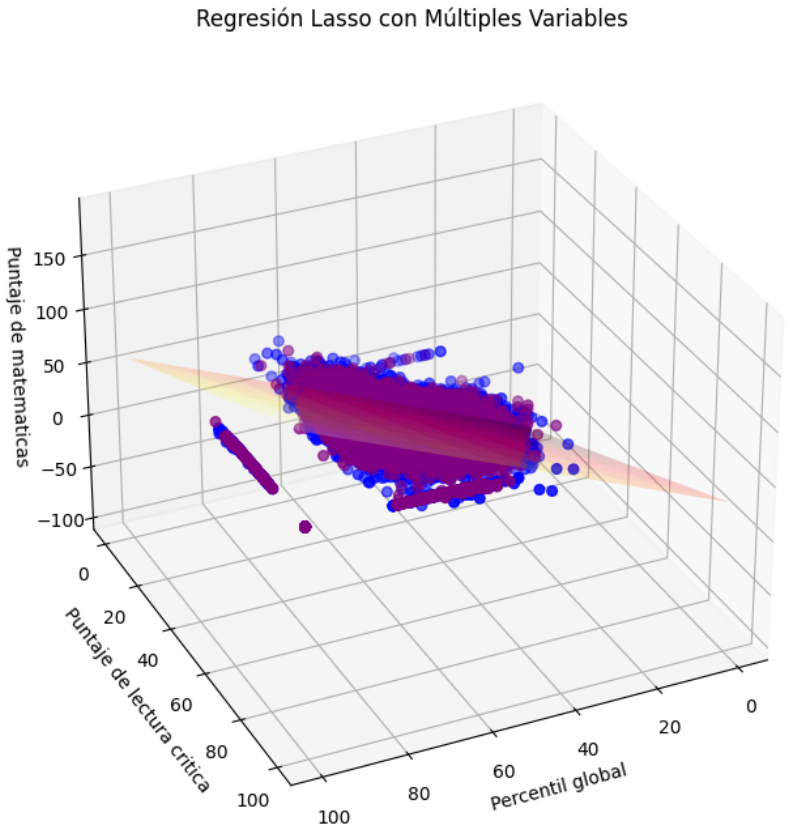


Figura 19: Gráfica de dispersión en el plano de regresión.
Fuente: Elaboración propia.

En conclusión de las tres regresiones se tiene lo siguiente:

| | Regresión Lineal | Regresión Ridge | Regresión Lasso |
|---|---|---|---|
| Intercepto | −91.4669 | −91.5383 | 49.8501 |
| $R^2$ | 0.88 | 0.88 | 0.88 |
| ECM | 94.02 | 93.33 | 93.74 |

Para esta segunda base de datos de la prueba saber 11, se deduce que el mejor modelo que se ajusta para estos datos es el modelo de regresión Ridge ya que es la que posee el menor error cuadrático medio. Sin embargo, se tiene un excelente coeficiente de determinación en las tres regresiones de 0.88. Por otro lado, debemos tener en cuenta que poseemos muchos datos y con el modelo Ridge mantiene todos los predictores en el modelo teniendo así una buena precisión.

Los códigos de estas implementación se pueden encontrar en un repositorio de Github. (https://github.com/Jerryjm/regresion).

# 9. Conclusiones

La regresión Ridge y Lasso son dos técnicas en el ámbito del aprendizaje automático para abordar el problema del sobreajuste en modelos de regresión lineal. Ambos métodos añaden un término de penalización a la función de pérdida de regresión lineal, lo que ayuda a evitar que los coeficientes de regresión se vuelvan demasiado grandes. La principal diferencia entre la regresión Ridge y la regresión Lasso es que la regresión Lasso puede obligar a algunos de los coeficientes de regresión a ser cero, mientras que la regresión ridge no. Esto significa que la regresión lasso puede utilizarse para seleccionar variables predictoras, mientras que la regresión Ridge no. Por otro lado, la regresión Ridge mantiene todos los predictores en el modelo, aunque atenúa su influencia. Estas dos técnicas ofrecen herramientas valiosas para gestionar la complejidad de modelos en el contexto de la predicción y selección de características en problemas de aprendizaje automático.

En la regresión Ridge se puede observar que tiene la ventaja que reduce el sobreajuste sin perder demasiada precisión, además es relativamente robusto a la multicolinealidad. Sin embargo, una desventaja que posee esta técnica es que no puede seleccionar variables predictoras. En cambio, una de las ventajas en la técnica Lasso es que se puede seleccionar variables predictoras y reducir el sobreajuste de forma más eficaz que la regresión Ridge. La desventaja de Lasso es que puede ser más sensible a la multicolinealidad y puede dar lugar a modelos menos precisos que la regresión Ridge. La elección del método de regresión lineal regularizado más adecuado depende de las características específicas del problema que se esté abordando. Si se

desea reducir el sobreajuste sin perder demasiada precisión, la regresión Ridge es una buena opción. Por otro lado, si se quiere reducir el sobreajuste y seleccionar variables predictoras, la regresión Lasso es una buena opción.

En la realización de las implementaciones del trabajo se hizo un análisis de los dos conjuntos de datos con las tres regresiones en estudio. Para el primer conjunto de datos se observo que las regresiones lineal y Lasso son las mejores técnicas que se ajustan para esos datos, ya que tienen el menor error cuadrático medio. De igual manera, en la segunda base de datos se observa que la mejor técnica que se ajusta es la de Ridge. Para la realización de estas implementaciones se tuvo el apoyo del lenguaje `Python` con sus respectivos paquetes y funciones ya estudiados anteriormente. En general, la regresión Ridge es una buena opción cuando se desea reducir el sobreajuste sin perder demasiada precisión. Mientras, la regresión Lasso es una buena opción cuando se desea reducir el sobreajuste y seleccionar variables predictoras.

Por último, al aplicar las tres regresiones en estudio en el primer conjunto de datos, se obtuvo un coeficiente de determinación muy bajo, lo cual significa que tienen poca relación las variables. Es decir, se consiguió en la regresión lineal un coeficiente de 0.047, para Ridge de 0.025 y en Lasso de 0.045. Se observa que en todos los modelos se obtuvo un resultado muy bajo, sin embargo, como el trabajo es de aprendizaje de máquina, el objetivo era percibir cual modelo se ajusta mejor a los datos, para esto se recurrió al ECM para la regresión lineal y Lasso, donde el resultado del error fue de 0.038, pero el mejor modelo que se ajusta es el de Ridge,

ya que posee 0.031 de ECM, siendo este el menor de todos.

Por otro lado, en el segundo conjunto de datos, se consiguió que al utilizar las tres regresiones nos da un excelente coeficiente de determinación de 0.88 en cada una de ellas, es decir, que el 88 % de las variables dependientes es predicha por las variables independientes. Además, se obtuvo en la regresión lineal un ECM de 94.02, en Ridge de 93.33 y en Lasso de 93.74. En los resultados obtenidos se evidencio que el mejor modelo que se ajusta para este segundo conjuntos de datos es el de Ridge, ya que posee el menor ECM.

De todo lo anterior, se observo que en los dos conjuntos de datos el mejor modelo fue Ridge. De lo cual se puede decir que el modelo tiene un menor sobreajuste que la regresión lineal y se tiene que es mas interpretable el modelo. Por ultimo, cabe recalcar que nuestro estudio se basa en buscar cual de los tres modelos se ajustaba mejor a los conjuntos de datos, ya que ambos conjuntos de datos se ajustaron al modelo Ridge.

# Referencias

[1] N. R. Draper y H. Smith, *Applied regression analysis*. John Wiley & Sons, 1998, vol. 326.

[2] A. C. Figueiredo, I. S. Gomes-Filho, R. B. Silva et al., "Maternal anemia and low birth weight: a systematic review and meta-analysis," *Nutrients*, vol. 10, n.º 5, pág. 601, 2018.

[3] A. Claessens y M. Engel, "How important is where you start? Early mathematics knowledge and later school success," *Teachers College Record*, vol. 115, n.º 6, págs. 1-29, 2013.

[4] T. Hastie, R. Tibshirani, J. H. Friedman y J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009, vol. 2.

[5] M. Balzarini, J. Di Rienzo, M. Tablada et al., "Estadística y biometría," *Ilustraciones del uso de Infostat en problemas de agronomía. Universidad Nacional de Córdoba*, 2012.

[6] D. J. Hand, B. J. Manly y M. J. Crowder, *Statistics: A concise introduction*. New York: Chapman amp; Hall, 2001.

[7] T. Hastie, R. Tibshirani y M. Wainwright, *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015.

[8] D. C. Montgomery, E. A. Peck y G. G. Vining, *Introduction to linear regression analysis*. John Wiley & Sons, 2021.

[9] E. L. Lima, "Análise no Rn," *Coleçao Matemática Universitária-IMPA, Rio de Janeiro*, 2002.

[10] A. E. Hoerl y R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, n.º 1, págs. 55-67, 1970.

[11] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 58, n.º 1, págs. 267-288, 1996.

[12] B. Efron, T. Hastie, I. Johnstone y R. Tibshirani, "Least angle regression," *The Annals of Statistics*, vol. 32, n.º 2, págs. 407-499, 2004.

[13] W. Albers, P. C. Boon y W. C. Kallenberg, "Testing equality of two normal means using a variance pre-test," *Statistics & probability letters*, vol. 38, n.º 3, págs. 221-227, 1998.

[14] S. S. Shapiro y M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, n.º 3-4, págs. 591-611, 1965.

[15] G. James, D. Witten, T. Hastie, R. Tibshirani et al., *An introduction to statistical learning*. Springer, 2013, vol. 112.

[16] R. R. Picard y R. D. Cook, "Cross-validation of regression models," *Journal of the American Statistical Association*, vol. 79, n.º 387, págs. 575-583, 1984.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, págs. 2825-2830, 2011.

[18] B. Zadrozny y C. Elkan, "Transforming classifier scores into accurate multiclass probability estimates," en *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, págs. 694-699.

[19] E. Bair y R. Tibshirani, "Semi-supervised methods to predict patient survival from gene expression data," *PLoS biology*, vol. 2, n.º 4, e108, 2004.

[20] S. Bakin et al., "Adaptive regression and model selection in data mining problems," 1999.

[21] L. Breiman y J. H. Friedman, "Predicting multivariate responses in multiple linear regression," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 59, n.º 1, págs. 3-54, 1997.

[22] L. Wilkinson, "Classification and regression trees," *Systat*, vol. 11, págs. 35-56, 2004.

[23] E. J. Candès et al., "Compressive sampling," en *Proceedings of the international congress of mathematicians*, Madrid, Spain, vol. 3, 2006, págs. 1433-1452.

[24] J. B. Copas, "Regression, prediction and shrinkage," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 45, n.º 3, págs. 311-335, 1983.

[25] T. Batard y M. Bertalmío, "Generalized gradient on vector bundle–application to image denoising," en *Scale Space and Variational Methods in Computer Vision: 4th International Conference, SSVM 2013, Schloss Seggau, Leibnitz, Austria, June 2-6, 2013. Proceedings 4*, Springer, 2013, págs. 12-23.

[26] S. C. Lean, H. Derricott, R. L. Jones y A. E. Heazell, "Advanced maternal age and adverse pregnancy outcomes: A systematic review and meta-analysis," *PloS one*, vol. 12, n.º 10, e0186287, 2017.

[27] J. Katon, G. Reiber, M. A. Williams, D. Yanez y E. Miller, "Antenatal haemoglobin A1c and risk of large-for-gestational-age infants in a multi-ethnic cohort of women with gestational diabetes," *Paediatric and perinatal epidemiology*, vol. 26, n.º 3, págs. 208-217, 2012.

[28] P. David, I. Grossman Stanley y D. C. Lay, "Algebra Lineal. Una introducción Moderna," *Espacios*, vol. 6, n.º 2, pág. 12, 2007.

[29] W. J. Fu, "Penalized regressions: the bridge versus the lasso," *Journal of computational and graphical statistics*, vol. 7, n.º 3, págs. 397-416, 1998.

[30] J. R. Perea Luque, "Técnicas de Regularización en el aprendizaje estadístico," 2019.

[31] A. N. Tikhonov, "Solution of incorrectly formulated problems and the regularization method.," *Sov Dok*, vol. 4, págs. 1035-1038, 1963.

[32] G. A. Seber y A. J. Lee, *Linear regression analysis*. John Wiley & Sons, 2012.

[33] J. Hamilton, "Time Series Analysis (Princeton University Press, Princeton, NJ) ISBN 0-691-04289-6," *International Journal of Forecasting*, 1994.

[34] M. Carrasco Carrasco, "Técnicas de regularización en regresión: Implementación y aplicaciones," 2016.

[35] K. Mardia, J. Kent y J. Bibby, "Multivariate analysis academic press inc," *London) Ltd*, vol. 15, pág. 518, 1979.

[36] R. Horst y H. Tuy, *Global optimization: Deterministic approaches*. Springer Science & Business Media, 2013.

[37] C. Hans, "Bayesian lasso regression," *Biometrika*, vol. 96, n.º 4, págs. 835-845, 2009.

[38] D. G. Kleinbaum, "Selecting the best regression equation.," *Applied regression analysis and other multivariate methods.*, págs. 314-340, 1988.

# Anexos

## Anexo 1. Fundamentos de álgebra lineal

### Matrices especiales

A continuación se define algunas propiedades de matrices especiales que serán de gran utilidad para realizar el proceso de estimación de los parámetros del modelo de regresión lineal general, para más detalles se puede revisar David, Grossman Stanley y Lay (2007).

**Definición 5.** *Sea* $\boldsymbol{X} = [x_{ij}]$ *una matriz de orden* $n \times p$*, entonces su traspuesta de* $\boldsymbol{X}^T$ *es una matriz de orden* $p \times n$ *que se obtiene convirtiendo cada elemento* $x_{ij}$ *en* $x_{ji}$*.*

Ejemplo

**Ejemplo 5.**

$$\boldsymbol{A} = \begin{bmatrix} 1 & 4 & 5 & 4 \\ 2 & 8 & 6 & 2 \\ 6 & -4 & 5 & 3 \end{bmatrix} \tag{26}$$

*Luego la matriz traspuesta queda de la siguiente manera*

$$\boldsymbol{A}^T = \begin{bmatrix} 1 & 2 & 6 \\ 4 & 8 & -4 \\ 5 & 6 & 5 \\ 4 & 2 & 3 \end{bmatrix} \tag{27}$$

Algunas propiedades de la traspuesta son:

1. $(\mathbf{X}^T)^T = \mathbf{X}$
2. $(\mathbf{X} + \mathbf{Y})^T = \mathbf{X}^T + \mathbf{Y}^T$
3. $(\mathbf{XY})^T = \mathbf{Y}^T\mathbf{X}^T$
4. $(K\mathbf{X})^T = K\mathbf{X}^T$

donde $\mathbf{X}$, $\mathbf{Y}$ son matrices y $K$ es un escalar.

**Definición 6.** *Una matriz $\boldsymbol{X}$ es simétrica si es igual a su traspuesta, $\boldsymbol{X} = \boldsymbol{X}^T$. En general, si $\boldsymbol{X}$ es simétrica entonces sus elementos son simétricos con respecto a la diagonal principal de $\boldsymbol{X}$.*

**Definición 7.** *Sea $\boldsymbol{X} \in M_n(\mathbb{R})$, entonces se dice que $\boldsymbol{X}$ es singular si y solo si $det(\boldsymbol{X}) = 0$, es decir, una matriz es singular si alguna de sus filas o columnas es linealmente dependiente.*

**Definición 8.** *Se dice que $\boldsymbol{X}$ es invertible si existe una matriz cuadrada $\boldsymbol{X}^{-1}$ de orden $n$, tal que $\boldsymbol{X}\boldsymbol{X}^{-1} = \boldsymbol{X}^{-1}\boldsymbol{X} = \boldsymbol{I}_n$, donde $\boldsymbol{I}_n$ es la matriz identidad y $\boldsymbol{X}^{-1}$ representa la inversa de $\boldsymbol{X}$.*

**Observación :** Si la matriz $\mathbf{X}$ es singular su determinante es igual a cero, lo que implica que $\mathbf{X}^{-1}$ no existirá, por lo tanto indica dependencia lineal.

Algunas propiedades de las matrices inversas:

1. $\mathbf{X}^{-1}$ existe y es única, si y solo si, $det(\mathbf{X}) \neq 0$.

2. $(\mathbf{X}^{-1})^{-1} = \mathbf{X}$

3. $(\mathbf{XY})^{-1} = \mathbf{Y}^{-1}\mathbf{X}^{-1}$

4. $(\mathbf{X}^{T})^{-1} = (\mathbf{X}^{-1})^{T}$

## Anexo 2. Derivada de matrices

A continuación se presenta dos casos particulares para derivar matrices, los cuales serán de gran utilidad para realizar el calculo matemático.

**Caso** 1 **:** Sea $\mathbf{y} = \mathbf{Ax}$ la forma matricial de un sistema de ecuaciones, donde $\mathbf{x}$, $\mathbf{y}$ son vectores de $n \times 1$ y $\mathbf{A}$ es una matriz de $n \times n$. Entonces, la derivada de $\mathbf{y}$ con respecto a $\mathbf{x}$ esta dada por:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \mathbf{A} \tag{28}$$

**Ejemplo 6.** *Sea* $\boldsymbol{y} = \boldsymbol{Ax}$ *el sistema de ecuaciones*

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 3 & 1 & 0 \\ 2 & -1 & 2 \\ 1 & 4 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

*entonces aplicamos el caso* 1*,*

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} = \begin{bmatrix} 3 & 1 & 0 \\ 2 & -1 & 2 \\ 1 & 4 & 3 \end{bmatrix}$$

**Caso 2 :** Sea $\mathbf{y} = \mathbf{x}^T\mathbf{A}\mathbf{x}$ un sistema de ecuaciones, donde $\mathbf{x}$, $\mathbf{y}$ son vectores de $n \times 1$ y $\mathbf{A}$ es una matriz simétrica de $n \times n$. Entonces, la derivada de $\mathbf{y}$ con respecto a $\mathbf{x}$ esta dada por:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = 2\mathbf{A}\mathbf{x}. \tag{29}$$

**Ejemplo 7.** *Se tiene el siguiente sistema de ecuaciones*

$$\boldsymbol{y} = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

*entonces, según la ecuación (29), se obtendrá*

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} = 2 \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} = \begin{bmatrix} 4 & -2 \\ -2 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} = \begin{bmatrix} 4x_1 - 2x_2 \\ -2x_1 + 6x_2 \end{bmatrix}$$

*De lo anterior se observa las derivadas parciales de* $\boldsymbol{y}$ *con respecto a cada componente de* $\boldsymbol{x} = [x_1, x_2]^T$*, es decir*

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}_1} = 4x_1 - 2x_2$$

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}_2} = -2x_1 + 6x_2$$

## Anexo 3. Descripción de los paquetes utilizados

### Función `RidgeCV`

Según Pedregosa, Varoquaux, Gramfort et al. [17] `RidgeCV` es una función del paquete `sklearn.linear_model` de `Python` el cual se utiliza para calcular el mejor valor del parámetro lambda para un modelo de regresión ridge. Para esto se utiliza por defecto la validacion cruzada dejando uno afuera (leave-one-out). El parámetro lambda controla la cantidad de penalización que se aplica.

La función `RidgeCV` funciona de la siguiente manera:

- Primero se divide el conjunto de datos en dos partes los de entrenamiento y el conjunto de prueba. Con el conjunto de entrenamiento se utiliza para entrenar el modelo de regresión ridge mientras con el conjunto de prueba se evalúa el modelo y se escoge el mejor valor para lambda.
- Se calcula el modelo de regresión ridge para un rango de valores de lambda en el conjunto de entrenamiento. Por lo general, se utiliza un rango de valores de lambda que se incrementa exponencialmente. Esto ayuda que el paquete explora un rango amplio de valores de lambda de manera eficiente.

- Luego se calcula la métrica de error con el conjunto de prueba para cada valor de lambda. Esta métrica de error que se maneja para evaluar el modelo depende de la aplicación la métrica que se utiliza es la de error cuadrático medio.
- El valor de lambda que tenga menor error en el conjunto de prueba se selecciona como el mejor valor de lambda. Con este valor de lambda se construye un modelo de regresión ridge que tenga un buen ajuste al conjunto de datos.

La función `RidgeCV` posee los siguientes parámetros:

- `alpha` : El valor de alpha debe ser un número entre 0 y 1. Cuando el valor de alpha es 0 corresponde a un modelo de regresión lineal, por otro lado cuando el valor de alpha es 1 corresponde a un modelo de regresión ridge completo.
- `fit_intercept` : Este parámetro controla si el modelo de regresión ridge debe incluir un término de intercepto. El término de intercepto es un valor que se suma a los valores predichos del modelo. El valor predeterminado de `fit_intercept` es `TRUE`, lo que significa que el modelo de regresión ridge incluirá un término de intercepto. Si es `FALSE` el modelo no incluirá un término de intercepto.
- `scoring` : Es el que controla la métrica que se utiliza para evaluar el modelo de regresión ridge durante la validación cruzada. El valor predeterminado de `scoring` es `deviance`, que es una medida de la diferencia entre los valores predichos y los valores observados.

- `cv` : Este parámetro es el que examina el método de validación cruzada que se utilizara para seleccionar el mejor valor de lambda. El valor predeterminado de `cv` es 5, lo que significa que se utilizara una validación cruzada de $K$-pliegues con $K = 5$.

  Existen otros valores para `cv` una de ellas es `LOOCV` que es la validación cruzada con un solo pliegue, otra opción es `nfolds` la cual consta de la validación cruzada $K$-pliegues con un valor especifico para $K$. Por último si obtenemos un `none` no se utilizara validación cruzada, luego el mejor valor de lambda se escoge con los datos de entrenamiento completo.

- `gcv_mode` : El parámetro `gcv_mode` controla el método que se utilizará para calcular la curva de información de la validación cruzada. El método predeterminado, `"gcv"`, utiliza la curva de información de la validación cruzada generalizada. Este método es el más preciso, pero también es el más lento. Una ventaja es que se selecciona el mejor lambda de manera más fiable al utilizar los datos de entrenamiento y así estimar el error de generalización del modelo.

- `store_cv_values` : Con este parámetro se guardan los valores de la validación cruzada. El valor predeterminado de `store_cv_values` es `TRUE`, lo que significa que se guardaran los valores de la validación cruzada. Si se establece `FALSE` los valores no se guardaran. Con este parámetro se permite realizar un análisis más detallado de los resultados de la validación cruzada.

- `alpha_per_target` : Vigila si se utilizará un valor de alpha diferente para

cada variable de respuesta. El valor predeterminado de `alpha_per_target` es `FALSE`, lo que significa que se utilizará el mismo valor de alpha para todas las variables de respuesta. Si se establece `alpha_per_target` en `TRUE`, se utilizará un valor de alpha diferente para cada variable de respuesta.

Ahora observemos los atributos que contiene la función `RidgeCV` :

- `cv_values_` : Este atributo contiene los valores de la validación cruzada para cada valor de lambda evaluado. Es un objeto de clase `matrix` cada fila de la matriz corresponde a un valor de lambda evaluado, y cada columna corresponde a una métrica de rendimiento. Las métricas que se utilizan son el error cuadrático medio, el error absoluto medio y la diferencia entre los valores predichos y los observados.

- `coef_` : Contiene los coeficientes del modelo de regresión ridge además es un objeto de clase `matrix`. Cada fila de la matriz corresponde a una variable predictora, y cada columna a un valor de lambda evaluado. Los valores de los coeficientes son los pesos asignados para cada variable predictora. El valor de lambda controla la cantidad de penalización. Un valor de lambda pequeño indica poca penalización, lo que puede provocar el sobreajuste. Un valor de lambda grande indica mucha penalización, lo que puede provocar una pérdida de precisión.

- `intercept_` : Este atributo contiene la intersección del modelo de regresión ridge, es un objeto de clase `numeric`. El valor del atributo `intercept_` es el valor que se suma a la predicción del modelo. Se calcula utilizando el método

de regularización ridge con este método penaliza los coeficientes del modelo para evitar el sobreajuste.

El valor de lambda controla la penalización. Dado un valor de lambda pequeño indica poca penalización, lo que puede provocar el sobreajuste. Un valor de lambda grande indica mucha penalización, lo que puede provocar una pérdida de precisión. Con este atributo se permite estimar el valor esperado de la variable de respuesta cuando todas las variables predictoras son cero.

- `alpha_` : Tiene el valor de alpha utilizando para con el hacer el modelo de regresión ridge, este atributo es de clase `numeric` el valor de `alpha_` controla la cantidad de penalización utilizada en el modelo. El valor de `alpha_` puede ser 0 ó 1 cuando es cero se utiliza ridge y cuando es uno se aplica Lasso.

- `best_score_` : Este atributo contiene el valor del mejor rendimiento en la validación cruzada, tenemos también que es un objeto de clase `numeric`. El valor de `best_score_` es el valor de la métrica de rendimiento utilizada en la validación cruzada para el valor óptimo de lambda. Este se calcula con el método de validación cruzada con eso se evalúa el modelo con los datos de validación para cada lambda.

- `n_features_in_` : Contiene el número de variables predictoras en el conjunto de datos de entrenamiento este atributo es un objeto de clase `integer`. El valor de `n_features_in_` es el número de columnas en el conjunto de datos de entrenamiento, excluyendo la columna de la variable respuesta.

El paquete `RidgeCV` posee varios métodos uno de ellos es el de `fit()` el cual se utiliza para entrenar el modelo de regresión ridge utilizando la validación cruzada. Este método tiene los siguientes argumentos:

- `X` : Es un objeto de clase `matrix` que contiene los datos de entrenamiento.
- `y` : Vector que contiene la variable respuesta de los datos de entrenamiento.
- `sample_weight` : Es un vector que contiene los pesos de los datos de entrenamiento.

El funcionamiento del método `fit()` inicia cuando dividimos el conjunto de datos en entrenamiento y validación. Luego se comprueba si `sample_weight` esta especificado si no lo esta, se asume que todas las muestras tienen el mismo peso. Si `sample_weight` esta especificado, se utiliza las muestras de entrenamiento.

Se realiza una búsqueda exhaustiva de los hiperparámetros alpha el cual esta en un intervalo especificado. Esto se hace para hallar el valor de alpha el cual genera el mejor rendimiento en el conjunto de datos de entrenamiento. Para cada valor de alpha, se entrena el modelo esto se hace utilizando el algoritmo de descenso de gradiente estocástico. Una vez que se han entrenado el modelo se evalúa el rendimiento de cada uno de ellos en un conjunto de validación, se selecciona el valor de alpha que genera el mejor rendimiento en el conjunto de validación.

En general el método `fit()` permite encontrar el valor de alpha que genera el mejor rendimiento en el conjunto de datos de entrenamiento, lo que puede mejorar

el rendimiento del modelo en el conjunto de datos de prueba.

Otro método de este paquete es `predict(X)` el cual es utilizado para predecir los valores de una variable dependiente a partir de un conjunto de datos de prueba. El método primero comprueba si el modelo ha sido entrenado, si no lo esta, se produce una excepción. Luego utiliza el modelo entrenado para predecir las etiquetas de cada muestra del conjunto de datos de prueba para predecir la etiqueta de una muestra, el método que se utiliza es el siguiente:

$$\texttt{y\_pred} = b + X \cdot w. \tag{30}$$

Donde `y_pred` es la etiqueta predicha, $b$ es el término independiente del modelo, $w$ es el vector de coeficientes del modelo y $X$ es la matriz de características de la muestra. Este método es muy útil para evaluar el rendimiento del modelo en el conjunto de datos de prueba.

El método `score()` en el paquete `RidgeCV` evalúa el rendimiento del modelo con los datos de prueba los argumentos de este método son $X$ el cual es la matriz de características de prueba, $y$ es el vector de etiquetas de prueba y `sample_weight` vector de pesos de las muestras de prueba.

El método `score()` devuelve una puntuación que mide el rendimiento del modelo en el conjunto de prueba, se calcula utilizando una métrica de evaluación de regresión, como el error cuadrático medio ó el coeficiente de determinación. La

puntuación se calcula mediante la siguiente formula:

$$\texttt{score = metric(y\_test, y\_pred)}, \tag{31}$$

donde `score` es la puntuación, `metric` es la métrica de evaluación, `y_test` es el vector de etiquetas de prueba por último `y_pred` es el vector de etiquetas predichas.

El método `set_fit_request()` establece los datos de entrenamiento y de validación que se utilizaran para entrenar el modelo. Los datos de entrenamiento se utilizaran para estimar los coeficientes del modelo, mientras que los datos de validación se utilizaran para seleccionar el mejor valor del alpha. Si se desea entrenar el modelo con datos específicos de entrenamiento y validación, esto puede resultar útil.

El método `set_params(params)` en el paquete `RidgeCV` establece los parámetros del modelo. El método comprueba si el modelo ha sido entrenado, si lo esta se produce una excepción. Los parámetros del modelo se obtiene de los atributos del modelo esos se pueden enumerar utilizando el método `dir()`.

El parámetro `alpha` es el hiperparámetro más importante del modelo `RidgeCV`. El valor de alpha controla la cantidad de regularización que se aplica al modelo. Una mayor alpha implica una mayor regularización, lo que reduce la complejidad del modelo y evita el sobreajuste.

El parámetro `normalize` controla si se deben normalizar las características antes de entrenar el modelo. La normalización puede ayudar a mejorar el rendimiento del modelo.

El parámetro `scoring` controla la métrica de evaluación que se utilizará para seleccionar el mejor valor de alpha. El valor predeterminado es `neg_mean_squared_error`, que mide el error cuadrático medio.

El método `set_score_request()` establece la métrica de evaluación que se utilizará para seleccionar el mejor valor del hiperparámetro alpha. El método establece la métrica de evaluación especificada esa métrica de evaluación se utiliza para evaluar el rendimiento del modelo en el conjunto de datos de validación.

Las métricas de evaluación específicas del modelo `RidgeCV` son las siguientes:

- `neg_mean_squared_error` : Error cuadrático medio negativo.
- `mean_squared_error` : Error cuadrático medio.
- `mean_absolute_error` : Error absoluto medio.
- `median_absolute_error` : Error absoluto mediano.
- `r2` : Coeficiente de determinación.

Estas métricas se pueden establecer utilizando el método `set_score_request()`.

**Función** `LassoCV`

Según Pedregosa, Varoquaux, Gramfort et al. en [17] `LassoCV` es una función del paquete `sklearn.linear_model` el cual implementa el método de validación cruzada para escoger el valor óptimo de la penalización. Este modelo se entrena en cada partición y se evalúa en la partición restante, el valor de la penalización que produce el mejor rendimiento en la evaluación se selecciona como el valor óptimo.

La función `LassoCV` funciona de la siguiente manera:

- Se crea un objeto del modelo `LassoCV`. Se puede especificar el valor de alpha, pero si no se especifica el modelo se escogerá automáticamente un valor óptimo.
- Se entrena el modelo en los datos de entrenamiento, luego el modelo utiliza el método de validación cruzada para seleccionar el valor óptimo de alpha.
- Se obtiene los coeficientes del modelo. Estos coeficientes indican la importancia de cada característica para el modelo.

Para `LassoCV` la validacion cruzada funciona de la siguiente forma:

- Los datos de entrenamiento se dividen en k particiones.
- El modelo se entrena en $k-1$ particiones y se evalúa en la partición restante.
- El proceso se repite $k$ veces, una vez para cada partición.
- El valor de alpha que produce el mejor rendimiento en la evaluación se selecciona como el valor óptimo.

En el paquete `LassoCV` el objetivo de optimizar es $(1/(2 \cdot \texttt{n\_samples})) \cdot ||y - Xw||_2^2 + \texttt{alpha} \cdot ||w||_1$.

La primera parte de la función objetivo, $||y - Xw||_2^2$, es la función de pérdida de regresión lineal regularizada, además `n_samples` es el numero de muestras. Esta función de pérdida mide la distancia entre los valores predichos y los valores reales. La segunda parte de la función objetivo, $\texttt{alpha} \cdot ||w||_1$, es la penalización L1. Esta penalización ayuda a reducir el sobreajuste al forzar a algunos de los coeficientes del modelo a ser cero.

El valor de alpha controla la cantidad de regularización que se aplica al modelo. Un valor de alpha pequeño significa que se aplica poca regularización, lo que puede conducir al sobreajuste y cuando el valor es grande se aplica mucha regularización.

La expresión $(1/(2 \cdot \texttt{n\_samples})) \cdot ||y - Xw||_2^2$ es la función de pérdida de regresión lineal regularizada. Esta función de pérdida mide la distancia entre los valores predichos y los valores reales. La distancia se mide utilizando la norma euclidiana. Por lo tanto, la optimización por Lasso busca encontrar el vector de coeficientes `w` que minimice la función de pérdida de regresión lineal regularizada y la penalización L1.

Los parámetros que posee la función `LassoCV` son:

- `eps` : Este parámetro del paquete `LassoCV` controla la precisión de la solución del problema de optimización, el valor predeterminado de `eps` es `1e-8`. Cuando el parámetro es pequeño la solución del problema de optimización es más precisa. Aunque, una solución más precisa puede tardar mucho tiempo en calcularse. Por otro lado, si el parámetro es grande la solución del problema de optimización sera menos precisa pero se calculara más rápido.

  Un valor de `eps` de `1e-8` suele ser un buen punto de partida. Sin embargo, es posible que necesite ajustar el valor de `eps` para obtener mejores resultados en su aplicación específica.

- `n_alphas` : Maneja el número de valores de alpha el cual se prueba en el proceso de validación cruzada, el valor usual de `n_alphas` es 10. Cuando el valor de `n_alphas` es alto significa que se probara más valores de alpha lo cual ayuda a encontrar el valor óptimo para el conjuntos de datos. Se recomienda utilizar un valor alto para tener una buena probabilidad de encontrar un valor óptimo de alpha, pero no tan alto para que el proceso no se tarde.

- `alphas` : El parámetro `alphas` controla los valores de alpha que se probara durante el proceso de validación cruzada, el valor predeterminado de `alphas` es `None` lo que nos dice que el modelo seleccionara automáticamente una serie de valores de alpha. Un valor pequeño de alpha significa que hay poca regularización, lo que conduce al sobreajuste en cambio un valor alto equivale a tener mucha regularización.

- `fit_intercept` : Este parámetro controla si el modelo debe estimar un coeficiente de intersección, el valor determinado de `fit_intercept` es `TRUE`, es decir, que el modelo estimara un coeficiente de intersección, ahora si es `FALSE` el modelo no estimara un coeficiente de intersección lo que nos dice que el modelo no tendrá un termino constante en la ecuación de regresión.

- `precompute` : Supervisa si el modelo debe calcular la matriz de Gram antes de entrenar el modelo su valor predeterminado de `precompute` es `auto`, lo que indica que el modelo utilizara la precompilación si la matriz de Gram es simétrica y no es demasiado grande. Si el valor de este parámetro es `TRUE` el modelo calculara la matriz de Gram antes de entrenar la cual se calculara con la ecuación $K = X^T X$ el modelo lo cual puede mejorar el modelo al reducir el tiempo para calcular los coeficientes del modelo.

  Ahora si el valor es `FALSE` el modelo no calculara la matriz Gram antes de entrenar el modelo lo cual se calcula de la siguiente manera `K = np.dot(X, X.T)`, esto puede reducir la cantidad de memoria utilizada por el modelo, así reducir el rendimiento del modelo.

- `max_iter` : Observa el número máximo de iteraciones que se realizaran durante el proceso de entrenamiento del modelo, el valor predeterminado del `max_iter` es 1000. Si se establece un valor menor el modelo no converge a una solución óptima y si escogemos un valor mayor el modelo se puede tardar mas tiempo en entrenarse.

La técnica `LassoCV` emplea un procedimiento de descenso gradiente para calibrar los coeficientes del modelo, el cual itera sobre ellos y realiza ajustes en cada paso con la finalidad de reducir la tasa de error durante su entrenamiento. El parámetro `max_iter` controla el número de veces que el algoritmo de descenso de gradiente itera sobre los coeficientes del modelo, si el algoritmo no ha convergido a la solución óptima después de `max_iter` iteraciones, el algoritmo finalizara.

- `tol` : Comprueba la tolerancia utilizada para detener el proceso de entrenamiento del modelo. El valor usual de `tol` es `1e-4`, si el error de entrenamiento del modelo disminuye por debajo de `tol` el proceso de entrenamiento termina. Si el valor de `tol` es menor el modelo sera más preciso el problema es que se tardara más tiempo en entrenarse y si el valor es mayor sera menos preciso el modelo pero tarda menos tiempo en entrenarse.

  El parámetro `tol` controla la cantidad de cambio en el error de entrenamiento que se considera un progreso significativo. Si el error de entrenamiento disminuye por debajo de `tol`, el algoritmo de descenso de gradiente finaliza, ya que se considera que el modelo ha convergido a la solución óptima.

- `copy_X` : Controla si los datos de entrenamiento se copiaran antes de que se usen para entrenar el modelo, el valor usual de `copy_X` es `TRUE` lo que significa que los datos de entrenamiento se copiaran si el valor es `FALSE` no se copiaran.

Lo cual puede mejorar el rendimiento del modelo pero causa problemas si los datos de entrenamiento se modifican después de que el modelo se haya entrenado.

- `cv` : Observa el método de validación cruzada que se utiliza para seleccionar el alpha más óptimo, el valor predeterminado de `cv` es 5 lo que significa que se utilizaran la validación cruzada con 5 particiones. Si se establece un valor mayor de `cv` el modelo sera más preciso pero se tardara más al entrenarse, ahora si el valor es menor el modelo sera menos preciso pero tarda menos tiempo en entrenarse.

  El parámetro `cv` funciona de la siguiente manera:

  - Los datos de entrenamiento se dividen en `cv` particiones.
  - Para cada valor de alpha especificado en `alphas`, el modelo se entrena en `cv-1` particiones y se evalúa en la partición restante.
  - El proceso se repite `cv` veces, una vez para cada partición.
  - El valor de alpha que produce el mejor rendimiento en la evaluación se selecciona como el valor óptimo.

- `verbose` : El parametro controla la cantidad de información que se imprimirá durante el entrenamiento del modelo. Establezca el valor de `verbose` a un valor lo suficientemente alto para proporcionar información útil sobre el progreso del entrenamiento, pero no tan alto que la salida sea demasiado larga o innecesaria.

- `n_jobs` : El parámetro `n_jobs` del paquete `LassoCV` observa el número de núcleos de la `CPU` que se utilizarán para entrenar el modelo. Establezca el valor de `n_jobs` a un valor igual al número de núcleos de la `CPU` disponibles para aprovechar al máximo el hardware disponible y acelerar el proceso de entrenamiento sin utilizar demasiada memoria.

- `positive` : Determina si todos los coeficientes del modelo deben ser positivos, el valor usual de `positive` es `FALSE` lo que significa que los coeficientes pueden ser negativos por otro lado si el valor es `TRUE` todos los coeficientes del modelo serán positivos. Esto es útil en los casos que todos los coeficientes deben ser positivos, como en el caso de modelos de regresión logística.

- `random_state` : Este parámetro controla la semilla del generador de números aleatorios que se utiliza para seleccionar las características a actualizar durante el entrenamiento del modelo.

  Si el valor de `random_state` es `None`, la semilla del generador de números aleatorios se establecerá aleatoriamente. Esto significa que las características seleccionadas para actualizar serán diferentes cada vez que se entrene el modelo.

  Si el valor de `random_state` es un número entero, la semilla del generador de números aleatorios se establecerá en ese número. Esto significa que las características seleccionadas para actualizar serán las mismas cada vez que se entrene el modelo con el mismo valor de `random_state`.

- `selection` : Controla el método de selección de características que se utilizará para seleccionar las características a incluir en el modelo. El valor que se tiene

predeterminado de `selection` es `cyclic`, lo que significa que las características se seleccionarán de forma cíclica.

Con la función `LassoCV` podemos observar los siguientes atributos:

- `alpha_` : Es un vector que contiene los valores de alpha utilizados para entrenar el modelo. El valor de alpha controla la penalización de la norma L1 de los coeficientes del modelo. Si el valor de alpha es mayor, la norma L1 penalizará más fuertemente y provocará que se vuelvan cero una cantidad mayor de coeficientes del modelo. En cambio, si el valor de alpha es menor, la penalización será menos intensa y permitirá un número superior a cero en los coeficientes del modelo. Normalmente el valor de alpha se toma de 0.0001.

- `coef_` : Es un vector que contiene los coeficientes del modelo entrenado que representan la importancia relativa de cada característica en la predicción del objetivo. Los valores de los coeficientes pueden ser positivos ó negativos. Un valor positivo indica que la característica está relacionada positivamente con el objetivo, mientras que un valor negativo indica que la característica está relacionada negativamente con el objetivo.

  El atributo `coef_` tiene el siguiente funcionamiento primero el modelo `LassoCV` entrena el modelo utilizando el algoritmo de descenso de gradiente con penalización de la norma L1, este algoritmo ajusta los coeficientes del modelo para minimizar el error de entrenamiento.

- `intercept_` : Es un escalar el cual representa el termino de intercepto del modelo entrenado. El termino de intercepto es el valor que se suma a la suma

de los productos de los coeficientes y las características para tener la predicción del modelo. El intercepto cuando es positivo significa que el modelo predice un valor mayor que el promedio para los datos de entrenamiento, cuando es negativo indica que el modelo predice un valor menor que el promedio de los datos de entrenamiento.

- `mse_path_` : Es un vector que contiene los errores cuadráticos medios del modelo entrenado para cada valor de alpha utilizado en la validación cruzada. un valor menor en el error cuadrático medio significa que el modelo es más preciso. Este atributo se puede utilizar para visualizar el rendimiento del modelo para diferentes valores de alpha.

  El atributo `mse_path_` funciona de esta manera primero con `LassoCV` entrena un modelo para cada valor de alpha en el vector `alpha_`. El error cuadrático medio del modelo se calcula para cada conjunto de datos de validación, luego el atributo se almacena como un vector, donde cada elemento representa el error cuadrático medio del modelo para un valor especifico de alpha.

- `alphas_` : Es un vector que contiene los valores de alpha utilizados para entrenar el modelo. El valor de alpha es el que controla la penalización de los coeficientes del modelo. Con un valor alto de alpha significa una penalización más fuerte de la norma L1, es decir, más coeficientes del modelo serán cero en cambio cuando el valor de alpha es pequeño la penalización es débil.

  El modelo `LassoCV` entrena un modelo para cada valor de alpha en el vector `alphas_`. El modelo con el mejor rendimiento se selecciona como el mejor

modelo. El atributo `alphas_` se almacena como un vector, donde cada elemento del vector representa el valor de alpha utilizado para entrenar un modelo específico.

- `dual_gap_` : El atributo `dual_gap_` se puede utilizar para visualizar el rendimiento del modelo para diferentes valores de alpha.

  Es un vector que contiene la brecha dual para cada valor de alpha utilizado en la validación cruzada. La brecha dual es una medida de la diferencia entre la solución primal y la solución dual del problema de regularización. Una brecha dual cero indica que la solución primal y la solución dual son iguales, lo que significa que el modelo ha alcanzado el óptimo.

- `n_iter_` : Es un entero que representa el número de iteraciones realizadas por el algoritmo de descenso de gradiente para entrenar el modelo. Con este algoritmo se utiliza para ajustar los coeficientes del modelo `LassoCV`, el número de iteraciones determina la precisión con la que el algoritmo converge a la solución óptima.

- `n_features_in_` : Es un entero que representa el número de características en los datos de entrenamiento, es útil para determinar el tamaño de la matriz de coeficientes del modelo. La matriz de coeficientes tiene una dimensión de `n_features_in_` por 1.

  Si el valor de `n_features_in_` es diferente del número de características en los datos de entrenamiento, puede indicar un problema con los datos de entrenamiento o con el modelo.

- `feature_names_in_` : Este atributo se utiliza para entender cómo el modelo está adecuando las características. Si el modelo no está utilizando todas las características, puede indicar que el modelo no está bien especificado o que los datos de entrenamiento no son adecuados para el modelo.

La función `LassoCV` tiene el método `fit()` el cual se utiliza para entrenar el modelo, el método toma dos argumentos obligatorios los cuales son:

- `X` : Los datos de entrenamiento.
- `y` : Los valores objetivo.

El método `fit()` empieza preparando los datos de entrenamiento. Esto incluye escalar los datos, si es necesario. El método `LassoCV` utiliza la función `StandardScaler()` para escalar los datos. Después que los datos ya estén preparados este método utiliza un algoritmo de validación cruzada para seleccionar el valor de alpha que produce el mejor rendimiento del modelo, utiliza el algoritmo de validación cruzada `K-fold`. El algoritmo divide los datos de entrenamiento en `k` conjuntos de datos este modelo se entrena `k` veces, utilizando un conjunto de datos diferente como conjunto de validación cada vez. El rendimiento del modelo se mide en el conjunto de validación.

El valor de alpha que produce el mejor rendimiento del modelo en la validación cruzada se selecciona como el valor de alpha para el modelo entrenado. Una vez que se ha seleccionado el valor de alpha, el método `fit()` entrena el modelo con el valor de alpha seleccionado. Se utiliza el algoritmo de descenso de gradiente para entrenar

el modelo.

Con este algoritmo se ajusta los coeficientes del modelo para así minimizar el error de entrenamiento el cual se mide como la suma de los cuadrados de los residuos. El método `fit()` devuelve el modelo entrenado. El modelo entrenado se puede utilizar para predecir valores objetivo para nuevos datos.

El método `fit()` es el método principal para entrenar un modelo `LassoCV`. Sin embargo, el paquete `LassoCV` también proporciona otros métodos para trabajar con modelos:

- `predict()` : Se utiliza para predecir valores objetivo para nuevos datos.
- `predict_proba()` : Se utiliza para predecir probabilidades para nuevos datos.
- `coef_` : Se utiliza para obtener los coeficientes del modelo.
- `intercept_` : Se utiliza para obtener la constante del modelo.

El método `get_metadata_routing()` se utiliza para obtener la ruta de los metadatos del modelo. El método devuelve un diccionario que contiene lo siguiente:

- `alpha` : El valor de alpha utilizado para entrenar el modelo.
- `cv` : El tipo de validación cruzada utilizado para seleccionar el valor de alpha.
- `n_folds` : El número de conjuntos de datos utilizados para la validación cruzada.

El método recupera los atributos `alpha_`, `cv_` y `n_folds_` del modelo, luego crea un diccionario que contiene los valores de estos atributos. Por ejemplo, si el valor de alpha es 0.1, y el número de iteraciones es 100, el parámetro `normalize` es `TRUE` y el parámetro `fit_intercept` es `FALSE`, la cadena sería la siguiente:

```
LassoCV:0.1:100:True:False
```

Una vez que se ha combinado la cadena, el método `get_metadata_routing()` codifica la cadena usando el algoritmo de codificación base64. La codificación base64 convierte los datos binarios en una cadena de caracteres que se puede transferir fácilmente a través de redes.

Otro método que podemos observar es `get_params()` el cual comienza recorriendo los parámetros del modelo. Los parámetros del modelo se definen en la clase `LassoCV`. Para cada parámetro este método obtiene su valor, el cual se obtiene utilizando el atributo correspondiente del modelo.

Una vez que se han obtenido los valores de todos los parámetros, el método `get_params()` almacena el nombre y el valor de cada parámetro en un diccionario. El valor del parámetro alpha controla la intensidad de la regularización. Un valor más alto de alpha significa que el modelo tendrá menos coeficientes no nulos.

El método `path()` se utiliza para calcular el camino de coeficientes creando un objeto de tipo Lasso con los parámetros especificados. Los parámetros especificados se pueden pasar al método `path()` como argumentos, este metodo devuelve un objeto

de tipo `LassoPath` que contiene los coeficientes del modelo para cada valor de alpha.

Una vez que se ha creado el objeto de tipo Lasso, el método `path()` itera sobre los valores de alpha especificados. El método `path()` utiliza el algoritmo de descenso de gradiente para entrenar el modelo para cada valor de alpha.

Los coeficientes del modelo se almacenan en el objeto de tipo `LassoPath`. El objeto de tipo `LassoPath` tiene los siguientes atributos:

- `alphas` : Los valores de alpha utilizados para entrenar el modelo.
- `coefs_` : Los coeficientes del modelo para cada valor de alpha.
- `intercepts_` : Las constantes del modelo para cada valor de alpha.

Ahora observemos los argumentos que toma el método `path()` los cuales son:

- `X` : Los datos de entrenamiento.
- `y` : Los valores objetivo.
- `eps` : El umbral de convergencia para el algoritmo de descenso de gradiente.
- `n_alphas` : El número de valores de alpha que se van a utilizar.
- `alphas` : Un vector de valores de alpha que se van a utilizar.

Donde el argumento de `n_alphas` controla el número de valores de alpha que se van a usar, cuando tenemos un valor alto significa que el método `path()`

devolverá mas información sobre el camino de coeficientes. El argumento `eps` controla la precisión con la que el algoritmo de descenso de gradiente converge a una solución óptima. Un valor más pequeño de `eps` significa que el algoritmo de descenso de gradiente convergerá con mayor precisión, pero también tomará más tiempo.

El método `predict()` se utiliza para predecir los valores objetivo para nuevos datos. Este método devuelve un vector de valores objetivo predichos. Donde comienza calculando la predicción para cada dato nuevo, la predicción se calcula utilizando la siguiente fórmula:

```
y_pred = X * coef_ + intercept_
```
(32)

donde `y_pred` es el valor objetivo predicho, `X` es el dato nuevo, `coef_` es el vector de coeficientes del modelo y `intercept_` es la constante del modelo. Ya cuando se hacen las predicciones para todos los datos nuevos, el metodo `predict()` devuelve un vector que contiene las predicciones.

Otro método del paquete `LassoCV` es `score()` el cual comienza calculando la métrica de puntuación. La métrica de puntuación que se utiliza se especifica mediante el argumento `scoring`. El argumento `scoring` puede tomar un valor de cualquier métrica de puntuación admitida por `sklearn.metrics`. Para utilizar el error cuadrático medio, puede pasar el argumento `scoring = 'neg_mean_squared_error'` al método `score()`. Una vez que se ha calculado la métrica de puntuación, el método `score()` devuelve el valor de la métrica de puntuación.

El método `set_fit_request()` se utiliza para especificar los datos de entrenamiento que se utilizarán para entrenar el modelo. Este comienza almacenando los datos de entrenamiento y los pesos de los datos de entrenamiento. Los datos de entrenamiento se almacenan en el atributo `X_` del modelo. Los pesos de los datos de entrenamiento se almacenan en el atributo `sample_weight_` del modelo. Cuando se tiene almacenados los datos de entrenamiento y los pesos de los datos de entrenamiento, el método `set_fit_request()` actualiza el estado del modelo para indicar que se ha especificado una solicitud de entrenamiento. Esto permite al modelo saber que está listo para ser entrenado.

El método `set_params()` se utiliza para establecer los parámetros del modelo. El método toma un diccionario como argumento. El diccionario contiene los nombres y los valores de los parámetros que se van a establecer. El método `set_params()` comienza recorriendo el diccionario de parámetros. Para cada parámetro, el método obtiene el nombre del parámetro y el valor del parámetro. Una vez que se han obtenido el nombre y el valor del parámetro. El método establece el valor del parámetro utilizando el atributo correspondiente del modelo.

El método `set_score_request()` sirve para especificar los datos de puntuación que se utilizaran para evaluar el modelo. El método también se utiliza para especificar los pesos de los datos de puntuación.

El método `set_score_request()` comienza almacenando los datos de puntuación y los pesos de los datos de puntuación. Los datos de puntuación se almacenan en el atributo `X_test_` del modelo. Los pesos de los datos de puntuación se almacenan en el atributo `sample_weight_test_` del modelo. Una vez que se han almacenado los datos de puntuación y los pesos de los datos de puntuación, el método `set_score_request()` actualiza el estado del modelo para indicar que se ha especificado una solicitud de puntuación. Esto permite al modelo saber que está listo para ser evaluado.

## Anexo 4. Implementación de las regresiones en `Python`

### Algoritmo de la regresión lineal

1. Paso 1: Preparación de datos

   Se dispone conjuntos de datos con pares $(x, y)$ donde $x$ es la variable independiente e $y$ es la variable dependiente.

2. Paso 2: Calcular medias

   Calcular las medias de "$x$" (`media_x`), "$y$" (`media_y`)

   `media_x = sum(x)/len(x)`, `media_y = sum(y)/len(y)`

3. Paso 3: Calcular la pendiente (m)

   Para calcular la pendiente se utiliza la formula:

   ```
   num = sum((xi-med_x)*(yi-medi_y) for xi,yi in zip(x,y))
   denom = sum((xi-media_x)**2 for xi in x)
   pendiente = num/denom
   ```

4. Paso 4: Calcular intercepto (b)

   Para calcular el intercepto (b) utilizando la siguiente formula:

   ```
   intercepto = media_y - pendiente * media_x
   ```

5. Paso 5: Crear la ecuación de regresión lineal.

6. Paso 6: Hacer perdiciones

   Utiliza la ecuación de regresión para hacer predicciones sobre nuevos datos

   ```
   def predecir_nuevo_valor(x_nuevo):
   return pendiente* x_nuevo + intercepto
   ```

7. Paso 7: Evaluar el modelo

   Se utiliza la métrica del error cuadrático medio (ECM) para evaluar el rendimiento del modelo

   ```
   def calcular_error_cuadrático_medio(y_ral, y_pred):
   n = len(y_real)
   mse = sum((yi-pi)**2 for yi,pi in zip(y_real, y_pred))/n
   return mse
   ```

**Algoritmo de la regresión Ridge**

1. Paso 1: Definir la función de regresión ridge

   ```
   def regresión_ridge(X, y, alpha, num_iteraciones, tasa_aprendizaje):
   ```

   La función toma la matriz de característica (X), el vector de valores de salida (y), el parámetro de regularización alpha, el numero de iteraciones para el descenso de gradiente `num_iteraciones` y la tasa de aprendizaje `tasa_aprendizaje`

2. Paso 2: Añadir columna de unos a la matriz de características

   `X_con_intercepto = np.c_[np.ones(X.shape[0]),X]`

   Se añade una columna de unos a la matriz de características para representar el termino independiente en la regresión.

3. Paso 3: Inicializar coeficientes

   `coeficientes = np.random.rand(X_con_intercepto.shape[1])`

   Se inicia los coeficientes aleatoriamente. Esta inicialización puede ajustarse según las necesidades.

4. Paso 4: Descenso de gradiente

   `for iteración in range(num_iteraciones):`

   Se utiliza el descenso de gradiente para actualizar los coeficientes y minimizar la función de costo regularizada.

5. Paso 5: Calcular predicciones y error

   ```
   y_pred = X_con_intercepto @ coeficientes
   error = y_pred - y
   ```

6. Paso 6: Calcular el gradiente

   `gradiente = 2 * X_con_intercepto.T @ error + 2 * alpha * coeficientes`

7. Paso 7: Actualizar coeficientes

   `coeficientes -= tasa_aprendizaje * gradiente`

8. Paso 8: Crear función de predicción

   `def predecir(X_nuevo): X_nuevo_con_intercepto =`

`np.c_[np.ones(X_nuevo.shape[0]), X_nuevo]`

`return X_nuevo_con_intercepto @ coeficientes`

Se crea una función que toma nuevas observaciones y devuelve predicciones utilizando los coeficientes calculados.

9. Paso 9: Devolver la función de predicción

   `return predecir`

**Algoritmo de la regresión Lasso**

1. Paso 1: Definir la función de regresión ridge

   `def regresión_ridge(X, y, alpha, num_iteraciones, tasa_aprendizaje):`

   La función toma la matriz de característica (X), el vector de valores de salida (y), el parámetro de regularización alpha, el numero de iteraciones para el descenso de gradiente `num_iteraciones` y la tasa de aprendizaje `tasa_aprendizaje`

2. Paso 2: Añadir columna de unos a la matriz de características

   `X_con_intercepto = np.c_[np.ones(X.shape[0]),X]`

   Añadimos una columna de unos a la matriz de características para representar el termino independiente en la regresión.

3. Paso 3: Inicializar coeficientes

   `coeficientes = np.random.rand(X_con_intercepto.shape[1])`

   Se inicia los coeficientes aleatoriamente. Esta inicialización puede ajustarse según las necesidades.

4. Paso 4: Descenso de gradiente

   `for iteración in range(num_iteraciones):`

   Se utiliza el descenso de gradiente para actualizar los coeficientes y minimizar la función de costo regularizada.

5. Paso 5: Calcular predicciones y error

   `y_pred = X_con_intercepto @ coeficientes`

   `error = y_pred - y`

6. Paso 6: Calcular el gradiente

   `gradiente = 2 * X_con_intercepto.T @ error`

   `+ alpha * np.sign(coeficientes)`

   Luego se calcula el gradiente de la función de costo regularizada utilizando la norma L1.

7. Paso 7: Actualizar coeficientes

   `coeficientes -= tasa_aprendizaje * gradiente`

8. Paso 8: Crear función de predicción

   `def predecir(X_nuevo): X_nuevo_con_intercepto =`

   `np.c_[np.ones(X_nuevo.shape[0]), X_nuevo]`

   `return X_nuevo_con_intercepto @ coeficientes`

   Se crea una función que toma nuevas observaciones y devuelve predicciones utilizando los coeficientes calculados.

9. Paso 9: Devolver la función de predicción

   `return predecir`