# Modelo de predicción de masa de pingüinos

**Autora:** Andrea Medina Rico **Matrícula:** A01705541

**Abstracto.** Este reporte detalla la creación e implementación de un modelo para predecir la masa en kilogramos de un pingüino, de acuerdo a algunas de sus características. Para esto, el modelo fue entrenado y evaluado con una regresión lineal múltiple, de la mano con un algoritmo para su optimización. Este modelo reporta un coeficiente de determinación de 0.814, indicando la capacidad de explicar el 81.4% en la variación de la masa de los pingüinos.

## I. INTRODUCCIÓN

En el ámbito de la zoología, un modelo de predicción de la masa de un animal a partir de algunas de sus medidas corporales puede resultar una gran herramienta. Esto reduciría la necesidad de procesos invasivos y costosos para mantener registro de sus medidas y su salud.

El objetivo de este modelo es generar una predicción de la masa de un pingüino a partir de algunas de sus medidas y sus características. El conjunto de datos 'penguins_size.csv', presenta información de pingüinos de tres distintas especies dentro de la misma familia. Se busca demostrar que se puede realizar la predicción de una característica corporal (en este caso, la masa), sin importar que un animal pueda tener variaciones dependiendo de la raza o especie que sea.

## II. EXTRACCIÓN, TRANSFORMACIÓN Y LIMPIEZA DE DATOS (ETL)

a. Descripción del conjunto de datos

La información de los pingüinos proviene del archivo 'penguins_size.csv'. Contiene información sobre pingüinos de tres islas distintas: Dream, Biscoe y Torgersen, las tres situadas en la Antártida. Además, distingue entre las tres especies de la familia Pygoscelis, conocida como "pingüinos de cola de cepillo". Estas son Adelia (Adelie), Gentú o Papúa (Gentoo) y Barbijo (Chinstrap).

Variables numéricas:

- *culmen_depth_mm:* Alto de la cresta superior del pico (culmen) medida en milímetros.
- *culmen_length_mm:* Longitud de la cresta superior del pico (culmen) medida en milímetros.
- *flipper_length_mm:* Longitud de la aleta medida en milímetros.
- *body_mass_g:* Masa del pingüino medida en gramos.

Variables categóricas:

- *sex:* Sexo (macho o hembra).
- *species:* Especie del pingüino.
- *island:* Isla de proveniencia.

b. Transformación de los datos

Primeramente, se eliminaron los ejemplos con datos faltantes o nulos, debido a que eran 11, una cantidad baja.

Las variables categóricas, al ser de clasificación en lugar de numéricas, no pueden ser directamente procesadas por el modelo. Por ello, se procesaron de dos maneras distintas.

La columna hembra podía tener dos valores distintos, 'male' o 'female'. Para transformarla a numérica, se le dio un valor binario donde todos los machos corresponden a 1 y las hembras a 0.

En el caso de la especie y la isla, ambas columnas pueden tener tres valores distintos. Una transformación numérica como en el caso anterior no es lo más ideal, debido a que el modelo comienza a asociarlo con una relación lineal, donde una categoría es “mayor o menor” que otra. Por ello, se aplicó one-hot encoding. Se crearon k - 1 nuevas columnas binarias, siendo k el número de valores únicos en la columna a dividir. Estas nuevas columnas se conocen como “variables dummy”.

Al generar k - 1 columnas, se evita caer en la trampa donde dos de las variables dummy tienen una colinealidad perfecta. Se toma como base la categoría no incluida (si todas las columnas son 0, corresponde a la categoría base). Se añadieron *species Adelie, species Gentoo, island Dream* e *island Biscoe*.

En las transformaciones numéricas, se realizó una conversión de unidades para que la diferencia entre las categorías no fuera tan grande. Los gramos pasaron a kilogramos y los milímetros a centímetros. Con esto, las columnas también fueron renombradas acorde con el cambio.

c. Detección de anomalías

Para observar el comportamiento de cada variable numérica, se realizó un histograma por cada una. Esto demostró que no existía ningún valor atípico que sesgara los datos, pero sí demostraron la tendencia de los datos a repartirse en dos grupos.

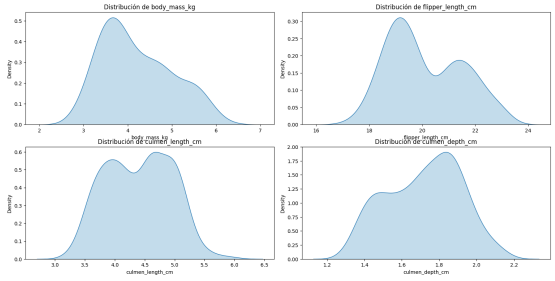


**Figura 1.** Distribución de variables numéricas

La distribución de los datos se aglomera tomando como media dos puntos distintos. Esto podría deberse a un solapamiento entre las mismas especies al ser estas parte de la misma familia. Por ejemplo, en una categoría, dos especies podrían presentar una característica similar, mientras que la tercera tiene valores distintos.

## III. PREPROCESAMIENTO DE LOS DATOS

a. Elección de variables

Para mayor entendimiento sobre cómo se relaciona la masa de los pingüinos con el resto de las categorías, se realizó una matriz de correlación. En cada celda, muestra un número entre -1 y 1, donde -1 es una fuerte relación negativa y 1, una fuerte relación positiva.

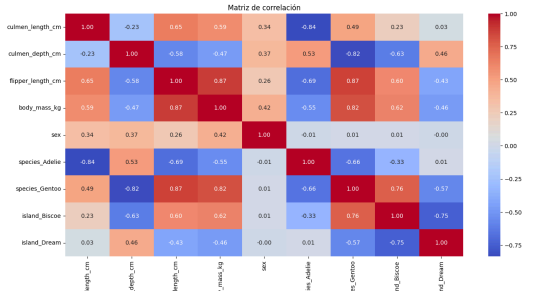


**Figura 2.** Matriz de correlación entre variables

En la fila o columna de *body_mass_kg,* se observa que, en general, tiene relaciones fuertes y medianas con la mayoría de las variables. Al no haber ninguna con correlación baja ni que se considere no importante para la predicción, se decidió conservarlas todas.

La relación entre la variable a predecir (la masa) contra cada una de las categorías se visualiza en la siguiente figura:

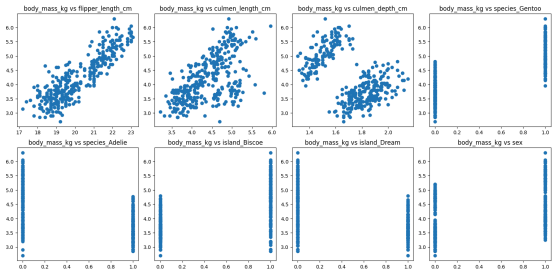


**Figura 3.** Diagramas de dispersión de las categorías selectas contra la masa en kg.

Se observa una relación cercana a lineal en las tres variables numéricas. El comportamiento de las demás gráficas se debe a que son los valores binarios (0 y 1) provenientes de las variables categóricas transformadas (*sex*, *species* e *island*). A pesar de no ser tan claro como en las numéricas, se observa una clara distinción de los valores de la masa cuando pertenecen o no a una categoría, es decir, cuando su valor es 0 o 1.

b. División de los datos

Una manera de asegurar que el modelo aprendió adecuadamente y sus resultados son acertados, es tener tres conjuntos de datos: entrenamiento (train), validación (validation) y pruebas (test). Así, el modelo aprende de datos distintos a los que se utilizan para su evaluación y no se genera una idea falsa del desempeño del mismo, sino que se realiza un diagnóstico adecuado.
En este contexto, tenemos un total de 333 instancias tras la limpieza. Con el fin de tener la mayor cantidad posible a destinar para entrenamiento, el conjunto se divide en 80% train y 20% pruebas, siendo 266 y 67 registros, respectivamente.

Posteriormente, se utiliza el algoritmo cross validation para la etapa intermedia de validación, debido a la limitada cantidad de instancias. Este funciona con el conjunto de datos de train, por lo que no es necesaria realizar otra división adicional.

## IV. MODELO DE REGRESIÓN LINEAL

a. Regresión lineal múltiple

Este modelo utiliza la regresión lineal para la predicción numérica de la masa de los pingüinos, tomando como predictoras las categorías seleccionadas. Implica su sometimiento a un entrenamiento para se
Para lograrlo, debe someterse a un entrenamiento y ajuste hasta que los resultados se acerquen a lo esperado, con ayuda de distintas funciones y algoritmos.

Este modelo sigue una regresión lineal múltiple para explicar la relación entre la variable a predecir (la masa del pingüino) respecto a las variables independientes seleccionadas.

$$y = \beta_0 + \sum_{i=1} \beta_i x_i \cdot$$

**Figura 4.** Función de hipótesis para regresión lineal múltiple.

La función de hipótesis es la ecuación lineal que describe el peso de cada variable para llegar al resultado final. El resultado de esta ecuación es la predicción de la variable dependiente.

b. Función de costo

La función de costo, también conocida como función de pérdida o función de error, compara el valor predicho por la función de hipótesis contra el valor real de nuestra variable a predecir (en este caso, la masa del pingüino). Como su nombre lo dice, mide el error o la pérdida que hubo. Este valor es de suma importancia, debido a que nos ayuda a evaluar el desempeño del modelo y saber qué tan cerca estamos matemáticamente de realizar una predicción acertada. El objetivo de este modelo será acercarlo a 0, indicando que la mayoría de los valores son acertados o

tienen poco rango de error. Se utilizan las dos ecuaciones siguientes:

$$\mathrm{MSE} = \frac{\Sigma(y_i - p_i)^2}{n}$$

**Figura 5.** Función del Mean Squared Error.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

**Figura 6.** Función del Mean Absolute Error.

La primera se utiliza en el algoritmo de optimización, para calcular el cambio en los parámetros necesario para minimizar el error. La segunda, se utiliza como error promedio interpretable en las unidades de medida de nuestra variable a predecir, en este caso, en kilogramos.

c. Gradiente descendiente

La manera en la que el modelo minimizará su error es utilizando el algoritmo de optimización gradiente descendiente (gradient descent) para su entrenamiento. Este algoritmo encuentra los parámetros de la función de hipótesis necesarios para minimizar lo más posible la función de costo. Se define como:

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta_0, \theta_1)$$

**Figura 7.** Función para calcular el nuevo valor de un parámetro con gradiente descendiente.

Para calcular el valor de un nuevo parámetro de la función de hipótesis (la ecuación lineal múltiple), a su valor actual se le resta el gradiente de la función de costo. Desarrollada para una regresión lineal múltiple, queda de la siguiente forma:

$$\Theta_j := \Theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}[(h_\Theta(x_i) - y)x_i]$$

**Figura 8.** Función del gradiente descendiente desarrollada para regresión lineal.

Hablar de hiper parámetros ****

d. Validación cruzada

El algoritmo de validación cruzada (cross validation) se utiliza para probar la eficiencia del modelo, con el objetivo de ajustar los hiper parámetros y mejorar el modelo. Se ajusta con este modelo debido a las pocas instancias en el conjunto de datos.

Del 80% de datos destinados a entrenamiento, se divide en k subpartes (en este caso 10). En cada iteración, se toman 9 partes para entrenar y una para validar, repitiendo hasta usar todos los datos para validación. Esto permite evaluar el error arrojado y ajustar los hiper parámetros, como el número de épocas a repetir el algoritmo de optimización o el learning rate (el tamaño de los pasos al optimizar).

e. Normalización de datos

La normalización de los datos es una técnica de escalamiento, la cual estandariza el rango de valores de cada característica. Esto evita que el modelo se sesgue a otorgarle mayor peso a las variables de valores más altos. Permite que cada variable tenga una contribución proporcional a la ecuación.

En este modelo, utilizaremos la estandarización o z-score, debido a que es comúnmente utilizado en regresiones lineales y no resulta tan afectado por valores extremos. Resulta favorable considerando la distribución de las categorías, aglomerada en dos grupos y con un rango de valores considerable. Está dada por la ecuación:

$$Z = \frac{X - \mu}{\sigma}$$

**Figura 9.** Función de estandarización de datos.

La estandarización se realiza tomando la media y desviación estándar por categoría del conjunto de datos destinados para entrenamiento. Se aplica a todas las categorías tanto de train como de test, a excepción de los valores predichos y por predecir.

f. Evaluación del modelo

Los resultados del modelo se evaluarán con el coeficiente de determinación $R^2$, un valor entre 0 y 1 que indica cuántos datos es capaz de explicar el modelo. Al multiplicarse por 100, es interpretable para nosotros como porcentaje. Se calcula como:

$$R^2 = \frac{\sum_{t=1}^{T} (\hat{Y}_t - \bar{Y})^2}{\sum_{t=1}^{T} (Y_t - \bar{Y})^2}$$

**Figura 10.** Función del coeficiente de determinación.

g. Criterios de diagnóstico

Hablar de under, over y fit

## V. EVALUACIÓN INICIAL

a. Diagnóstico en validación

Los datos de train se utilizaron en la validación cruzada con una constante k = 10. Así, por cada iteración se mantuvo el 90% para entrenamiento y 10% para evaluación. En cada iteración, se estandarizaron los datos tomando la media y desviación estándar del respectivo conjunto de train. Para cada iteración, se obtuvo la pérdida por épocas.

**Hiper parámetros:**

*Learning rate = 0.001*
*Número de épocas = 5000*

**Resultados:**
*MAE promedio en train = 0.9045*
*MAE promedio en validation = 0.9066*
*R2 promedio = 0.8552*
*Porcentaje de precisión = 85.52%*

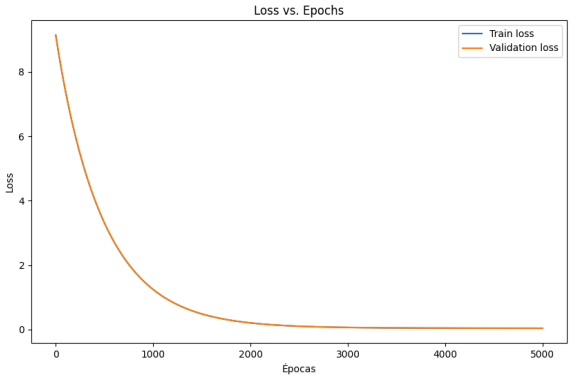


**Figura 11.** Pérdida de entrenamiento contra la de validación en cross validation con 5000 épocas.

En la gráfica, se observa que la diferencia de errores aparenta ser mínima a partir de la época 3000. Por ello, compararemos ambos resultados para evaluar la significancia de la diferencia entre ellos.

**Hiper parámetros:**
*Learning rate = 0.001*
*Número de épocas = 3000*

**Resultados:**
*MAE promedio en train = 1.3442*
*MAE promedio en validation = 1.3445*
*R2 promedio = 0.7793*
*Porcentaje de precisión = 77.93%*

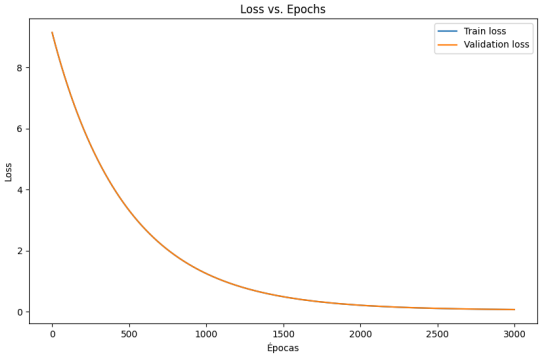

**Figura 12.** Pérdida de entrenamiento contra la de validación en cross validation con 3000 épocas.

Si bien el modelo ejecutado con 5000 épocas presenta mejores resultados, la diferencia no es tan remarcable, así como lo muestra la gráfica, indicador de que se encuentra cerca de la convergencia a partir de la época 3000. Se opta por la opción de mantener 5000 épocas, sabiendo que cualquier número mayor no tendría una mejora impactante.

En cuanto al comportamiento de la pérdida, la comparación entre entrenamiento y validación demuestra resultados casi idénticos. En ambos casos, se acerca a errores casi nulos, por lo que el modelo generado tiene un bajo sesgo al no tener resultados desajustados a lo esperado (underfitting).

Además, los resultados indican que el modelo no se sobreajustó en el entrenamiento tal que su desempeño de evaluación resultara opuesto al de train (overfitting), debido a la similaridad de resultados en distintos conjuntos de datos. El modelo se cataloga con la capacidad de aprender del conjunto de entrenamiento y adaptarse a nuevos valores (fitting).

Cabe destacar que el error promedio aún resulta alto (0.9 kg). Sin embargo, este no es un valor relevante al momento debido a que el objetivo de la validación cruzada fue emitir un diagnóstico previo con los hiper parámetros seleccionados.

b. Diagnóstico del entrenamiento en pruebas

Con la media y desviación estándar del conjunto completo de train, se estandarizaron los valores predictores de train y test, dando el siguiente modelo como resultado.

**Hiper parámetros:**
*Learning rate = 0.001*
*Número de épocas = 5000*
**Parámetros:**
Escribir aquí el valor final de los parámetros

**Resultados:**
*MAE en train = 0.229*
*MAE en test= 0.290*
*R2 promedio = 0.8142*
*Porcentaje de precisión = 81.42%*

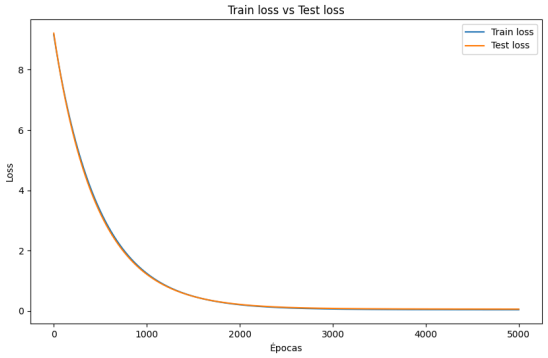


**Figura 13.** Pérdida de entrenamiento contra la pérdida de pruebas.

Así como en la validación, el comportamiento de pérdida es bastante similar entre el entrenamiento y las pruebas, tal que ambas líneas se solapan. Por consiguiente, se indica el mismo diagnóstico de que el modelo es capaz de aprender de un conjunto de datos y adaptarse a nuevos sin conocerlos previamente (fit).

El MAE dio un valor de 0.229 en el entrenamiento y 0.290 en las pruebas. Esto quiere decir que, en promedio, la pérdida de datos fue de 229 gramos en entrenamiento y 290 gramos en pruebas. En las pruebas, el modelo fue capaz de explicar el 81.4% del comportamiento de los datos.

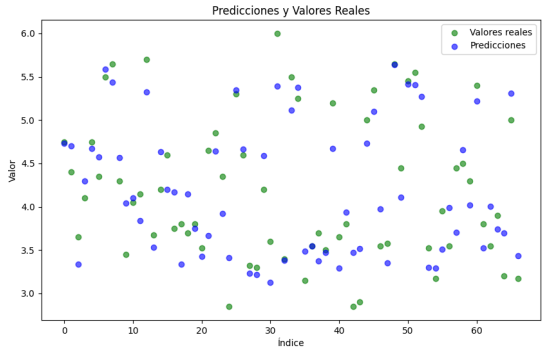


**Figura 13.** Comparación de valores predichos del modelo contra los valores reales.

En el diagrama de dispersión, se plasma el comportamiento de los valores predichos y reales sobre la masa de los pingüinos. A simple vista, se observa que tienen una varianza y comportamiento similar. Una gran parte de los puntos está sobrepuesto, indicando la predicción correcta, o muy cercano, indicando un error mínimo que no sobrepasa los 200 gramos. En el resto de puntos, aunque son menos, se observa un error mayor. Este caso se presenta más en los pesos reales muy bajos o muy altos, fuera del promedio. Esto puede indicar que al modelo le cuesta trabajo adaptarse a valores extremos.

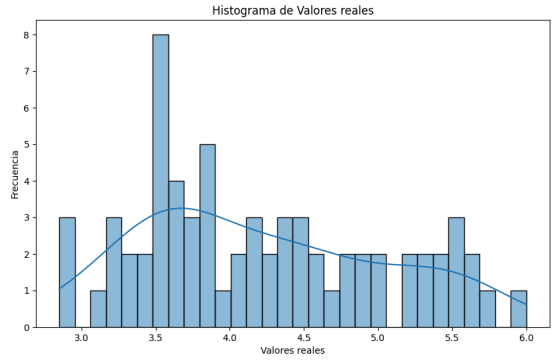


**Figura 14.** Histograma y distribución de los valores **reales** de masa (test).

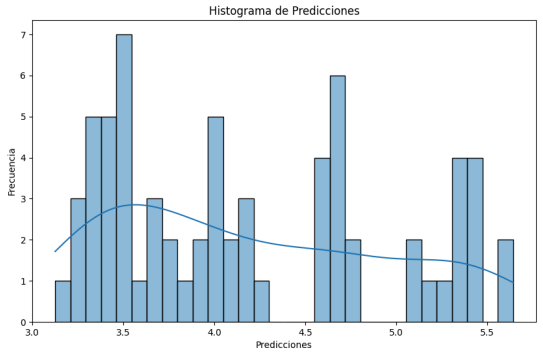


**Figura 15.** Histograma y distribución de los valores **predichos** de masa (test).

Las gráficas demuestran que la cantidad de valores repetidos no es la misma, existe una varianza evidente. Esto se explica con el coeficiente de determinación, pues hay un 18.6% de datos que nuestro modelo no alcanza a predecir adecuadamente, por lo que el modelo no es completamente preciso. Sin embargo, se observa una distribución bastante similar en ambas. Esto quiere decir que el modelo sí aprende y sigue la tendencia general de los datos.

## VI. MODELO CON RANDOM FOREST

La mejora del modelo de predicción se realizará aplicando Random Forest con la librería de python scikit-learn. Random Forest es un algoritmo que utiliza múltiples árboles de decisión débiles y los une con el método de ensamble Bagging (Bootstrap Aggregating). Los árboles generados no están correlacionados entre sí debido a la aleatoriedad de la muestra de datos y las características empleadas, lo que otorga un resultado estadísticamente significativo.

Se generan diversas muestras de los datos con reemplazo, lo que significa que una instancia individual puede repetirse en diversas muestras. Cada vez que se obtiene una muestra, se selecciona un conjunto aleatorio de características a considerar para la creación del árbol. Así, el árbol de decisión se genera tomando en cuenta las

características seleccionadas, la profundidad máxima y la cantidad máxima de hojas establecidas.

En el contexto de regresión, donde se busca predecir un valor numérico, se promedian los árboles para obtener un modelo capaz de generalizar datos.

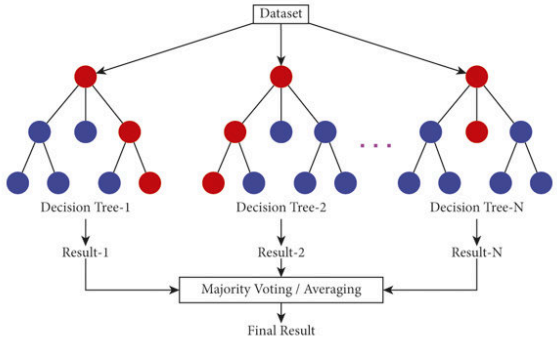


**Figura 14.** Diagrama explicativo del Random Forest.

Existen diversos hiper parámetros a definir para guiar el desempeño del modelo. Los utilizados en este son:

- **n_estimators**: Cantidad de árboles. Aumentar la cantidad aumenta el desempeño.
- **max_features**: Máximo de características a considerar por árbol.
- **max_depth**: Máximo de niveles por árbol. Un valor bajo conduce a underfitting y un valor alto, a overfitting.
- **max_leaf_nodes**: Máximo de nodos por árbol. Limita el tamaño y complejidad del árbol.
- **max_samples**: Máximo de instancias utilizadas como muestra para los árboles.
- **min_samples_split**: Mínimo de instancias necesarias para que un nodo pueda dividirse en otros.

## VII. EVALUACIÓN CON FRAMEWORK

a. Diagnóstico en validación

Al igual que el modelo anterior, los datos de train se utilizaron en la validación cruzada con una constante k = 10. Así, por cada iteración se mantuvo el 90% para entrenamiento y 10% para evaluación. En cada iteración, se estandarizaron los datos tomando la media y desviación estándar del respectivo conjunto de train. Para cada iteración, se obtuvo la pérdida final, así como los residuos por cada instancia predicha.

**Hiper parámetros:**
*n_estimators = 300*
*max_features = sqrt*
*max_depth = 10*
*max_samples = 0.8*
*min_samples_split = 4*
*max_leaf_nodes = 100*
*random_state = 42*

**Resultados:**
*MSE promedio en train = 37.23 * 10E-3*
*MSE promedio en validation = 88.53 * 10E-3*
*MAE promedio en train = 0.15*
*MAE promedio en validation = 0.2316*
*R2 promedio = 0.8487*
*Porcentaje de precisión = 84.87%*

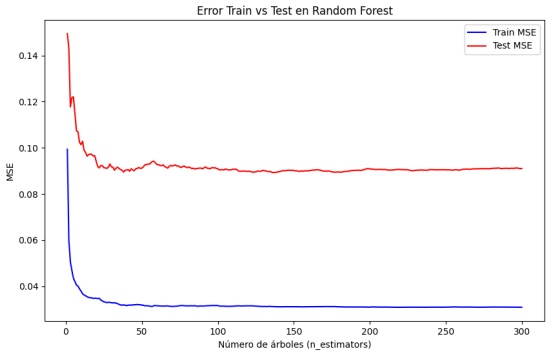


**Figura 14.** Pérdida de entrenamiento contra la pérdida de validación por cantidad de árboles.

Hacer diagnóstico ****

Parece overfit pero realmente el rango de error es muy poco.

b. Diagnóstico del entrenamiento en pruebas

**Hiper parámetros:**
*n_estimators = 300*
*max_features = sqrt*
*max_depth = 10*
*max_samples  = 0.8*
*min_samples_split = 4*
*max_leaf_nodes = 100*
*random_state = 42*

**Resultados:**
*MSE promedio en train = 37.23 * 10E-3*
*MSE promedio en validation = 88.53 * 10E-3*
*MAE promedio en train = 0.15*
*MAE promedio en validation = 0.2316*
*R2 promedio = 0.8487*
*Porcentaje de precisión = 84.87%*

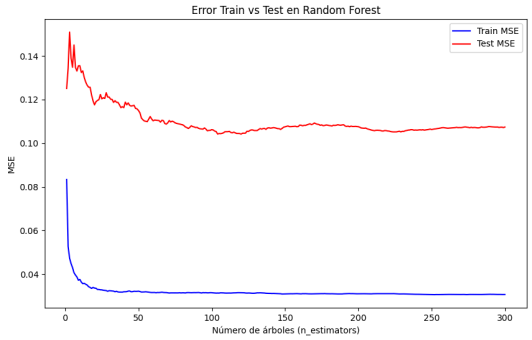


**Figura 14.** Pérdida de entrenamiento contra la pérdida de validación por cantidad de árboles.

Con la media y desviación estándar del conjunto completo

## VIII. MEJORA DEL MODELO

a. {lskfnlskdfm

lskdjfhuashdbfñWODJF

## IX. REFERENCIAS

Bobbitt, Z. (2021, 2 febrero). What is the Dummy Variable Trap? (Definition & Example). Statology. https://www.statology.org/dummy-variable-trap/

3.1. Cross-validation: evaluating estimator performance. (s. f.). Scikit-learn.

https://scikit-learn.org/stable/modules/cross_validation.html

Brownlee, J. (2016, 21 junio). How to choose the right test options when evaluating machine learning algorithms. Machine Learning Mastery. https://machinelearningmastery.com/how-to-choose-the-right-test-options-when-evaluating-machine-learning-algorithms/

Wagavkar, S. (2024, 20 noviembre). Introduction to the Correlation Matrix. Built In. https://builtin.com/data-science/correlation-matrix

Donges, N. (2024, 1 agosto). Gradient Descent in Machine Learning: A Basic Introduction. Built In. https://builtin.com/data-science/gradient-descent

Validación cruzada y k-fold cross-validation | Codificando Bits. (2024b, febrero 25). Codificando Bits. https://codificandobits.com/blog/validacion-cruzada-k-fold-cross-validation/

Jaiswal, S. (2024, 4 enero). What is normalization in machine learning? A comprehensive guide to data rescaling. Datacamp. https://www.datacamp.com/tutorial/normalization-in-machine-learning

Bergmann, D., & Stryker, C. (2025). Loss Function. IBM. https://www.ibm.com/think/topics/loss-function

Stewart, & Ken. (2025, 19 julio). Mean squared error (MSE) | Definition, Formula, Interpretation, & Facts. Encyclopedia Britannica. https://www.britannica.com/science/mean-squared-error

Siddiqui, L. (2025, 8 julio). Coeficiente de determinación: lo que nos dice el R cuadrado. Datacamp. https://www.datacamp.com/es/tutorial/coefficient-of-determination

https://scott.fortmann-roe.com/docs/BiasVariance.html

https://www.ibm.com/mx-es/think/topics/random-forest

https://www.geeksforgeeks.org/machine-learning/random-forest-hyperparameter-tuning-in-python/