Implementación de una técnica de aprendizaje máquina

Kevin Alejandro Ramírez Luna - A01711063@tec.mx

27 ago 2025

***Abstract*— The next document presents the development and implementation of a linear regression model using machine learning techniques to predict the energy demand per hour of a power plant. The model was trained using the gradient descent (GD) algorithm to optimize its performance. The process consisted of a thorough data cleaning and transformation phase, followed by an exploratory analysis to identify relevant characteristics.**

Resumen— El siguiente trabajo presenta el desarrollo e implementación de un modelo de regresión lineal utilizando técnicas de aprendizaje automático (machine learning) para predecir la demanda de energía/hora de una planta energética. El modelo se entrenó utilizando el algoritmo de descenso de gradiente (GD) para optimizar su rendimiento. El proceso consistió en una fase exhaustiva de limpieza y transformación de los datos, seguida de un análisis exploratorio para identificar características relevantes.

I. INTRODUCCIÓN

Usando el software de Python queremos demostrar el funcionamiento de una regresión lineal haciendo uso de uno de los algoritmos base para el Machine Learning; el algoritmo de Gradiente Descendente. Un algoritmo muy utilizado para problemas de regresión lineal o logística. En nuestro caso: regresión lineal para la predicción de una variable dependiente (variable numérica). Pero antes de lanzarnos a programar nuestro algoritmo, necesitamos recaudar los datos de entrada para el algoritmo. La página UC_IRVINE es un buen repositorio con varios datasets útiles en problemas de Machine Learning.

Buscando entre la gran diversidad de datasets que proponen, se seleccionó el siguiente:

Fig. 1. Dataset a analizar [1]

El conjunto de datos contiene 9568 instancias de datos recopilados en una central eléctrica de ciclo combinado durante 6 años (2006-2011), cuando la central funcionaba a plena carga.

Se seleccionó el dataset al ser un buen candidato para lo que queremos hacer: predecir datos a través de una regresión lineal. Además que cuenta con una buena cantidad de datos y llega a ser lo suficientemente desafiante para una primera demostración/acercamiento del algoritmo.

II.PREPARACIÓN DEL ENTORNO

Antes de empezar a programar necesitamos crear un entorno con todas las dependencias/librerías necesarias para la realización de esta tarea. Para ello, estaremos usando Python por su facilidad y gran catálogo de librerías de Machine Learning; tales como Numpy, Pandas, MatplotLib, Seaborn, etc.

Para no generar problemas con proyectos futuros/pasados, creamos un entorno de desarrollo Pip. Los comandos a ingresar en el shell son:

```shell
# Creación del enviroment
python -m venv IA_AVANZA

# Activación del enviroment
source IA_AVANZADA/bin/activate

# Instalación de las librerias a utilizar
pip install pandas scikit-learn numpy
openpyxl seaborn
```

Código 1. Comandos para el enviroment [2]

Cabe recalcar que lo anterior se hizo para trabajar en VisualStudio Code sin embargo, perfectamente funcionaria en otros entornos como Anaconda. En Google Collab no son necesarios los comandos anteriores al contar con estas dependencias ya instaladas.

III. ETL - EXTRACT

Una vez seleccionado el dataset, se descarga y se coloca en una carpeta local. Con esto preparado y leyendo la

documentación del dataset podemos proseguir sabiendo que el dataset cuenta con las dimensiones de 9568 instancias y 4 features. Donde las features dadas tiene la siguiente nomenclatura y significado:

Variable X	Rango
Temperature (T)	1.81°C and 37.11°C
Ambient Pressure (AP)	992.89-1033.30 milibar.
Relative Humidity (RH)	25.56% to 100.16%
Exhaust Vacuum (V)	25.36-81.56 cm Hg
Net hourly electrical energy output (EP) - Valor a predecir	420.26-495.76 MW

Fig. 2. Features del dataset

Esto será importante a la hora de debuggear y selección de variables independientes para la elaboración de nuestro *Hypothesis Model*, es decir: la regresión lineal

Con lo anterior en mente, sabemos cuales son nuestras posibilidades candidatas para la regresión, así como nuestro target Y (EP)

Al final, la estructura del proyecto estará basada en un script de Python donde hagamos la implementación y ETL del dataset, así como el respectivo dataset en nuestra carpeta local.

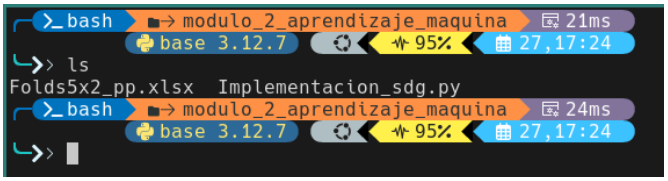

Fig. 3. Estructura del proyecto

IV. ETL - TRANSFORM

Con los datos cargamos, procederemos a hacer transformaciones y limpieza de datos para poder trabajar en la regresión lineal. La limpieza y transformación de datos debe estar orientada en:

1. Limpieza de datos

Eliminación de valores nulos: Se aplicó df.dropna() para eliminar cualquier registro con valores faltantes en el dataset (si es que los hay)

2. Escalar valores a rangos similares

Escalar valores a rangos similares para no dar más peso uno al otro usando StandardScaler para estandarizar todas las variables a una escala similar (para no tener problemas con variables X con más rango, una que otras).

3. Selección de variables

Selección de variables independientes y dependientes para la elaboración del modelo. La elección de nuestras X´s y Y´s debe estar dada de una manera cualitativa que nos permita tener un buen acercamiento a lo que queremos modelar. En este caso, para elegir de una mejor manera estas variables, se implementó una matriz de correlación:

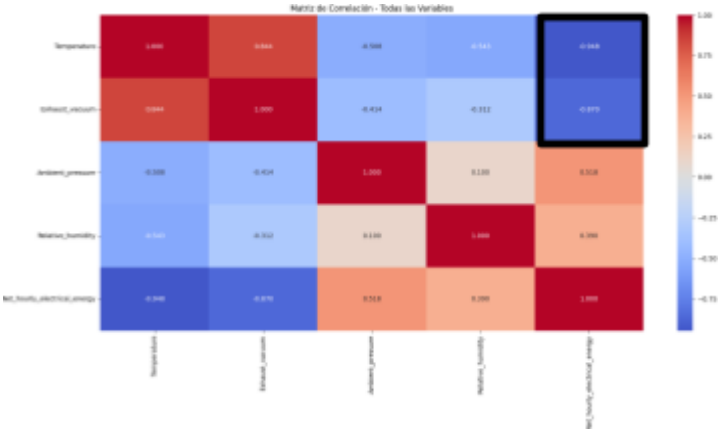

Fig. 4. Matriz de correlación

Nótese que los valores más altos en la matriz de correlación para nuestra salida llegan a ser negativas (T y AP) y son cercanas a -1.0. Esto quiere decir que hay una relación lineal negativa (hay una relación inversamente proporcional)

Conociendo esto y que estamos en un problema de regresión donde queremos conocer un valor energético, puede que estas dos variables signifiquen mucho por la mera naturaleza del problema; donde normalmente la temperatura influye mucho en la producción de energía. Esta será nuestra *hipótesis inicial* (Ho) para el modelo.

4. Redimensión de listas por matrices

Al seleccionar variables independientes y dependientes en un problema de regresión lineal es seguro que la columna Y (nuestro target) quede como una lista y no una matriz como podría ser con las filas seleccionadas en Y. Es ahí cuando usamos .reshape(-1, 1) para poder transformar una lista a una matriz (n,1).

5. División de sets de datos para entrenamiento y prueba de resultados

Dividimos nuestros datos en una proporción de 80/20: 7654 muestras para entrenamiento (80%) y 1914 para prueba (20%). Para controlar el split de nuestros datos usamos una semilla para tener una aleatoriedad controlada.

$$7654 + 1914 = 9568\, datos$$

Ecuación 1. Total de datos

V. MODELADO DE LA SOLUCIÓN

Conociendo que la ecuación de la regresión lineal está definida como

$$y = \theta_1 x_1 + \theta_2 x_2 + \ldots \theta_n x_n + \beta$$

Ecuación 2. Regresión lineal

Donde:

$\theta = parámetro\ del\ modelo\ (peso\ sináptico)$

$x = valor\ de\ entrada\ (feature)$

$\beta = bias$

Para nuestro caso específico, con las dos variables seleccionadas (Temperature y Exhaust Vacuum), nuestro modelo toma la forma:

$$Energy\ output = Temperature \cdot \theta_1 + Ambient\ Pressure \cdot \theta_2 + \beta$$

Ecuación 3. Función de hipótesis desarrollada

El proceso de implementación siguió estos pasos:

- Inicialización de parámetros iniciales: comenzamos con valores iniciales de θ y β en cero.
- Implementamos un valor de Learning Rate lo suficientemente bajo para descender lo más controlado posible. Se eligió de forma arbitraria el valor de 0.1; ya que es un valor óptimo para que los gradientes convergen de manera lenta y precisa
- Función de hipótesis: Implementamos la función que calcula las predicciones del modelo según la ecuación (3)
- Función de costo: Utilizamos el error cuadrático medio (MSE) para medir el desempeño del modelo.
- Implementación del algoritmo (SG): Implementamos el algoritmo de optimización con mini-batches.
    - Entrenamiento: Iteramos a través de los datos para ajustar los parámetros del modelo. Se hizo un total de 1000 épocas para ver la evolución del error en un número grande de iteraciones e intentando no caer en problemas de *Overfitting o Underfitting*
    - Acumulación del error: Al ir iterando, guardamos el error en cada época para graficarlo posteriormente
- Evaluación: Medimos el desempeño del modelo en datos de entrenamiento y prueba. Graficamos para la evaluación del modelo

VI. RESULTADOS Y EVALUACIÓN

Tras completar el proceso de entrenamiento con las 1000 épocas y una tasa de aprendizaje de 0.1, obtuvimos los siguientes resultados:

Parámetros del modelo final

$\theta_1(Temperature) = -12.7351$

$\theta_2(Exhaust\ Vacuum) = -4.1241$

$\beta = 454.3706$

Nótese que los coeficientes negativos para *Temperature* y *Exhaust Vacuum* coinciden con lo observado en la matriz de correlación (Fig. 4). Esto quiere decir que nuestra hipótesis (Ho) se cumplió, ambas variables son inversamente proporcionales a la salida. Esto tiene sentido físico: a mayor temperatura, menor eficiencia en la conversión de energía eléctrica.

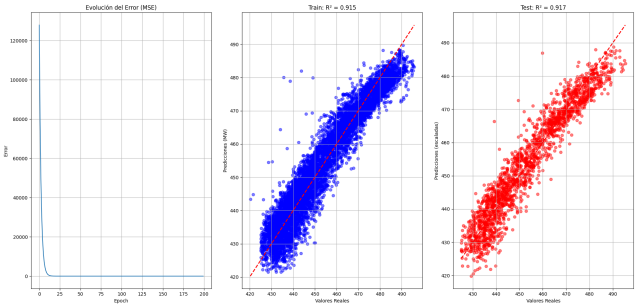

Fig. 5. Evolución del error y desempeño del modelo - 2 features

	MSE	R^2
Entrenamiento	24.6644	0.9154
Prueba	24.0683	0.9170

Fig. 6. Métricas de evaluación

Recapitulando y rescatando lo mejor de lo anterior podemos observar que el valor de la R^2 tanto en entrenamiento como en prueba indica que el modelo explica aproximadamente el 91% de la variabilidad en la producción de energía. Lo que nos indica que se ajusta de una buena manera a los datos; las entradas si se están ajustando a la salida esperada.

Al comparar la similitud entre ambas métricas de entrenamiento y prueba sugiere que el modelo no tiene problemas de overfitting, se observa que generaliza adecuadamente a datos no vistos durante el entrenamiento. Aunado a esto, se puede ver que el MSE es relativamente bajo considerando el rango de nuestra salida Ep (Fig 2.)

Sacamos nuestro RMSE para ver la diferencia entre el valor predicho:

$$RMSE_{Prueba} = \sqrt{MSE}_{Prueba}$$

$$RMSE_{Prueba} = \sqrt{24.0683}$$

$$RMSE_{Prueba} = \pm 4.9054$$

Nuestra predicción se desvía por un valor de aproximada de 4.9054MW, un valor muy bajo

VII. CONCLUSIONES y MEJORAS

Como posibles mejoras detectadas a la hora de el desarrollo de nuestro modelo de Machine Lerning se detectó que al añadir más features para tener una mayor cantidad de datos llega a ser favorable. Al añadir las otras dos features RH,V que también muestran una varianza moderada positiva (Fig 2.) se obtienen estos resultados:

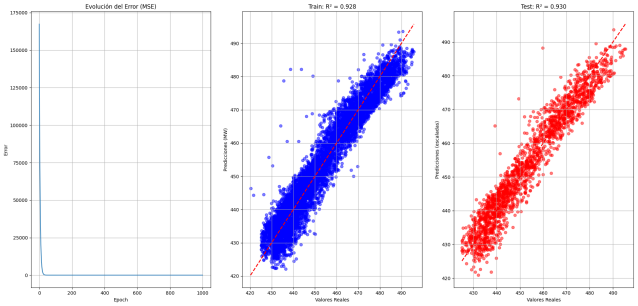

Fig. 6. Evolución del error y desempeño del modelo - Todas las features

Observando que nuestro modelo tiene un ajuste del 93% (R^2) a los datos de prueba. Dando a entender que una mayor de datos generan un mejor desempeño en el modelo.

A manera de síntesis el trabajo demostró exitosamente la implementación de un modelo de regresión lineal utilizando el algoritmo seleccionado. Mostró que se ajusta bien a los datos dados dando un buen resultado tomando en cuenta solo dos features. Sin embargo, se concluye que también considerar otras variables , por más mínimas que puedan ser, pueden favorecer en mejores/peores resultados, ya es cuestión de analizar algunas otras métricas estadísticas que nos permitan tomar estas decisiones. Aunado a esto, la importancia de la selección de un algoritmo como SGD en comparativa a GD puede ser favorable para hacer que no se necesiten tantos epoch ni procesamiento de datos más lento. Estas cuestiones dependen del tamaño de nuestros datos así como de los resultados a los que queramos llegar para poder mejorar la tasa de producción energética a cambio de un mayor gasto computacional por todo el procesamiento que hay detrás de nuestro algoritmo.

VIII. REFERENCIAS

[1] Combined Cycle Power plant. (2014, 25 marzo). UCI Machine Learning Repository. Recuperado 27 de agosto de 2025, de https://archive.ics.uci.edu/dataset/294/combined+cycle+power+plant

[2] Entornos virtuales y paquetes. (s. f.). Python Documentation. Recuperado 29 de agosto de 2025, de https://docs.python.org/es/3/tutorial/venv.html