Predicción de ventas minoristas (RETAIL SALES)

Jorge Martínez López
Tecnológico de Monterrey campus Querétaro
Querétaro, México
A01704518@tec.mx
jorgemartinez2555@hotmail.com

***Abstract*— The advancement of technologies such as artificial intelligence has improved academic and commercial sectors, although few understand how they work. In Mexico, small businesses are facing economic difficulties, with many closing. Implementing machine learning models, such as linear and logistic regression, could help optimize their performance and improve their competitiveness.**

I. INTRODUCCIÓN

El surgimiento de diferentes tecnologías vanguardistas ha generado un crecimiento exponencial en ámbito académico, laboral, social, entre otras, las cuales podemos destacar el uso de inteligencia artificial para resolver dudas, optimizar y tomar decisiones.

No obstante, son contados los sectores de la población que entiende que hay trasfondo en estos algoritmos que simulan el aprendizaje. Por ende, por esta brecha de conocimiento se generan ideas ambiguas de su funcionamiento, al contrario, su implementación ayuda a maximizar el flujo de trabajo como de productividad.

En la actualidad, gran parte de los sectores mercantiles pequeños y medianos, se ven solapados por aquellas empresas que empiezan implementar el uso de IA a sus procesos, ocasionando que los productores minoritas quiebren.

II. HISTORIA ML - PROBLEMA

El génesis de la inteligencia artificial surge entre 1943-1955, parte de tres fuentes de conocimiento de la época, las cuales radica en la filosofía básica y el funcionamiento de las neuronas, el análisis de la Lógica proporcional de Russell y la Teoría Computacional de Turing, estos trabajos fueron los cimientos de lo que se conoce hoy como redes neuronales. Sin embargo, hasta el 2010 fue que empezó a tener avances significativos las redes neuronales gracias a su desempeño en competencias, por otro lado, antes de que esto sucediera. A mediados de los 80°, dentro del área de AI surgieron algoritmos que simulan agentes inteligentes, que prácticamente son algoritmos que maximizan sus métricas de rendimiento para alcanzar un objetivo en particular, estos se conocen como autómatas, ejemplos A*, Knapsack, BFS, etc.., dando origen a dos categorías básicas de algoritmos, supervisados y no supervisados (*Artificial Intelligence: A Modern Approach, 4th Global Ed.*, 2022).

En este trabajo profundizaremos en el área de aprendizaje supervisado, que son las bases de una red neuronal.

A. Situación Actual

En la actualidad México cuanta un crecimiento exponencial del sector económicas dedicadas al comercio al por menor, se estima que de 2024 a 2027 el mercado minorista crecerá un 33.3%, para llegar a los 664 mil millones de dólares. Además, según Instituto Nacional de Estadística y Geografía, 2020 esta industria representó aproximadamente el 11.5% del Producto Interno Bruto, que engloba al 17% de la población económicamente activa (Ruiz-Healy, 2024).

Sin embargo, el mantener un comercio de este tipo es todo un desafío, se enfrenta a entornos económicos y social cambiantes.

En México tres de cada cuatro tienditas se ven obligadas a cerrar, debido a los hábitos de consumo y pobreza del cliente y del entorno, una encuetas realizada por la Anpec (Alianza Nacional de Pequeños Comerciantes), los datos arrojaron que 94% de las tienditas observa que el consumo de su cliente es bajo e insuficiente y 64% ha visto que los clientes compran porciones menore al del producto (Elizabeth Meza Rodríguez, 2022).

Por ende, para poder hacer frente a las adversidades a las que se enfrentan la mayoría de las tiendas minoristas.

Podría ser una respuesta el implementar modelos de Machine Learning para impulsar los negocios minoritas, tomando en cuenta su complejidad, su asertividad y su implementación.

III. MODELOS

Como mencionamos anteriormente, existen dos categorías de modelos de Machine Learning, supervisados y no supervisados, cada una de ellas tiene su complejidad al momento de aplicarlas, para poder abordar la situación de la manera óptima y menos trivial, es necesario acudir aquellos modelos que demanden menos recurso computacional y de almacenamiento, y que puedan ser entrenados con un conjunto de datos más pequeños o grandes.

Por ello, trabajaremos con la categoría de entrenamiento supervisado, y los dos modelos que destacan es Regresión Lineal y Regresión Logística.

A. Regresiones Lineales

$$Y = Xm + B$$

Una regresión lineal es el modelo más simple de predicción, pero el más poderoso si se implementa bien.

Un modelo de este tipo cuenta con una variable dependiente o conocida como Label que será la que va a predecir, luego X que es una variable independiente que es conocida como Features, y serán las variables de entrada para nuestro modelo, también tenemos la pendiente o beta que es m, que será la razón de cambio o mejor dicho describe como va a variar nuestro modelo, por último, el B que es conocido como el Bias que es la intercepción de nuestro modelo cuando es cero, o como va iniciar nuestro modelo.

Lo excelente de una regresión lineal, es que puede contener de una a N variables de entrada para hacer predicciones, esto se conoce como múltiples dimensionalidades.

$$Y = (X_1 m_1 \dots X_n m_n) + B_0$$

B. Regresiones Logística

$$Y = \frac{1}{1 + e^{-(Xm+B)}}$$

Una regresión logística tiene la finalidad de clasificar, esta surge de multiplicar la Regresión Lineal por una función de activación sigmoidal. La finalidad de esta función de activación convierte el valor real de xm+b entre un valor de 0 y 1, que se interpreta como la probabilidad de que la muestra pertenezca a una clase positiva, ejemplo si la probabilidad es mayor o igual a 0.5, se clasifica como clase positiva (1), y si es menor será de tipo negativo (0). Este modelo se le conoce como clasificación binaria.

No obstante, se puede utilizar múltiples parámetros de entrada para alimentar el modelo y hacer más complejo para clasificación predicción correcta.

$$Y = \frac{1}{1 + e^{-((X_1 m_1 \dots X_n m_n) + B_0)}}$$

IV. DATASET

Antes de empezar con la programación y optimización de nuestros modelos, es primordial escoger un banco de información con la que estaremos alimentando nuestro modelo. Al ser una problemática de predecir tendencias de ventas, escogí una Dataset, que podamos describir el suceso y así replicar el modelo para cualquier tienda minorista.

La Dataset fue obtenida de la organización data Montgomery, la tabla contiene una lista de ventas y movimiento por producto y departamento por mes.

El nombre de la Dataset es Warehouse and Retail Sales.

Sus features son:

- YEAR (Año)
- MONTH (Mes)
- SUPPLIER (Nombre del proveedor)
- ITEM CODE (Código del producto)
- ITEM DESCRIPTION (Descripción del producto)
- ITEM TYPE (Tipo de producto: licores)
- RETAIL SALES (Producto vendido en los dispensarios en dólares)
- RETAIL TRANSFERENS (Producto transferido a dispensarios en dólares)
- WAREHOUSE SALES (Producto vendido a licenciatarios de MC)

La base de datos cuenta con alrededor de 300,000 registros, que son suficientes para poder hacer particiones y validaciones.

A su vez con esta información podemos procesarlas para encontrar patrones en las tendencias de compras, hacer predicciones de ventas minoritas (Retail Sales), incluso predicciones de ventas por proveedor (Supplier).

Warehouse and Retail Sales.

V. IMPLEMENTACIÓN DEL MODELO

Por cuestiones dinamismo, es importante escoger un modelo entre regresión lineal y logístico, por ende, vamos a escoger el modelo que ocupe variables continuas (todo aquello que sea un valor numérico real), ya que un modelo logístico ocupa variables categóricas (todo aquello que entre un rango de cero a uno), esto conlleva modificar la tabla de la dataset.

Un modelo de Machine Learning está compuesto de la siguiente estructura para su óptimo funcionamiento:

1. Extract Transfrom Load (ETL) -Dataset
    a. Conocer la dataset
    b. Limpia de datos
        i. Imputación
        ii. Remover variable redudantes
    c. Definir variable a predecir con respecto a sus independientes.
    d. División de la muestra poblacional
        i. Set de entrenamiento 75%
        ii. Set de validación 10%
        iii. Set de prueba 15%
2. Algoritmo de machine learning
    a. Optimización de pesos
    b. Optimización de hiperparametros
3. Validación, Test y Entrenamiento
    a. Entrenamiento del modelo usando el algoritmo
        i. Primero entrenamiento
        ii. Segunda validación
        iii. Tercera prueba
    b. Evaluación del modelo
        i. Compara las pruebas y evaluarlas con estadística lineal
            1. R^2, MSE y RMSE

V.1. Extracción Transformación y Cargar (ETL siglas en inglés)

ETL se unos del paso más importante para la implementación de modelos lineales o cualquier modelo, ya que será el alimento para que nuestro modelo aprenda.

Ocuparemos las siguientes librerías para el procesamiento de datos.

- Pandas
- Numpy
- Matplotlib

Version1 paso a paso del procesamiento de datos: GitHub

	YEAR	MONTH	SUPPLIER	ITEM DESCRIPTION	ITEM TYPE	RETAIL SALES	RETAIL TRANSFERS	WAREHOUSE SALES
0	2019	3	SUTTER HOME WINERY INC	FOLIE A DEUX DRY CRK ZIN - 750ML	WINE	0.33	0.00	0.0
1	2019	3	CASTLE BRANDS USA CORP	GOSLING DK & STORMY W/GINGER BEER - 750ML	LIQUOR	0.68	0.00	0.0
2	2017	12	SURVILLE ENTERPRISES CORP	FL GUARDIAN CRIANZA - 750ML	WINE	0.00	0.00	2.0
3	2020	7	THE WINE GROUP	FISH EYE CHARD - 1.5L	WINE	7.27	7.83	38.0
4	2019	4	LEGENDS LTD	SOUTHERN TIER 2XVARIETY PACK 2/12 NR	BEER	0.00	0.00	2.0
...	...	...	...	...	...	...	...	...
307640	2017	12	BACCHUS IMPORTERS LTD	KAPNOS PRIVATE LABEL RED - 750ML	WINE	0.00	0.00	2.0
307641	2019	9	TI BEVERAGE GROUP LTD	VAMPIRE CAB - 750ML	WINE	0.08	0.00	0.0
307642	2017	12	LEGENDS LTD	OKOCIM OK NR 20/CS - 16.9OZ	BEER	4.50	6.00	0.0
307643	2018	2	MICHAEL R DOWNEY SELECTIONS INC	VIETTI BARBERA D'ASTI VIGNE - 750ML	WINE	0.16	0.00	1.0
307644	2017	12	JOS VICTORI WINES	IN SITU RESA S/BLC - 750ML	WINE	0.64	0.00	15.0

307645 rows × 8 columns

Img 1. Visualización de la Dataset original.

Visualizamos a más profundidad los datos.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307645 entries, 0 to 307644
Data columns (total 9 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   YEAR              307645 non-null  int64
 1   MONTH             307645 non-null  int64
 2   SUPPLIER          307478 non-null  object
 3   ITEM CODE         307645 non-null  object
 4   ITEM DESCRIPTION  307645 non-null  object
 5   ITEM TYPE         307644 non-null  object
 6   RETAIL SALES      307642 non-null  float64
 7   RETAIL TRANSFERS  307645 non-null  float64
 8   WAREHOUSE SALES   307645 non-null  float64
dtypes: float64(3), int64(2), object(4)
memory usage: 21.1+ MB
```

Img 2. Observación de tipos de variable de cada columna.

Echamos un vistazo general de los datos de cómo están estructurados.

	YEAR	MONTH	RETAIL SALES	RETAIL TRANSFERS	WAREHOUSE SALES
count	307645.000000	307645.000000	307642.000000	307645.000000	307645.000000
mean	2018.438525	6.423862	7.024071	6.936465	25.294597
std	1.083061	3.461812	30.986238	30.237195	249.916798
min	2017.000000	1.000000	-6.490000	-38.490000	-7800.000000
25%	2017.000000	3.000000	0.000000	0.000000	0.000000
50%	2019.000000	7.000000	0.320000	0.000000	1.000000
75%	2019.000000	9.000000	3.267500	3.000000	5.000000
max	2020.000000	12.000000	2739.000000	1990.830000	18317.000000

Img 3. Estructura de la dataset, mean, std, min, max, etc..

Al conocer un poco más la dataset, es importante replantearnos con exactitud cuántos valores hay con NAN, que son valores que aportan ruido a nuestro modelo.

```
YEAR                0
MONTH               0
SUPPLIER          167
ITEM CODE           0
ITEM DESCRIPTION    0
ITEM TYPE           1
RETAIL SALES        3
RETAIL TRANSFERS    0
WAREHOUSE SALES     0
dtype: int64
```

Img 4. Datos con Nan de la dataset.

Es importante eliminarlos para evitar ambigüedades, debidos a que representa aproximadamente 0.055% de la dataset total, no hay mucho impacto el eliminar estos los valores.

Cambiamos las variables categóricas por valor numérico para poder procesarlo con nuestro modelo. Ocuparemos codificación numérica. Lo que hacemos es cambiamos las variables categóricas por la cantidad de repetición que hay en la columna, aplicaremos esta columna para ITEM DESCRIPTION, SUPPLIER y ITEM TYPE un número por cada categoría única.

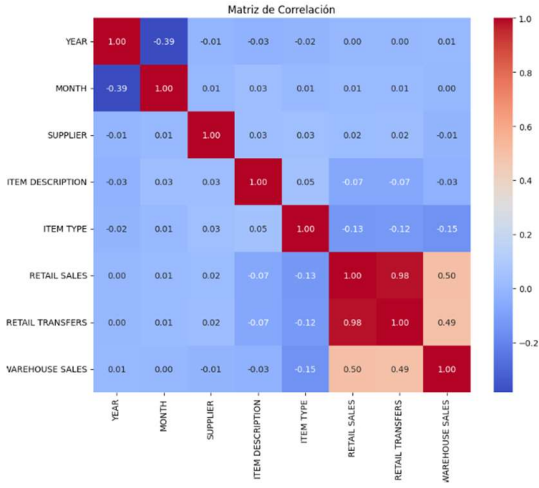
	YEAR	MONTH	SUPPLIER	ITEM DESCRIPTION	ITEM TYPE
0	2019	7	1	1	1
1	2019	10	2	2	2
2	2017	9	1	3	3
3	2019	9	3	4	3
4	2019	7	4	5	3
...	...	...	...	...	...
307640	2017	11	18	4893	3
307641	2019	11	74	12053	2
307642	2019	4	15	5031	2
307643	2019	7	19	21573	3
307644	2019	5	15	8493	2

307477 rows × 8 columns

Img 5. Visualización de cambio de SUPPLIER, ITEM DESCRIPTION y ITEM TYPE a valor numérico.

En este momento es sumamente importante encontrar correlaciones de las variables dependientes con la variable independiente.

Graf 1. Gráfica de correlaciones entre variables.



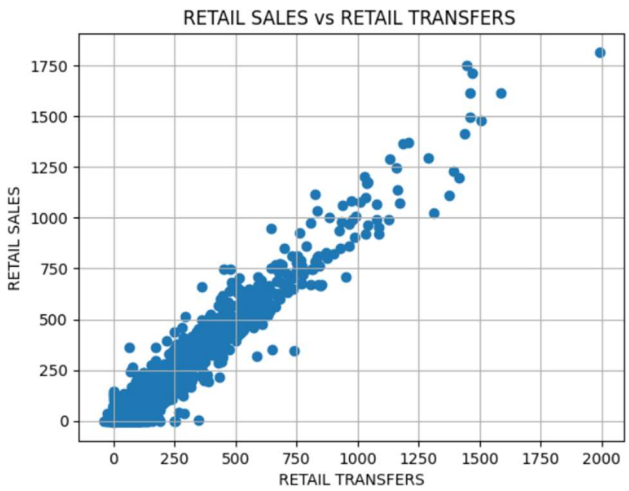
A simple vista se ve una correlación importante de RETAIL TRANSFERS con RETAIL SALES, no obstante, requerimos de una visualización más exacta de aquellos variables independentes que influyen en la variable RETAIL SALES.

Substraemos todos aquellos valores mayores o menores a 0, esto significa que visualizaremos las correlaciones positivas y negativas con respecto a RETAIL SALES.

```
RETAIL SALES          1.000000
RETAIL TRANSFERS      0.979434
WAREHOUSE SALES       0.501256
SUPPLIER              0.016976
MONTH                 0.011550
YEAR                  0.003717
ITEM DESCRIPTION     -0.068377
ITEM TYPE            -0.127319
Name: RETAIL SALES, dtype: float64
```

Img 6. Correlaciones positivas y negativas de las variables.

Vamos a visualizar con mayor detalle RETAIL TRANSFERS y WAREHOUSE SALES porque cuentan con mayor correlación.



Graf 2. Correlación de RETAIL SALES con RETAIL TRANSFERS.

El comportamiento de esta variable es lineal debido a su tendencia, podemos deducir que esta variable tiene un alto impacto positivo para poder describir el comportamiento de RETAIL SALES.

Graf 3. Correlación de RETAIL SALES con RETAIL TRANSFERS.

En la gráfica 3 existe cierto patrón, no obstante, debido a su dispersión y origina que disminuya la correlación con la variable de investigación.

Evaluamos las variables con estadística generar para una previsualización de nuestro modelo.

OLS Regression Results

Dep. Variable:	RETAIL SALES	R-squared:	0.960
Model:	OLS	Adj. R-squared:	0.960
Method:	Least Squares	F-statistic:	1.048e+06
Date:	Tue, 03 Sep 2024	Prob (F-statistic):	0.00
Time:	10:40:37	Log-Likelihood:	-9.9208e+05
No. Observations:	307477	AIC:	1.984e+06
Df Residuals:	307469	BIC:	1.984e+06
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-42.5618	22.225	-1.915	0.055	-86.122	0.998
YEAR	0.0212	0.011	1.923	0.054	-0.000	0.043
MONTH	0.0198	0.003	5.764	0.000	0.013	0.027
SUPPLIER	-0.0014	0.000	-7.110	0.000	-0.002	-0.001
ITEM DESCRIPTION	-1.069e-05	1.41e-06	-7.568	0.000	-1.35e-05	-7.92e-06
ITEM TYPE	0.0290	0.016	1.863	0.062	-0.002	0.060
RETAIL TRANSFERS	0.9717	0.000	2303.722	0.000	0.971	0.973
WAREHOUSE SALES	0.0030	5.15e-05	57.809	0.000	0.003	0.003

Img 7. Tabla de regresión lineal con indicadores de tendencias estadísticas.

Con siete features para el label RETAIL SALES, se llega a conseguir a aproximadamente con una r cuadrada de 96%, nuevamente sobre sale el coeficiente RETAIL TRANSFERS que cuenta con un 97% de impacto en el modelo.

Con estos resultados relativos, da pie a empezar a seccionar nuestra dataset en segmentos de entrenamiento, validación y prueba.

- Set de entrenamiento 75% → 230,607 datos
- Set de validación 10% → 30,747 datos
- Set de prueba 15% →46,121 datos

V.2. Implementación de Algoritmo de Machine Learninng

Como se ha mencionado anteriormente se ocupará regresión lineal para predecir las ventas al por menor. Un modelo de Machine Learning se caracteriza por su auto ajuste de parámetros para mejor las predicciones o los resultados de un modelo, por lo que esta es la gran diferencia de Machine Learning a un modelo de regresión con simple estadística. Estos algoritmos cuentan con el siguiente funcionamiento.

1. Parámetros de entrada, Bias, Features y Labels.
2. Escalamiento de los samples de entrenamiento, prueba y validación.
3. Algoritmo de autoajuste Descenso por gradiente que simula el autoaprendizaje.
4. Validación con los datos
5. Evaluar el modelo con el segmento de prueba.
6. Visualización del error.

Entonces, los features serán nuestras entradas para el modelo

que nos permitirá predecir la etiqueta label que es nuestra variable dependiente.

Features → Year, Month, Supplier, Item Description, Item Type, Retail Transfers y Warehouse Sales.

Label → Retail Sales.

Lo anterior estará encapsulado de la siguiente manera

Datos de entrenamiento 75%

- X_train
- Y_train

Datos de validación 10%

- X_val
- Y_val

Datos de prueba 15%

- X_test
- Y_test

Después de tener todo estructurado para alimentar el algoritmo, a los samples X_train, X_val y X_test es necesario de estandarizarlos, o normalizarlos, con el fin de aumentar la velocidad de convergencia y que las entradas estén dentro de un tipo de dominio en igual.

Existen muchas técnicas de normalización y estandarización, para cuestiones prácticas ocuparemos la técnica Mean-Max Scaling.

Mean-Max Scaling

$$Scaled = \frac{X_{ij} - Mean_i}{Max_i}$$

Scaled → Array estandarizado de la columna

X_ij → Iteración de la columna en valor

Mean_i → Promedio de la columna

Max_i → Máximo de la columna

```
def scale_samples(self, samples):
    """Scales the sample values for better gradient descent performance."""
    samples = np.array(samples).T #Transpuesta para ir columna por columna
    for i in range(1, len(samples)): #Un for para recorrer la columna
        avg = np.mean(samples[i]) #El promedio de la columna
        max_val = np.max(samples[i]) #El valor máximo de la columna
        samples[i] = (samples[i] - avg) / max_val #Estandariza
    return np.array(samples).T.tolist() #Regresa al formato original
```

Alg 1. Mean-Max Scaling.

Esta consiste en normalizar los features para que estén dentro de un rango de -1 a 1, ya que al restar por la media los datos, esto se ubicarán alrededor del cero y al dividirlos se localizarán entre un rango de -1 a 1 aproximamente.

Al tener nuestros samples de entrenamiento, validación y prueba, podemos empezar a entrenar nuestro modelo con el sample de entramiento.

El entrenamiento consiste auto-ajustar los coeficientes del modelo dependiendo del error de predicción de la muestra, matemáticamente lo podemos demostrar con la fórmula de descenso por gradiente, esta surge del principio de la regla de la cadena, para mayor información, cabe mencionar que es el principio básico de las regresiones logísticas y de los modelos de Deep learning.

Gradient Descent

$$\theta_j = \theta_j - \frac{\alpha}{m}\sum_{i=0}^{n}[h_\theta(x_i) - y_{real})x_i]$$

α → ir o learning rate

m →valores de iteraciones del dataset

$h_\theta(x_i)$→Hipótesis

y_{real}→Valor del label en esta instancia de la hipótesis

x_i→Valor de x en la hipótesis de esa iteración

θ_j→Valor del parámetro o theta dado por nosotros

La fórmula anterior consiste en una serie de pasos secuenciales repetitivos para alcanzar el mejor ajuste de los pesos, estos pasos son los siguientes:

1. Estructuración de la hipótesis (Nuestro modelo de regresión lineal)
2. Encapsulamos la función hipótesis para estructuración de la función de costo (Mean Squared Error)
3. Encapsulamiento de la funcion costo para la estructuración de la función descenso por gradiente
4. Actualización de los coeficientes

```
def gradient_descent(self, samples, y):
    """Performs one iteration of gradient descent."""
    temp_params = np.copy(self.params) #Copiamos nuestra betas
    for j in range(len(self.params)): #Recorremos cada beta
        gradient = 0 #El gradiente para cada beta
        #Función de costo
        for i in range(len(samples)):#Recorremso para renglon
            error = self.h(samples[i]) - y[i] #Error
            gradient += error * samples[i][j] # Por su valor real
        temp_params[j] -= self.learning_rate * gradient / len(samples) #Actualizamos
    self.params = temp_params #Devuelve array de las betas actualizadas de un ciclo
```

Alg 2. Gradient Descent.

V.3. Validación, Test y Entrenamiento

Por último, validación del algoritmo, que es medir el error de precisión de nuestro modelo entrenado, para ello la segmentación de las muestras de validación (test y validation). Veremos su rendimiento al someterlo a estas pruebas.

Al tener muestras reales de la variable a predecir, alimentamos nuestro modelo entrenando para calcular y visualizar qué tanto se equivoca nuestro modelo en predecir por los outlayers esperado de la muestra real, y calculamos su error con el MSE o función de costo. Esto se puede graficar para observar el error.

Mean Squared Error

$$Mse = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Mse → Mean Squared Error

y_i → Iteración de la columna en valor

hat-y_i → Promedio

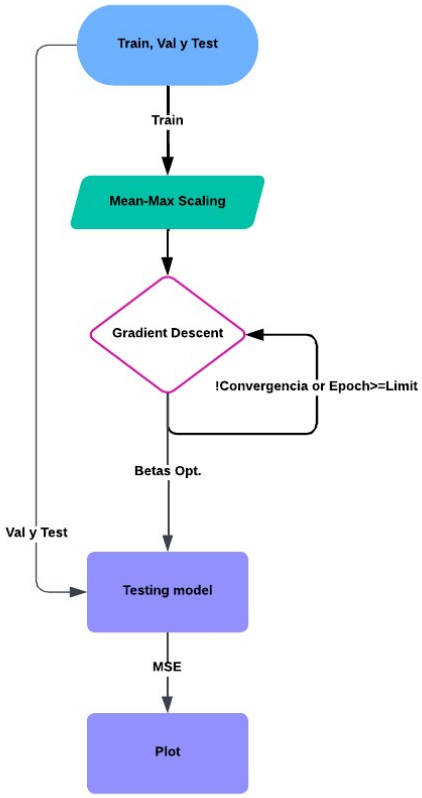
```python
def compute_error(self, samples, y):
    """Calculates the mean squared error."""
    total_error = 0 # Error total
    for i in range(len(samples)): #Iteracion
        error = self.h(samples[i]) - y[i] #Error entre y-hatY
        total_error += error ** 2 # Eliminar los negativos y suma
    mean_error = total_error / len(samples) #Calcula el promedio del error
    return mean_error # Devuelve el error promedio de todo el sample
```

Alg 3. Mean Squared Error.

VI. Algoritmo completo de una regresión lineal (Modelo 1)

Prácticamente, su funcionamiento se vería de la siguiente manera.

Cabe recalcar que primeramente se entrena y luego se evalúa la muestra de validación y al último el segmento prueba.



Diag 1. Funcionamiento del algoritmo de regresión lineal

En código completo se vería de la siguiente manera.

```python
#----Scaling----
# TRAIN
samples = X_train.tolist()
for i in range(len(samples)):
    samples[i] = [1] + samples[i]
samples = self.scale_samples(samples)
# VAL
val_samples = X_val.tolist()
for i in range(len(val_samples)):
    val_samples[i] = [1] + val_samples[i]
val_samples = self.scale_samples(val_samples)
# TEST
test_samples = X_test.tolist()
for i in range(len(test_samples)):
    test_samples[i] = [1] + test_samples[i]
test_samples = self.scale_samples(test_samples)
```

Alg 4. Escalamiento de los samples del algoritmo de Machine Learning de regresión lineal

```python
#----Loop----
epochs = 0
while True:
    old_params = np.copy(self.params)
    #----Gradient Descent----
    self.gradient_descent(samples, y)
    #----Training Error
    train_error = self.compute_error(samples, y)
    self.train_errors.append(train_error)
    epochs += 1
    if np.array_equal(old_params, self.params) or epochs >= self.max_epochs:
        break
#----Testing Error----
self.val_error = self.compute_error(val_samples, y_val.tolist())
self.test_error = self.compute_error(test_samples, y_test.tolist())
#----Plot Error
print(f"Training completed after {epochs} epochs")
print(f"Final parameters: {self.params}")
print(f"Final training error: {self.train_errors[-1]}")
print(f"Final validation error: {self.val_error}")
print(f"Final test error: {self.test_error}")
self.plot_errors()
```

Alg 5. Encapsulamiento del funcionamiento de Gradient Descent, Testing Model, MSE (Test y Val) y Plot.

Tuvimos que dividir el código completo, ya que en la sección del scaling se hace para cada uno de los samples de validez.

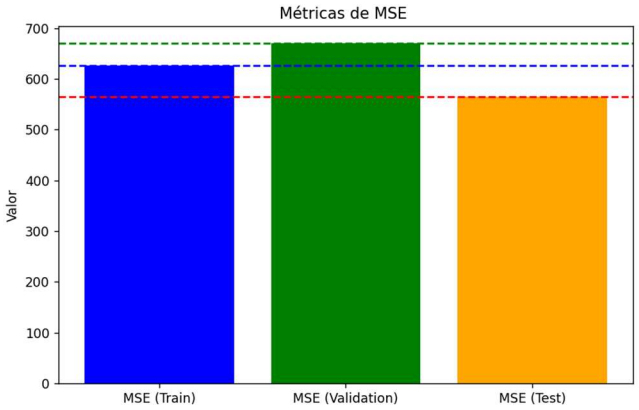
Una vez ya planteado el funcionamiento básico de los algoritmos de Machine Learing, podemos evaluar nuestro modelo hecho desde cero, con finalidad de empezar a observar su funcionamiento para predecir la variable RETAIL SALES, además que tan rentable fue implementar un modelo de estos desde cero. Código GitHub

```
MODELO 1
----MSE----
Train MSE: 626.368077293516
Validation MSE: 670.2571568829834
Test MSE: 564.5600148647733
----R^2----
Train R^2: 0.3280609177624131
Validation R^2: 0.3631540919791518
Test R^2: 0.2882083561931251
----Time----
Seg: 24415.15
----Sesgo o Bias----
Train: 2.662120401618005e-16
Validation: -0.11642208752858568
Test: 0.2937749213141736
```

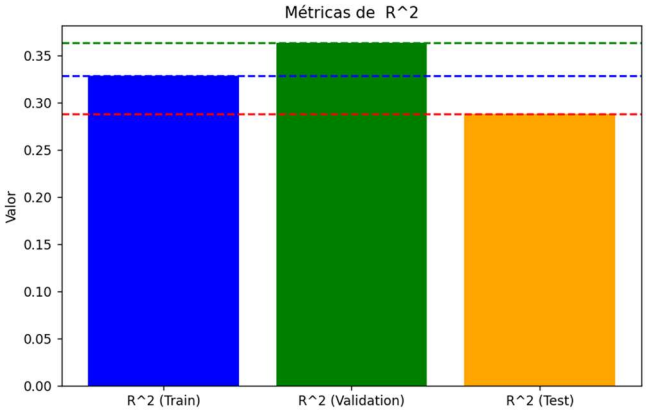
Img 8. Resultados numéricos del modelo 1.

Es posible precisar que el modelo está muy mal optimizado debido a que se tardó entrenándose 6.7hrs con hiperparámetros de 1000 épocas de entrenamiento con una tasa de aprendizaje de 0.5, y su rendimiento es prácticamente nulo, ya que el error MSE es de train(626), val(670) y test(564) unidades de variación para predecir las ventas al por menor a futuro, además el modelo en promedio tiene una capacidad descriptiva del 32%. Sin embargo, al tener mucha variación en el error y un sesgo mayo a 0.1 podemos concluir que los modelo está underfitting.

Graf 4. MSE Modelo 1 contra variables.

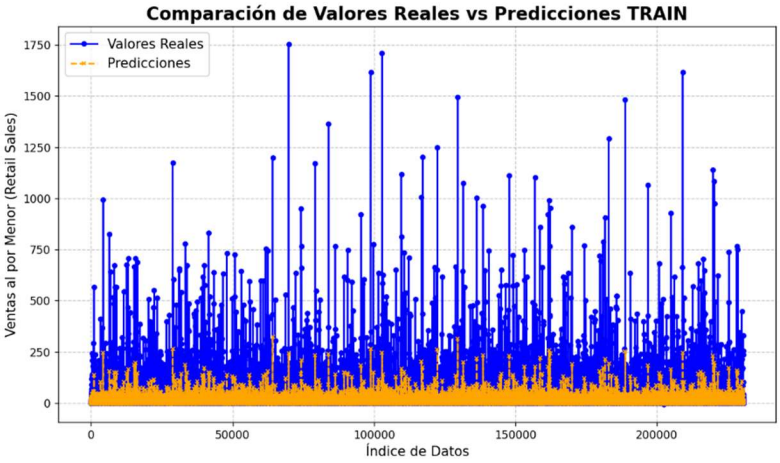


En general el modelo cuenta con error MSE muy alto, esto conlleva que sus estimaciones varíen mucho y sean imprecisas.



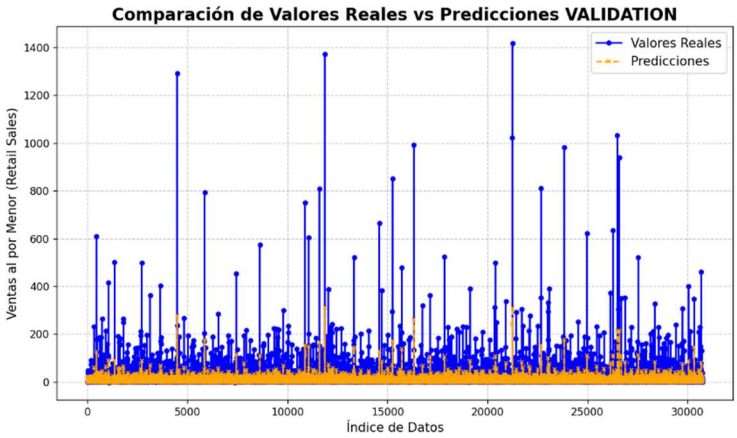
Graf 5. R^2 del Modelo 1 contra variables.

Debido a su imprecisión, el modelo se llega a considerar underfiting, esto se traduce a un modelo muy malo para predecir tendencias, y que no pude predecir ni las métricas con la que se entrenó.

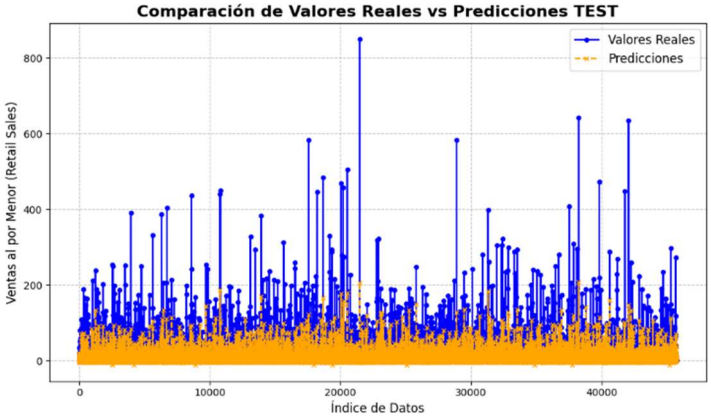


Graf 6. Predicciones Train vs Real Modelo 1.

El modelo cuenta con underfitting por su rendimiento precario, pues en el modelo no pudo predecir los valores con los que se entrenó, esta es la prueba más sencilla para una previsualización.



Graf 7. Predicciones Validation vs Real Modelo 1.



Graf 8. Predicciones Test vs Real Modelo 1.

El desempeño de nuestro algoritmo fue muy malo, esto se puede deber a múltiples factores, no obstante, los que destacan son: Optimización, Scaling, y Tiempo de entrenamiento.

Optimización: Nuestro modelo, utiliza estructuras cíclicas, esto quiere decir que los procesos son loops que van calculando por iteraciones, al utiliza tipos de estructuras así, la complejidad sube a O(n), entre mayor la dataset, menos eficiente se vuelve y tarda más.

Scaling: Hay distintos tipos de escalamiento, y cada uno tiene la posibilidad de que se adapte distinto a tu dataset favoreciendo al mejoramiento del entrenamiento, los más comunes son Mean-Max y Min-Max, por los que, es importante probar con diferentes escalamientos.

Tiempo de entrenamiento: Este punto tiene una correlación con la optimización del código, ya que, si entrenamos el modelo con una cantidad alta de épocas, va a tardar 10^n iteraciones debido a su complejidad del algoritmo.

VII. Algoritmo con Framework de una regresión lineal (Modelo 2)

De otro modo, es posible mejorar drásticamente la predicción de nuestro modelo debido a que ya existen librería optimizada para el uso de algoritmos de Machine Learning, estas librerías constan de codigo encapsulado como lo es nuestro algoritmo hecho por su servidor anteriormente, la diferencia radica en que estos están optimizados. Estas son algunas librerías:

- Scikit-learn
- TensorFlow
- PyTorch
- Keras

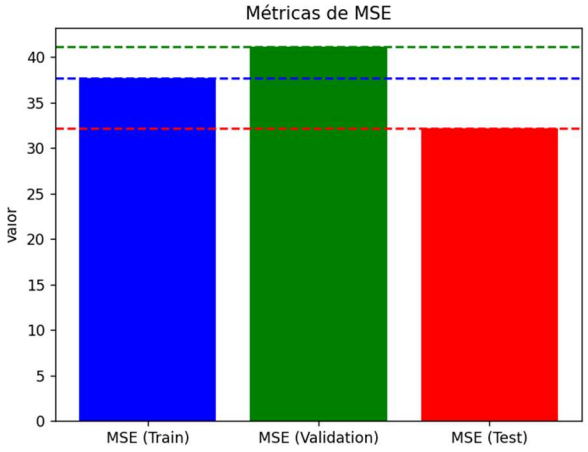
Por ende, vamos a implementar el mismo algoritmo con framework para comparar el primer modelo con este, y ver cuánto mejora. Código: GitHub
Utilizaremos el Framework (Scikit-learn). El modelo nos arroja la siguiente información:

```
MODELO 2
----MSE----
Train MSE: 37.662000035583524
Validation MSE: 41.13160749110712
Test MSE: 32.19110091683725
----R^2----
Train R^2: 0.9595979254746488
Validation R^2: 0.9609187374546754
Test R^2: 0.95941378058267 2
----Time----
Seg: 0.07
----Sesgo o Bias----
Train: 8.701189831214406e-17
Validation: 0.007940620313413695
Test: -0.0011679065544221497
```

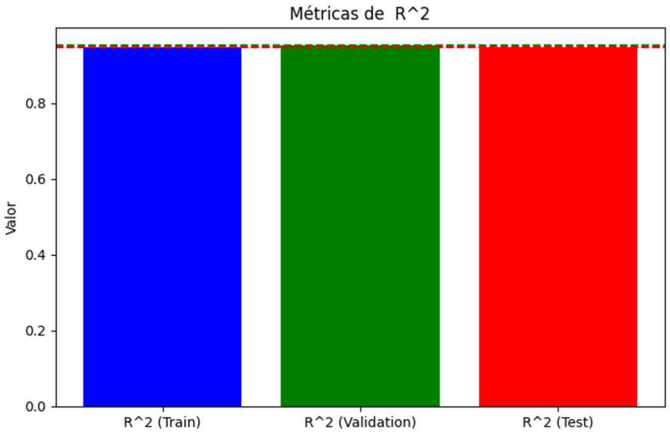
Img 9. Resultados numéricos del modelo 2.

El promedio MSE es 36 unidades, esto significa que, nuestro modelo tendrá un error de predicción de 36 unidades. Asimismo, el promedio de R^2 es de 0.95, que representa que nuestro modelo describe el comportamiento con una precisión del 95%. Se tardó entrenándose 0.07 segundos. Además, el bias de train es cero, el modelo se ajusta muy bien a los datos de entrenamiento, con validation -0.008 es un bais bajo y negativo, que significa el modelo subestima ligeramente los valores verdaderos, y en test es de -0.001 que el modelo está fitting por su sesgo cercano a cero.

Graf 9. MSE del Modelo 2 contra variables.

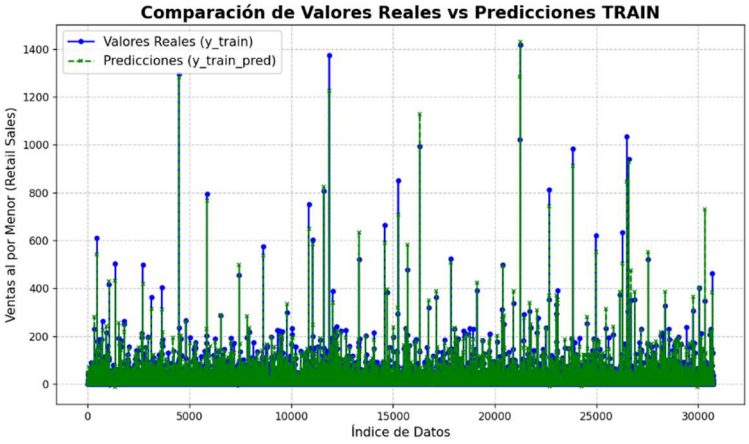


Como podemos ver nuestro modelo no está sobre ajustado, ya que el MSE de Train es de 37 unidades, sino seria de cero o menor a los otros MSE, en cambio el error MSE para test es de 41 y val de 32 unidades, por lo que concretamos que nuestro modelo predice los RETAIL SALES a futuro con margen de error mucho más pequeño comparado al modelo 1.



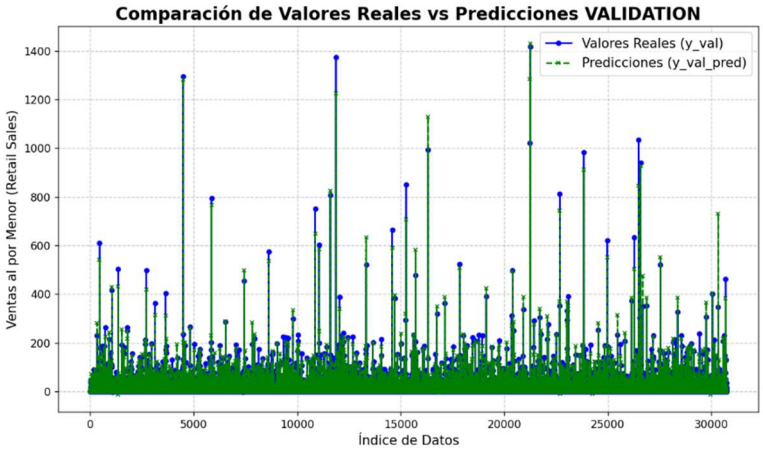
Graf 10. R^2 del Modelo 2 contra variables.

En teoría, el modelo cuenta con coeficientes con fuerza explicativa para predecir RETAIL SALES a futuro con un 95% de efectividad aproximadamente de todas las pruebas.



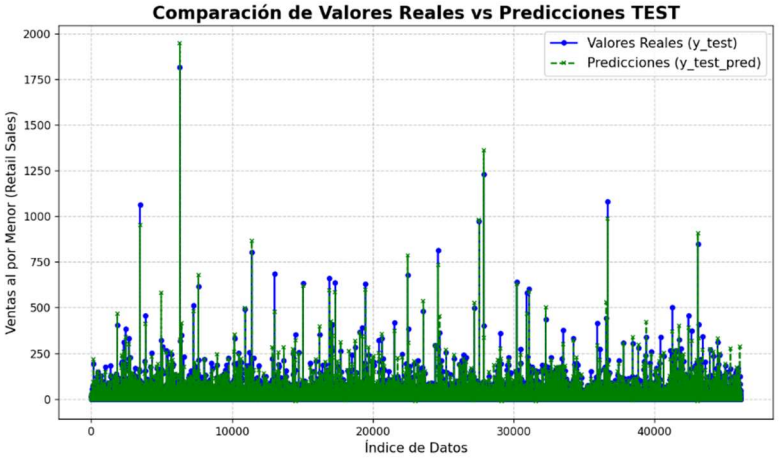
Graf 11. Predicciones Train vs Real Modelo 2.

El modelo no tiene ovedrfitting, ya que su error MSE de 36, y tampoco es underfitting debido a su precisión descriptiva, no obstante, es la primera prueba, que es la predicción con los datos de entrenamiento.



Graf 12. Predicciones Validation vs Real Modelo2

En esta prueba el modelo obtuvo un MSE menor al train y test, y con una R^2 de 95% de poder descriptivo. Podemos ver que hacer el sesgo cercano cero, tiende los modelos a tener mejores predicciones.



Graf 13. Predicciones Test vs Real Modelo 2.

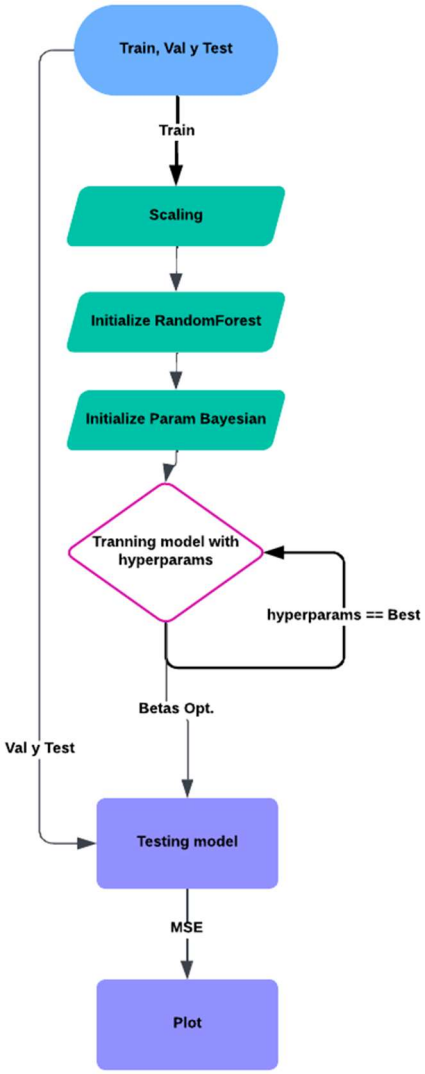
En esta prueba los rendimientos del modelo fueron de MSE de 32 unidades, su poder descriptivo del 95%.
Podemos decir que este modelo está fitting debido a que su sesgo de las dos pruebas es cercano a cero, que predice los valores reales con una eficiencia del 95% debido a que superó las tres pruebas con valores decentes, de MSE, R^2 y Sesgo.

VIII. Algoritmo RandomForest Modelo 3

Vamos ahora a comparar el modelo de regresión lineal con otro modelo más robusto, y visualizar si hay mejoras.
Ocuparemos el algoritmo RandomForest, es un método que consta de una arquitectura bagging con árboles de decisión, este consta de tener múltiples arboles pequeños que se entrenar en paralelo.

Esto se traduce, que tendremos pequeños arboles de decisiones que van subdivididos, hasta estar en el rango esperado para predecir. No entraremos a detalle al modelo matemático, solo de su implementación con framework.

Librerías para ocupar de Machine Learning son:
Sklearn y Skopt; esta última librería la ocuparemos para optimizar los hiperparámetros de nuestro modelo con el fin obtener mejores estimaciones.
Código: Github
En la estimación de los hiperparámetros óptimizados para los árboles, hay miles de técnicas, la que ocuparemos se llama Bayesian Optimization, consiste en la búsqueda del conjunto óptimo de hiperparámentros, modelando la función de rendimiento como una función probabilística, y busca en las áreas más relevantes. La gran ventaja de esta técnica es que explora de manera inteligente el espacio del hiperparámetro.



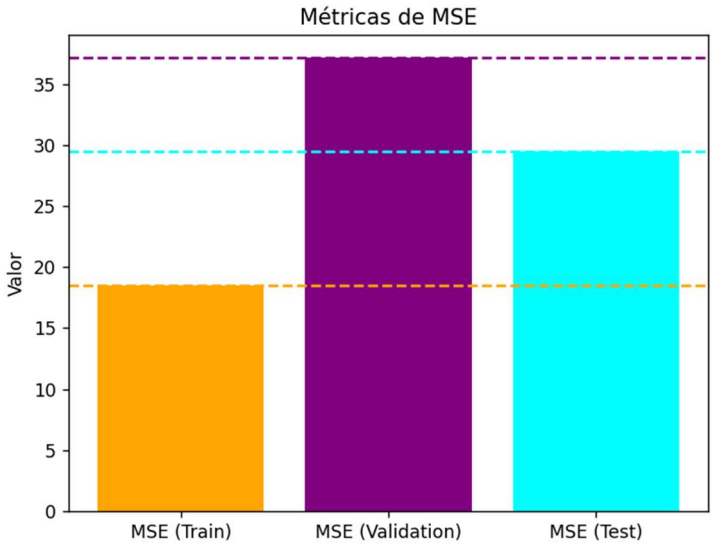
Diag 2. Funcionamiento de RandomForest con Bayesian.

Al entender funcionamiento del modelo, es posible correr el algoritmo.

```
----MSE----
Train MSE: 18.53296341069122
Validation MSE: 37.15449125579457
Test MSE: 29.497800425811857
----R^2----
Train R^2: 0.980118682805297
Validation R^2: 0.9646976007971578
Test R^2: 0.962809467016881
----Time----
Seg: 27129.99
----Sesgo o Bias----
Train: 0.0019162677394928612
Validation: 0.0037845259815234567
Test: -0.005996032301792962
```

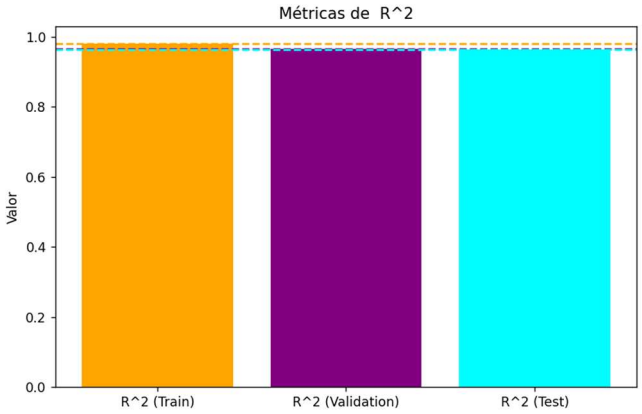
Img 10. Resultados numéricos del modelo 3.

El modelo obtuvo una precisión del 96% en promedio, un MSE de 28 unidades en promedio, el modelo le tomó 7.5hrs en optimizarse y entrenar, además el sesgo es muy cercano a cero para cada uno de los casos, por lo que podemos decir que nuestro modelo está fitting.



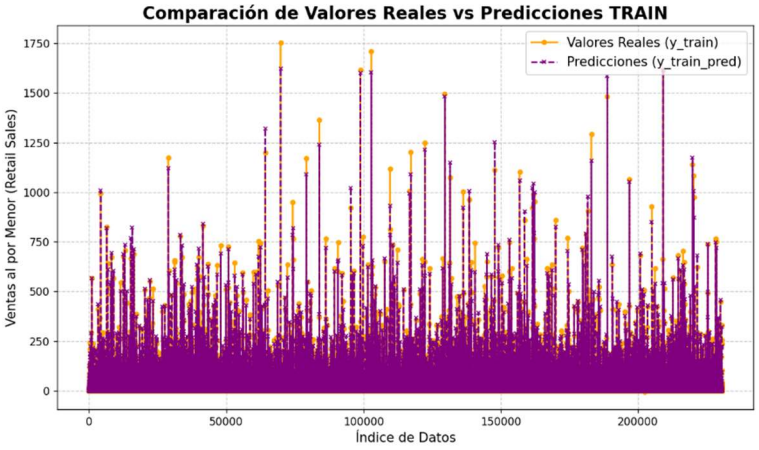
Graf 14. MSE del Modelo 3 contra variables.

El modelo pareciera que tuvo un sobre ajuste por la prueba de train que obtuvo un error en promedio de 18 unidades, no obstante, en general error se redujo a por debajo de las 40 unidades, mejorando la predicción.



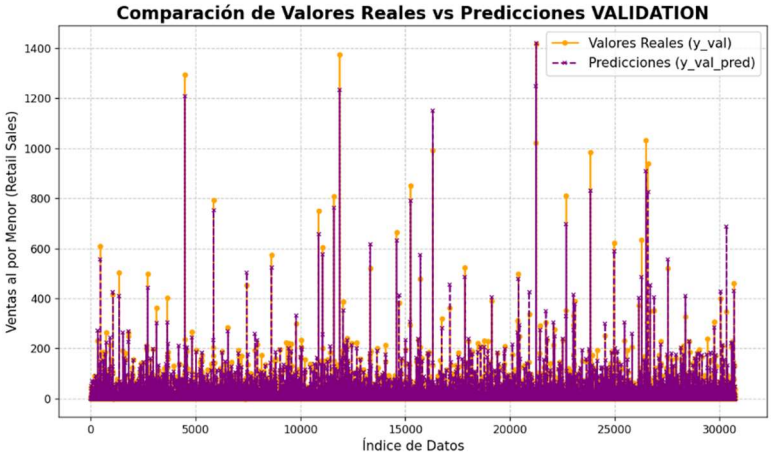
Graf 15. R^2 del Modelo 3 contra variables.

El modelo de RandomForest tiene un poder explicativo de las tendencias de la venta minorista del 96% con un margen de error del 4% en promedio.



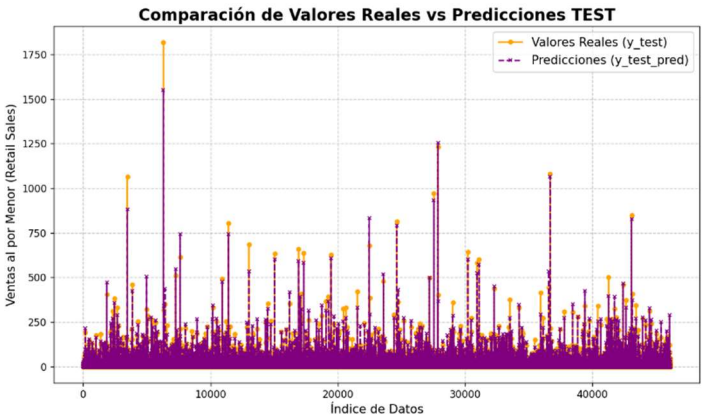
Graf 16. Predicciones Train vs Real Modelo 3.

Como podemos observar, el modelo está prediciendo los valores muy bien, pero esta es la primera prueba.



Graf 17. Predicciones Val vs Real Modelo 3.

La prueba de validación fue el peor en MSE, pero el sesgo fue mucho menor al de Test, y quiere decir nuestro modelo hacer predicciones muy cercanas a lo real, pero que varía mucho los valores reales.



Graf 18. Predicciones Test vs Real Modelo 3.

El modelo fue un poco más certero en esta prueba, concluyendo que este modelo es el óptimo para predecir los RETAIL SALES a futuros años y meses, comparado a los otros dos modelos.

IX. RESULTADOS

Los últimos dos modelos de framework dieron rendimientos óptimos para predecir, a grasso modo las ventas al por menor de las bebidas alcohólicas de las tiendas de EE. UU de la zona de Montgomery.

Cabe destacar que parte del éxito se debió a la dataset, esta cuenta con 300,000 datos para alimentar los algoritmos, esto nos da entender que la esencia de los algoritmos radica en conocer los datos.

Los modelos de Machine Learning dieron buenos resultados al momento de predecir las ventas minorita a futuros, el algoritmo del modelo 2 usando framework, mostró rendimientos sorprendentes, el modelo cuenta con un 95% de asertividad para predecir y se tardó en entrenar 0.07 segundos, y sesgo fue muy cercano a cero, por lo que se dice que predice de la manera correcta.

Un excelente modelo para ampliar el panorama de las ventas futuras, sin embargo, el primer modelo nos ayudó a visualizar con mayor claridad cómo funciona estos algoritmos.

Por último, el algoritmo RandomForest, tuvo un desempeño impecable, al ser un modelo más robusto, ajustó sus hiperparámetros hasta el punto de promediar 28 unidades de error por estimación en promedio, mucho menor al modelo 2, que es un cambio significativo comparado al modelo anterior, pero este modelo al utilizar la técnica bayesian de optimización de hipeparámetros profundamente, esto genera que el modelo tarde demasiado en entrenarse.

En general podemos observar que cada modelo cuenta con un propósito en específico, el primer modelo con fines didácticas, el segundo para una previsualización de las tendencias y el último para un modelo predictivo con mucho robustes para predecir tendencias en dataset más complejas.

X. CONCLUSIONES

Nuestra hipótesis dice:

Podría ser una respuesta el implementar modelos de Machine Learning para impulsar los negocios minoritas, tomando en cuenta su complejidad, su asertividad y su implementación.

El talón de Aquiles de esta tecnología es la información, por ende, es sumamente importante contar con dataset estructuradas para evitar meterle mano a los datos y ensuciarlos, así evitamos predicciones erróneas, lo bueno es que existen herramienta que se pueden usar como base de datos y así abaratan los costos. De otro modo hoy en día basta con solo programar tres líneas de código para que el algoritmo tenga vida, y podamos hacer uso de estas herramientas tecnológicas.

En México en general existe una brecha de conocimiento tecnológico en múltiples sectores, parte de la esencia o capacidades para poder trabajar con estas herramientas se focalizan en ganas de aprender nuevas tecnologías y saber programar.

Esto a su vez nos lleva a indagar, que el conocimiento se traduce a dinero, abordar estas áreas tiene precio y el tiempo también se traduce en dinero.

Aprender estos temas me llevó 5 semana full, a que añadirle que aprendí estos conocimientos en la mejor Universidad del país, y como alumno promedio pago una colegiatura alrededor de 151,272pesos, el ciclo escolar dura 18 semanas, por cada semana estoy pagando 8,404 pesos, y de acuerdo con la zona norte del país una persona en promedio gana 1,562 pesos a la semana (Dainzú Patiño, 2022).

Por los conocimientos adquiridos de Machine Learning estoy pagando 5.3 veces el salario mínimo en una semana.

Sin embargo, esta es una inversión a futuro, al tener los conocimientos y aplicarlos al día a día, obtenemos nuestro retorno, pero hablando de la industria en México es muy distinto hablando a yo como persona a una empresa, estas cuentan con distintos gastos de operación. No obstante, esta herramienta da una ventaja competitiva en la industria para pronosticar futura tendencias, y hacer tomas de decisiones. El problema radica que en México enfocarse a estas áreas de la tecnología es muy caro, por lo que no es rentable invertir en estas áreas para las empresas minoritas, que viven del día al día.

REFERENCIAS

Artificial Intelligence: A Modern Approach, 4th Global ed. (2022). Berkeley.edu. https://aima.cs.berkeley.edu/global-index.html

Elizabeth Meza Rodríguez. (2022, August 18). *3 de cada 4 tienditas desaparecen cuando llega una tienda de conveniencia a la colonia*. El Economista; El Economista. https://www.eleconomista.com.mx/el-empresario/3-de-cada-4-tienditas-desaparecen-cuando-llega-una-tienda-de-conveniencia-a-la-colonia--20211105-0053.html

Ruiz-Healy, E. (2024, February 22). *El impresionante crecimiento del sector minorista mexicano*. El Economista; El Economista. https://www.eleconomista.com.mx/opinion/El-impresionante-crecimiento-del-sector-minorista-mexicano-20240221-0113.html

Urrego, N. (2023, July 10). *Codificación de variables categóricas con Python y R: Técnicas y conceptos clave.* Medium; Medium. https://nicolasurrego.medium.com/codificaci%C3%B3n-de-variables-categ%C3%B3ricas-t%C3%A9cnicas-y-conceptos-clave-bca17b6164b2#:~:text=La%20codificaci%C3%B3n%20de%20frecuencia%20es,cada%20categor%C3%ADa%20en%20los%20datos.

¿A cuánto asciende el salario mínimo en México y para qué debe alcanzarte? By Dainzú Patiño Container: Expansión Year: 2022 URL: https://expansion.mx/economia/2022/07/19/salario-minimo-en-mexico-semana-quincena-mes-ano