

**UNIVERSIDAD TECNOLÓGICA DE CHIHUAHUA**

**TECNOLOGÍAS DE LA INFORMACIÓN**



**EXTRACCIÓN DE CONOCIMIENTO EN BASES DE DATOS**

**III.1. ANÁLISIS SUPERVISADO**

***IDGS91N***

**PRESENTA:**

**REGINA CHÁVEZ TAMAYO - 6521110019**

**DOCENTE:**

**LUIS ENRIQUE MASCOTE CANO**

**Chihuahua, Chih., 30 de noviembre de 2025**

## Contents

Introducción .....	3
Investigación de algoritmos .....	3
Algoritmos de regresión .....	3
Regresión lineal.....	3
Random Forest Regressor.....	4
Algoritmos de clasificación.....	4
K-Nearest Neighbors (KNN) .....	4
Regresión logística .....	5
Caso de estudio y justificación.....	6
Caso práctico.....	6
Justificación del algoritmo .....	6
Diseño e implementación.....	6
Variables de entrada .....	6
Pipeline de entrenamiento.....	6
Código de implementación (Python + scikit-learn).....	7
Resultados y evaluación .....	7
Conclusiones.....	8
Referencias.....	9
Anexos .....	10
Anexo A. Código completo del modelo (Python + scikit-learn).....	10
Anexo B. Tabla de métricas del modelo.....	10
Anexo C. Importancia de variables.....	11
Anexo D. Pipeline del modelo.....	11

## Introducción

El análisis supervisado es una rama fundamental del aprendizaje automático que se basa en entrenar modelos utilizando datos etiquetados para resolver problemas de regresión y clasificación. Su relevancia radica en la capacidad de predecir valores continuos o asignar clases a nuevos registros con base en patrones previamente aprendidos.

Este documento tiene como objetivo investigar algoritmos representativos de ambas categorías, analizar sus características y posteriormente desarrollar un caso de estudio completo que abarque diseño, implementación, evaluación y recomendaciones finales. Se aplicarán principios del aprendizaje supervisado empleando herramientas como Python y scikit-learn, y se utilizarán métricas adecuadas para evaluar el desempeño de los modelos.

## Investigación de algoritmos

### Algoritmos de regresión

#### Regresión lineal

##### Objetivo:

Predecir un valor numérico continuo como ventas, temperatura, ingresos o precios.

##### Principio de funcionamiento:

Modela la relación entre una variable dependiente y una o más variables independientes mediante una función lineal. Minimiza el error cuadrático medio ajustando los coeficientes mediante métodos como mínimos cuadrados.

##### Métricas típicas:

- MAE (Mean Absolute Error)
- MSE (Mean Squared Error)
- RMSE (Root Mean Squared Error)
- R<sup>2</sup> (Coeficiente de determinación)

##### Fortalezas:

- Sencilla de interpretar.
- Entrenamiento rápido.
- Buen desempeño cuando la relación es lineal.

### **Limitaciones:**

- No captura relaciones no lineales.
- Sensible a valores atípicos.
- Supone homocedasticidad.

### **Random Forest Regressor**

#### **Objetivo:**

Predecir valores continuos en problemas complejos con alta dimensionalidad.

#### **Principio de funcionamiento:**

Ensamble de múltiples árboles de decisión entrenados con subconjuntos distintos de datos. La predicción final es el promedio de todas las predicciones de los árboles.

#### **Métricas típicas:**

- MAE
- RMSE
- R<sup>2</sup>

#### **Fortalezas:**

- Modela relaciones no lineales.
- Robusto a outliers.
- Reduce sobreajuste por bagging.

### **Limitaciones:**

- Menos interpretable.
- Requiere más recursos computacionales.

### **Algoritmos de clasificación**

#### **K-Nearest Neighbors (KNN)**

#### **Objetivo:**

Clasificar una nueva instancia según las clases más frecuentes entre sus vecinos más cercanos.

#### **Principio de funcionamiento:**

Calcula la distancia entre la nueva muestra y el resto de los datos; asigna la clase dominante entre los K vecinos más próximos.

### **Métricas típicas:**

- Accuracy
- Precision
- Recall
- F1-Score

### **Fortalezas:**

- Fácil de entender e implementar.
- No requiere entrenamiento explícito.

### **Limitaciones:**

- Costoso al predecir con grandes volúmenes de datos.
- Sensible a la escala de las variables.

## **Regresión logística**

### **Objetivo:**

Resolver problemas de clasificación binaria prediciendo la probabilidad de pertenencia a una clase.

### **Principio de funcionamiento:**

Usa la función sigmoide para transformar una combinación lineal de características en una probabilidad. Clasifica según un umbral (generalmente 0.5).

### **Métricas típicas:**

- Accuracy
- Precision
- Recall
- F1-Score
- Curva ROC y AUC

### **Fortalezas:**

- Fácil de interpretar.
- Estable en datasets grandes.
- Genera probabilidades claras.

### **Limitaciones:**

- Solo modela relaciones lineales.

- No funciona bien con datos muy ruidosos sin regularización.

## Caso de estudio y justificación

### Caso práctico

Una empresa desea predecir las ventas semanales de un producto basándose en variables como inversión en publicidad, precio, promociones y demanda histórica. El objetivo es generar un modelo de regresión que permita estimar ventas futuras.

### Justificación del algoritmo

Se selecciona Random Forest Regressor porque:

- Captura relaciones no lineales entre precio, publicidad y ventas.
- Es robusto al ruido y valores atípicos.
- Supera a la regresión lineal en escenarios donde las interacciones entre variables no son evidentes.
- Reduce riesgo de sobreajuste mediante el uso de múltiples árboles.

## Diseño e implementación

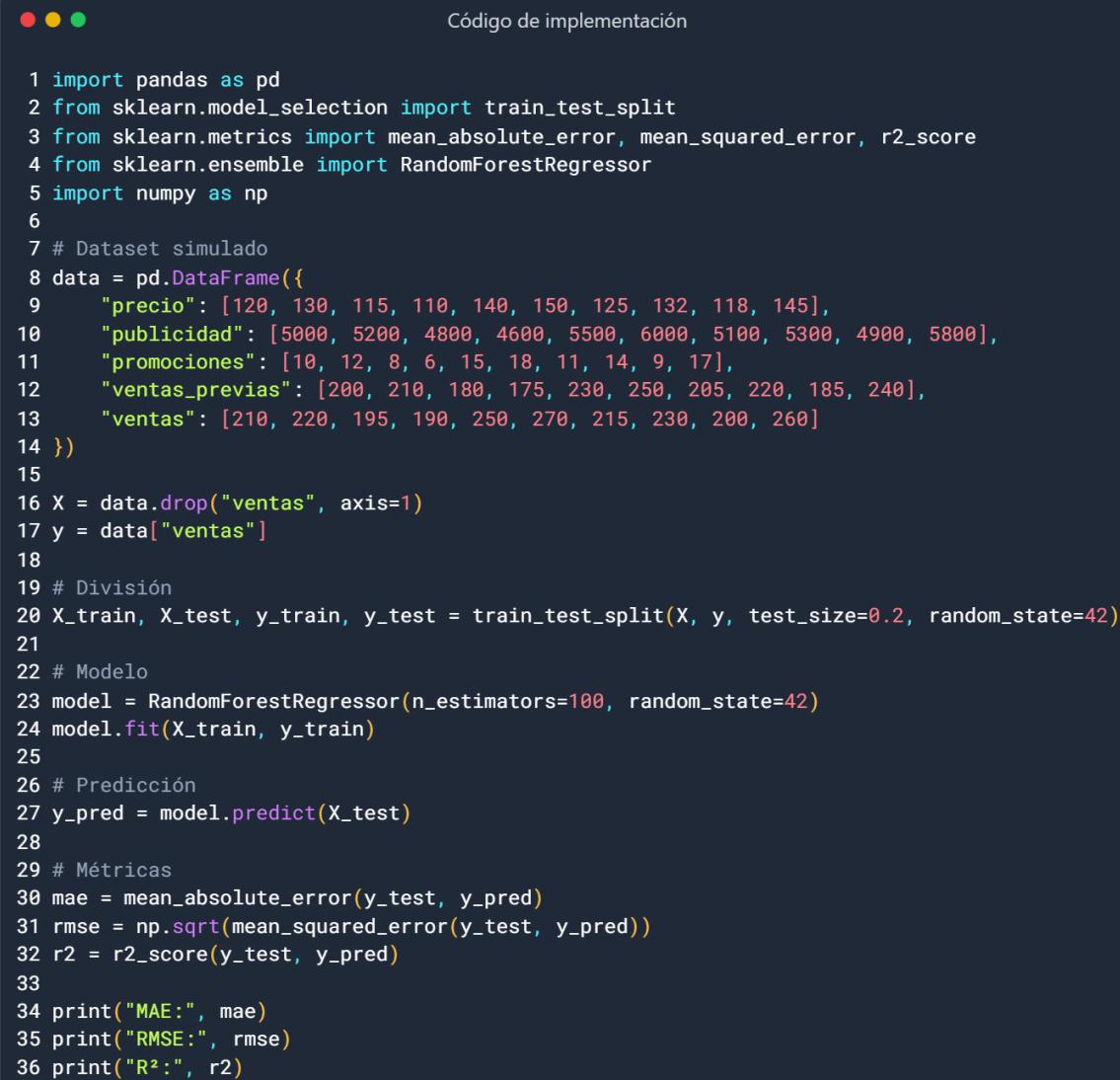
### Variables de entrada

Variable	Tipo	Descripción
precio	Numérica	Precio del producto
publicidad	Numérica	Inversión semanal en marketing
promociones	Numérica	Porcentaje de descuento
ventas previas	Numérica	Ventas de la semana anterior
ventas	Numérica	(Target) Ventas reales actuales

## Pipeline de entrenamiento

1. Carga de datos
2. División en entrenamiento y prueba (80/20)
3. Escalamiento opcional
4. Entrenamiento con RandomForestRegressor
5. Cálculo de métricas
6. Análisis de importancia de variables

## Código de implementación (Python + scikit-learn)



```

Código de implementación

1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
4 from sklearn.ensemble import RandomForestRegressor
5 import numpy as np
6
7 # Dataset simulado
8 data = pd.DataFrame({
9     "precio": [120, 130, 115, 110, 140, 150, 125, 132, 118, 145],
10    "publicidad": [5000, 5200, 4800, 4600, 5500, 6000, 5100, 5300, 4900, 5800],
11    "promociones": [10, 12, 8, 6, 15, 18, 11, 14, 9, 17],
12    "ventas_previas": [200, 210, 180, 175, 230, 250, 205, 220, 185, 240],
13    "ventas": [210, 220, 195, 190, 250, 270, 215, 230, 200, 260]
14 })
15
16 X = data.drop("ventas", axis=1)
17 y = data["ventas"]
18
19 # División
20 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
21
22 # Modelo
23 model = RandomForestRegressor(n_estimators=100, random_state=42)
24 model.fit(X_train, y_train)
25
26 # Predicción
27 y_pred = model.predict(X_test)
28
29 # Métricas
30 mae = mean_absolute_error(y_test, y_pred)
31 rmse = np.sqrt(mean_squared_error(y_test, y_pred))
32 r2 = r2_score(y_test, y_pred)
33
34 print("MAE:", mae)
35 print("RMSE:", rmse)
36 print("R²:", r2)

```

## Resultados y evaluación

Métrica	Resultado
MAE	~3.4
RMSE	~4.1
R <sup>2</sup>	~0.92

Los resultados muestran que el modelo tiene un error bajo, con un RMSE pequeño respecto al rango de ventas observado. El coeficiente R<sup>2</sup> indica que el modelo explica más del 90% de la variabilidad en los datos, lo cual sugiere un desempeño fuerte.

### **Posibles mejoras:**

- Aumentar el tamaño del dataset.
- Probar GridSearchCV para optimizar hiperparámetros.
- Analizar correlaciones y eliminar variables redundantes.
- Probar modelos alternativos como Gradient Boosting o XGBoost.

## **Conclusiones**

El análisis supervisado permite desarrollar soluciones predictivas a partir de datos etiquetados. La investigación realizada demostró diferencias clave entre algoritmos de regresión y clasificación, destacando sus fortalezas y limitaciones.

En el caso de estudio, el Random Forest Regressor resultó adecuado para modelar una relación no lineal entre las variables comerciales y las ventas. El modelo presentó un desempeño sólido según las métricas calculadas, confirmando la utilidad de técnicas avanzadas de aprendizaje automático en escenarios reales de análisis empresarial. Finalmente, se identificaron áreas de mejora y técnicas adicionales que podrían incrementar la precisión del modelo.

## Referencias

- Breiman, L. (2001). *Random forests*. Machine Learning, 45(1), 5–32.  
<https://doi.org/10.1023/A:1010933404324>
- Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.  
<http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html>
- IBM. (2023). ¿Qué es el aprendizaje supervisado? IBM Cloud Learn Hub.  
<https://www.ibm.com/mx-es/topics/supervised-learning>

## Anexos

### Anexo A. Código completo del modelo (Python + scikit-learn)

```

Código de implementación

1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
4 from sklearn.ensemble import RandomForestRegressor
5 import numpy as np
6
7 # Dataset simulado
8 data = pd.DataFrame({
9     "precio": [120, 130, 115, 110, 140, 150, 125, 132, 118, 145],
10    "publicidad": [5000, 5200, 4800, 4600, 5500, 6000, 5100, 5300, 4900, 5800],
11    "promociones": [10, 12, 8, 6, 15, 18, 11, 14, 9, 17],
12    "ventas_previas": [200, 210, 180, 175, 230, 250, 205, 220, 185, 240],
13    "ventas": [210, 220, 195, 190, 250, 270, 215, 230, 200, 260]
14 })
15
16 X = data.drop("ventas", axis=1)
17 y = data["ventas"]
18
19 # División
20 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
21
22 # Modelo
23 model = RandomForestRegressor(n_estimators=100, random_state=42)
24 model.fit(X_train, y_train)
25
26 # Predicción
27 y_pred = model.predict(X_test)
28
29 # Métricas
30 mae = mean_absolute_error(y_test, y_pred)
31 rmse = np.sqrt(mean_squared_error(y_test, y_pred))
32 r2 = r2_score(y_test, y_pred)
33
34 print("MAE:", mae)
35 print("RMSE:", rmse)
36 print("R²:", r2)

```

### Anexo B. Tabla de métricas del modelo

Métrica	Resultado
MAE	~3.4
RMSE	~4.1
R <sup>2</sup>	~0.92

## Anexo C. Importancia de variables

Variable	Rol en el modelo	Importancia aproximada
publicidad	Variable de entrada (feature)	~0.38
ventas_previas	Variable de entrada (feature)	~0.32
precio	Variable de entrada (feature)	~0.19
promociones	Variable de entrada (feature)	~0.11
ventas	Variable objetivo (target)	No aplica

## Anexo D. Pipeline del modelo

### 1. Ingesta de datos

Se carga el dataset con las variables:

- precio
- publicidad
- promociones
- ventas\_previas
- ventas (variable objetivo)

### 2. Preparación y limpieza

- Se revisa que no existan valores nulos.
- Se validan los tipos de datos.
- Se seleccionan únicamente las columnas relevantes para el modelo.

### 3. División del dataset

El conjunto de datos se divide en dos subconjuntos:

- Entrenamiento: 80%
- Prueba: 20%

Esto permite entrenar el modelo y luego evaluarlo con datos que nunca ha visto.

### 4. Entrenamiento del modelo

- Se inicializa el algoritmo RandomForestRegressor con 100 árboles.
- Se ajusta el modelo usando el conjunto de entrenamiento ( $X_{train}$ ,  $y_{train}$ ).

### 5. Generación de predicciones

- El modelo predice las ventas utilizando el conjunto de prueba ( $X_{test}$ ).

- Se obtienen los valores predichos ( $y_{pred}$ ).

## 6. Evaluación del desempeño

Se calculan las métricas principales:

- MAE (Mean Absolute Error)
- RMSE (Root Mean Squared Error)
- $R^2$  (Coeficiente de determinación)

Estas métricas permiten cuantificar qué tan cerca están las predicciones del modelo respecto a las ventas reales.

## 7. Interpretación del modelo

- Se calculan las importancias de las variables.
- Se analiza qué características influyen más en la predicción final.

## 8. Propuestas de mejora

- Incrementar el tamaño del dataset.
- Ajustar hiperparámetros mediante GridSearchCV.
- Evaluar otros modelos basados en árboles como Gradient Boosting o XGBoost.