

# UNIVERSIDAD TECNOLÓGICA DE CHIHUAHUA

## Ingeniería en Desarrollo y Gestión de Software



### Extracción de Conocimientos de Bases de Datos

#### III.1. Análisis Supervisado (50%)

**IDGS91N**

PRESENTAN:

Giselle Cantú Chávez

NOMBRE DEL DOCENTE:

Ing. Luis Enrique Mascote Cano

Chihuahua, Chih., 29 de noviembre de 2025

## Índice

<b>1. Introducción .....</b>	<b>3</b>
<b>2. Investigación de algoritmos .....</b>	<b>3</b>
<b>2.1 Algoritmos de regresión.....</b>	<b>4</b>
<b>2.2 Algoritmos de clasificación .....</b>	<b>6</b>
<b>3. Caso de estudio y justificación.....</b>	<b>8</b>
<b>3.1 Descripción del problema .....</b>	<b>8</b>
<b>3.2 Justificación del algoritmo elegido .....</b>	<b>8</b>
<b>4. Diseño e implementación .....</b>	<b>9</b>
<b>4.1 Diseño del modelo y pipeline .....</b>	<b>9</b>
<b>4.2 Implementación (Python + scikit-learn).....</b>	<b>9</b>
<b>5. Resultados y evaluación.....</b>	<b>12</b>
<b>6. Conclusiones y recomendaciones.....</b>	<b>13</b>
<b>7. Fuentes de apoyo .....</b>	<b>14</b>

## 1. Introducción

En esta evidencia me enfoco en la parte de aprendizaje supervisado de la materia Extracción de Conocimientos en Bases de Datos, específicamente en los modelos de regresión y clasificación. Ambos tipos de algoritmos comparten la misma idea general: aprenden a partir de ejemplos etiquetados, es decir, de pares entrada–salida donde ya conocemos el valor correcto y el modelo busca aproximar lo mejor posible.

En los problemas de regresión, la salida es un valor numérico continuo, como ventas mensuales, temperatura o precio de una casa. En los problemas de clasificación, la salida es una etiqueta o categoría, por ejemplo “compra / no compra”, “baja / no baja”, “aprobado / reprobado”.

El objetivo de este trabajo es:

- Investigar al menos dos algoritmos de regresión y dos de clasificación, describiendo su objetivo, cómo funcionan, qué métricas se usan para evaluarlos y cuáles son sus ventajas y limitaciones.
- Proponer un caso de estudio cercano a un escenario real (clasificación de clientes según su probabilidad de abandonar un servicio).
- Diseñar un pipeline de modelado, implementar un ejemplo en Python con scikit-learn y revisar el comportamiento del modelo mediante métricas como accuracy, precision, recall, F1-score, MAE o RMSE, según corresponda.

Con esto busco no solo cumplir la evidencia, sino dejar una base clara para seguir trabajando con modelos supervisados en otros contextos de minería de datos.

## 2. Investigación de algoritmos

En esta sección reviso cuatro algoritmos muy utilizados en la práctica:

- Regresión (salida numérica):

- Regresión lineal
- Random Forest Regressor
- Clasificación (salida categórica):
  - Regresión logística
  - Random Forest Classifier

## 2.1 Algoritmos de regresión

### 2.1.1 Regresión lineal

#### Qué resuelve.

La regresión lineal se usa cuando quiero predecir un valor continuo en función de varias características. Un ejemplo típico es estimar el monto de ventas de un mes a partir de variables como número de clientes activos, inversión en publicidad o precios promedio.

#### Principio de funcionamiento.

La idea es ajustar una recta (o un hiperplano en más dimensiones) que minimiza el error entre las predicciones del modelo y los valores reales. El modelo supone que la variable objetivo es una combinación lineal de las características:

$$\hat{y} = w_0 + w_1x_1 + \cdots + w_px_p$$

#### Métricas típicas.

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

#### Fortalezas.

- Es fácil de interpretar.
- Es rápida de entrenar incluso con muchos datos.

### **Limitaciones.**

- Supone una relación aproximadamente lineal.
- Es sensible a outliers.

### **2.1.2 Random Forest Regressor**

#### **Qué resuelve.**

Random Forest Regressor permite modelar relaciones no lineales y efectos complejos entre variables cuando quiero predecir un valor numérico, como demanda, consumo de energía o satisfacción.

#### **Principio de funcionamiento.**

Es un ensamble de muchos árboles de decisión entrenados sobre muestras ligeramente distintas del dataset. Cada árbol da una predicción y el modelo final toma el promedio.

#### **Métricas típicas.**

MAE, RMSE y R<sup>2</sup>.

#### **Fortalezas.**

- Captura relaciones no lineales sin que yo las defina.
- Resistente al ruido.
- Acepta muchas características.

#### **Limitaciones.**

- Difícil de interpretar.
- Puede requerir más recursos computacionales.

## 2.2 Algoritmos de clasificación

### 2.2.1 Regresión logística

#### Qué resuelve.

Ideal cuando la salida es binaria, como “abandona / no abandona”, “fraude / normal”.

#### Principio de funcionamiento.

Calcula una combinación lineal de las variables y luego aplica una función logística (sigmoide) que devuelve una probabilidad entre 0 y 1.

#### Métricas típicas.

- Accuracy
- Precision
- Recall
- F1-score
- ROC-AUC

#### Fortalezas.

- Simple, rápida y estable.
- Interpretación clara de coeficientes.

#### Limitaciones.

- Se queda corta en problemas altamente no lineales.
- Sensible a multicolinealidad y escalas no normalizadas.

## 2.2.2 Random Forest Classifier

### Qué resuelve.

Clasifica observaciones en categorías: segmentación de clientes, clasificación de transacciones, detección de abandono.

### Principio de funcionamiento.

Cada árbol “vota” por una clase y el bosque elige la clase más votada.

### Métricas típicas.

Accuracy, precision, recall, F1-score, ROC-AUC.

### Fortalezas.

- Muy buen desempeño general.
- Maneja datos mixtos.
- Resistente al ruido.

### Limitaciones.

- Poco interpretable.
- Tamaño del modelo grande.

### 3. Caso de estudio y justificación

#### 3.1 Descripción del problema

Planteo un escenario donde una empresa quiere predecir si un cliente dará de baja su suscripción en los próximos meses. Esta variable se llama churn (1 = se va; 0 = permanece).

Variables de entrada ejemplo:

- edad
- antiguedad\_meses
- monto\_mensual
- num\_productos
- reclamaciones\_ultimo\_año
- pago\_atrasado

Este tipo de problema se usa mucho en banca, telefonía, seguros y plataformas de suscripción.

#### 3.2 Justificación del algoritmo elegido

Elijo Random Forest Classifier como modelo principal porque:

- Captura relaciones no lineales entre variables.
- Suele superar a la regresión logística en precisión en datos tabulares.
- Permite revisar la importancia de variables.

La regresión logística la uso como baseline porque es interpretativa y sirve para comparar el desempeño del bosque.

## 4. Diseño e implementación

### 4.1 Diseño del modelo y pipeline

El flujo que sigo es:

1. Cargar y explorar datos.
2. Seleccionar características numéricas y categóricas.
3. Preparar datos (entrenamiento/prueba, codificación, estandarización).
4. Entrenar modelos: regresión logística y Random Forest.
5. Evaluar con accuracy, precision, recall, F1-score.
6. Ajustar hiperparámetros si es necesario.

### 4.2 Implementación (Python + scikit-learn)

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import Pipeline

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear_model import LogisticRegression

# 1. Carga de datos
```

```
data = pd.read_csv("clientes_churn.csv")
```

## # 2. Selección de variables

```
features = ["edad", "antiguedad_meses", "monto_mensual",  
"num_productos", "reclamaciones_ultimo_año"]
```

```
target = "churn"
```

```
X = data[features]
```

```
y = data[target]
```

## # 3. División entrenamiento / prueba

```
X_train, X_test, y_train, y_test = train_test_split(
```

```
X, y,
```

```
test_size=0.2,
```

```
stratify=y,
```

```
random_state=42
```

```
)
```

## # 4. Pipeline regression logística (baseline)

```
log_reg_pipeline = Pipeline([
    ("scaler", StandardScaler()),
    ("model", LogisticRegression(max_iter=1000))
])
```

```
log_reg_pipeline.fit(X_train, y_train)
```

```
y_pred_log = log_reg_pipeline.predict(X_test)
```

## # 5. Pipeline Random Forest

```
rf_pipeline = Pipeline([
    ("scaler", StandardScaler()),
```

```
    ("model", RandomForestClassifier(
```

```
        n_estimators=200,
```

```
        random_state=42
```

```
))
```

```
])
```

```
rf_pipeline.fit(X_train, y_train)
```

```
y_pred_rf = rf_pipeline.predict(X_test)
```

## # 6. Métricas

```

print("== Regresión logística ==")

print("Accuracy:", accuracy_score(y_test, y_pred_log))

print(classification_report(y_test, y_pred_log))

print("== Random Forest Classifier ==")

print("Accuracy:", accuracy_score(y_test, y_pred_rf))

print(classification_report(y_test, y_pred_rf))

print("Matriz de confusión (Random Forest):")

print(confusion_matrix(y_test, y_pred_rf))

```

## 5. Resultados y evaluación

Ejemplo de resultados:

Modelo	Accuracy	Precision (churn)	Recall (churn)	F1 (churn)
Regresión logística	0.82	0.74	0.68	0.71
Random Forest Classifier	0.88	0.81	0.80	0.80

Aquí, Random Forest supera a la regresión logística, especialmente en F1-score y recall. Esto lo hace más útil para detectar clientes que realmente se darán de baja.

Interpretación:

- Accuracy mide aciertos totales.
- Precision dice qué tan confiable es cuando predice churn.
- Recall indica cuántos churn reales detecta.
- F1 balancea precision y recall.

## 6. Conclusiones y recomendaciones

En esta evidencia revisé cuatro algoritmos de aprendizaje supervisado. Confirmé que:

- La regresión lineal funciona bien cuando la relación es más o menos lineal y se requiere interpretabilidad.
- Random Forest suele tener mejor rendimiento en datos complejos.
- La regresión logística es un baseline útil, pero modelos basados en árboles pueden superarla en muchos contextos reales.

Recomiendo:

- Usar validación cruzada para ajustar mejor los modelos.
- Manejar desbalanceo de clases si existe.
- Probar modelos adicionales como Gradient Boosting o XGBoost.

## 7. Fuentes de apoyo

Álava, J. N. M., Macías Bermeo, L. A., Morales-Carrillo, J., y Cedeño-Valarezo, L. (2024). **Modelos de aprendizaje automático: aplicación y eficiencia.** Revista Científica de Informática ENCRYPTAR, 7(14), 87–114. <https://doi.org/10.56124/encryptar.v7i14.005>

Géron, A. (2019). **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow** (2.a ed.). O'Reilly Media.

López Barriga, M. G., Pozo Valdiviezo, A. E., Pérez Londo, N. A., y Ramos Araujo, C. E. (2024). **Algoritmos de aprendizaje automático en la predicción del rendimiento académico en la educación superior.** Pol. Con., 9(7), 2683–2701.

Morales Hernández, M. Á., González Camacho, J. M., Robles Vásquez, H., Del Valle Paniagua, D. H., y Durán Moreno, J. R. (2022). **Algoritmos de aprendizaje automático para la predicción del logro académico.** Revista Iberoamericana para la Investigación y el Desarrollo Educativo, 12(24), e341. <https://doi.org/10.23913/ride.v12i24.1180>

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., etc. (2011). **Scikit-learn: Machine learning in Python.** Journal of Machine Learning Research, 12, 2825–2830.

Scikit-learn developers. (2025). **1. Supervised learning.** [https://scikit-learn.org/stable/supervised\\_learning.html](https://scikit-learn.org/stable/supervised_learning.html)

Scikit-learn developers. (2025). **3.4. Metrics and scoring: quantifying the quality of predictions.** [https://scikit-learn.org/stable/modules/model\\_evaluation.html](https://scikit-learn.org/stable/modules/model_evaluation.html)

Google Developers. (s. f.). **Classification: Accuracy, precision, recall, and related metrics.** <https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall>

Mendoza Álava, J. N., Macías Bermeo, L. A., Morales-Carrillo, J., y Cedeño-Valarezo, L. (2024). **Modelos de aprendizaje automático: aplicación y eficiencia.** <https://publicacionescd.uleam.edu.ec/index.php/encryptar/article/view/1009>

Noviana, F. R., y otros. (2024). **Performance comparison Random Forest and Logistic Regression algorithms.** International Journal of Computer Applications, 186(16), 1–7.