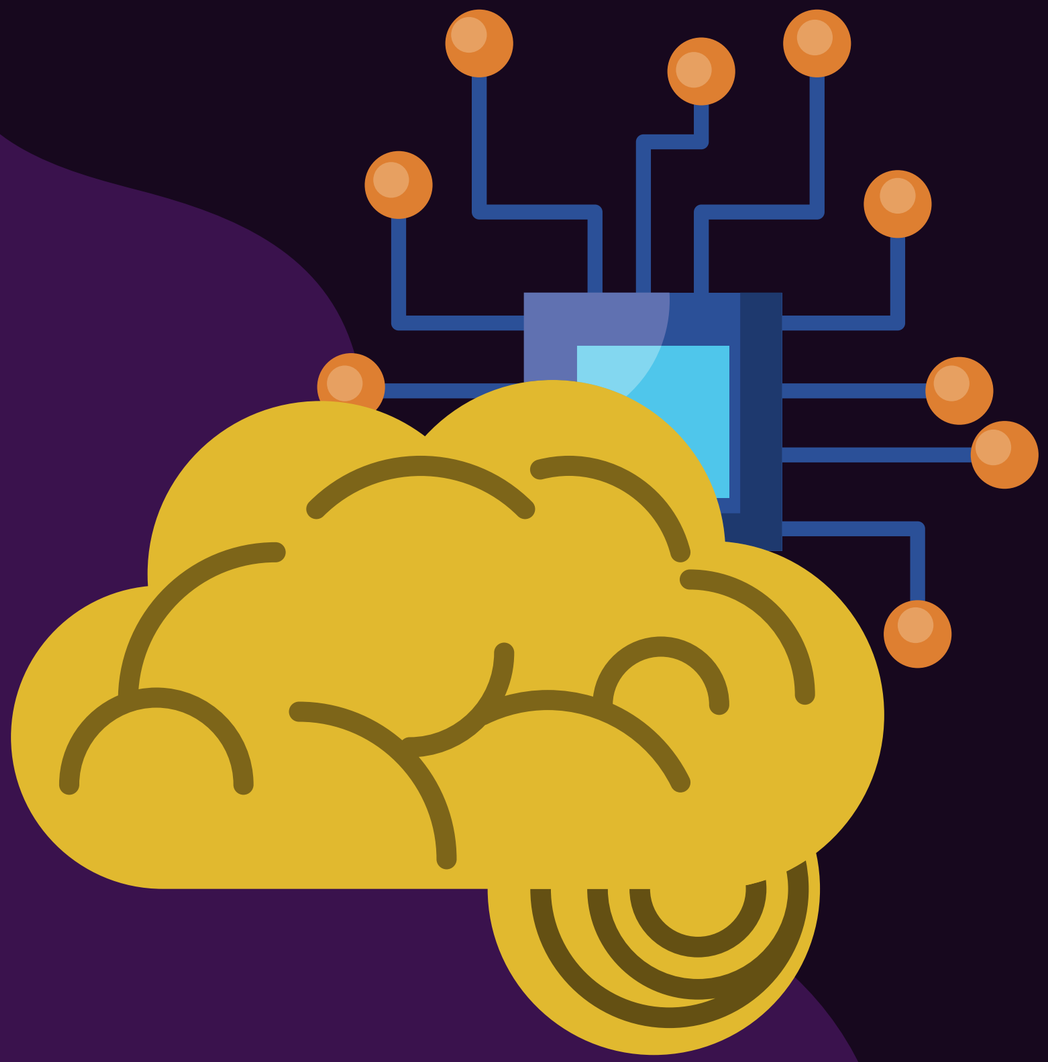


DIGIT RECOGNITION

En busqueda del mejor modelo

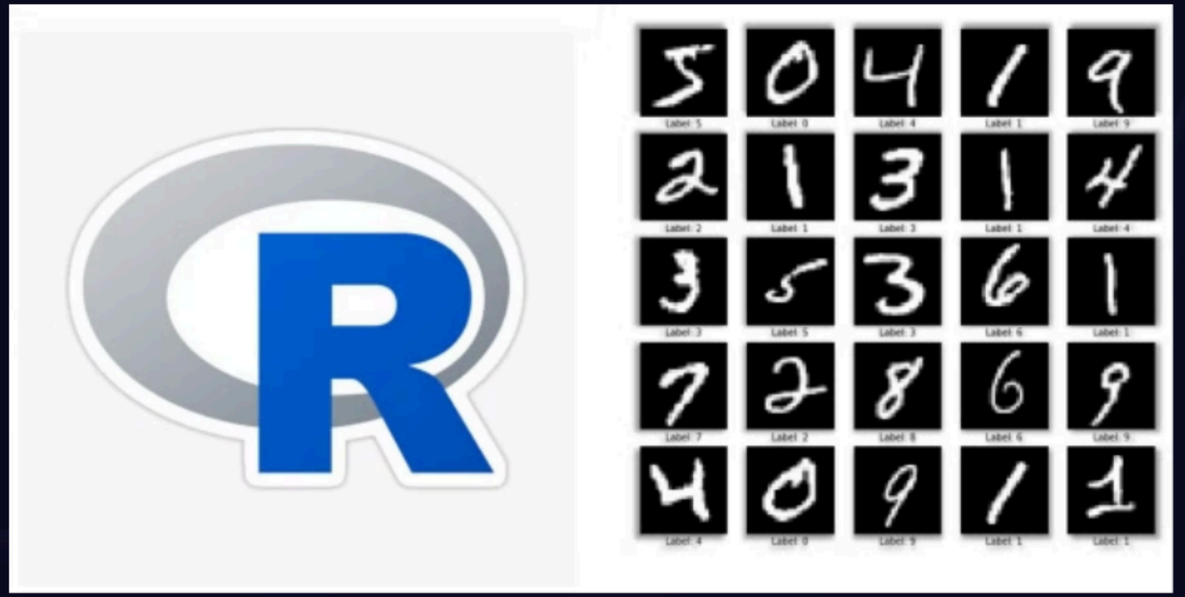
Realizado por : Rocío Guzmán ,Alberto Sánchez ,
Adrián Pradas y Rodrigo Carreira

Introducción al Reconocimiento de Patrones

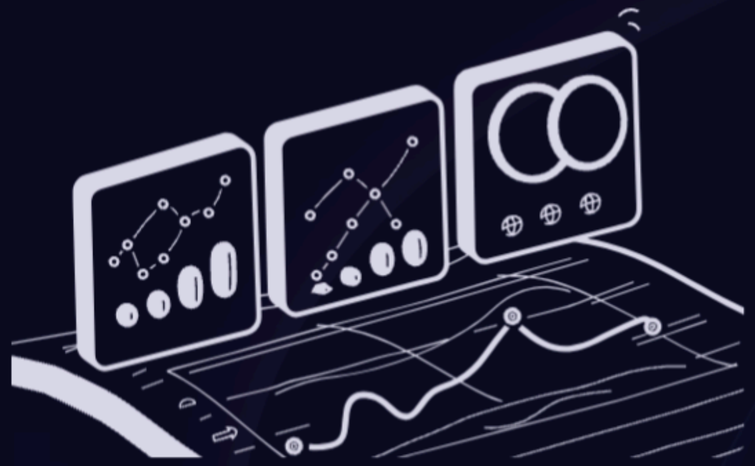
El reconocimiento de patrones es un área fundamental en la inteligencia artificial y el aprendizaje automático.

Digit Recognition es un desafío técnico que implica variabilidad en la escritura humana y ruido en los datos.

Este tipo de tarea tiene aplicaciones en campos como la clasificación de documentos, el reconocimiento óptico de caracteres (OCR) y el análisis de datos.



Análisis Del Dataset



Dimensiones Del Dataset

El archivo train.csv tiene 42,000 filas y 785 columnas.

Estructura Del Dataset

El dataset es consistente con la estructura del MNIST, utilizado en tareas de clasificación y reconocimiento de dígitos.

Subconjuntos Utilizados

Se utiliza un subconjunto de 5,000 filas del archivo train.csv para experimentos preliminares.

Preprocesamiento de Datos

Normalización

Los valores de los píxeles se normalizan para garantizar que todos los datos estén en el mismo rango.

Carga y División

Los archivos se cargan y se divide el conjunto de entrenamiento en subconjuntos.

Submuestreo

Se selecciona un subconjunto de 5,000 filas para experimentos preliminares.

Verificación de Etiquetas

Se calcula la distribución porcentual de las etiquetas en el conjunto de entrenamiento.

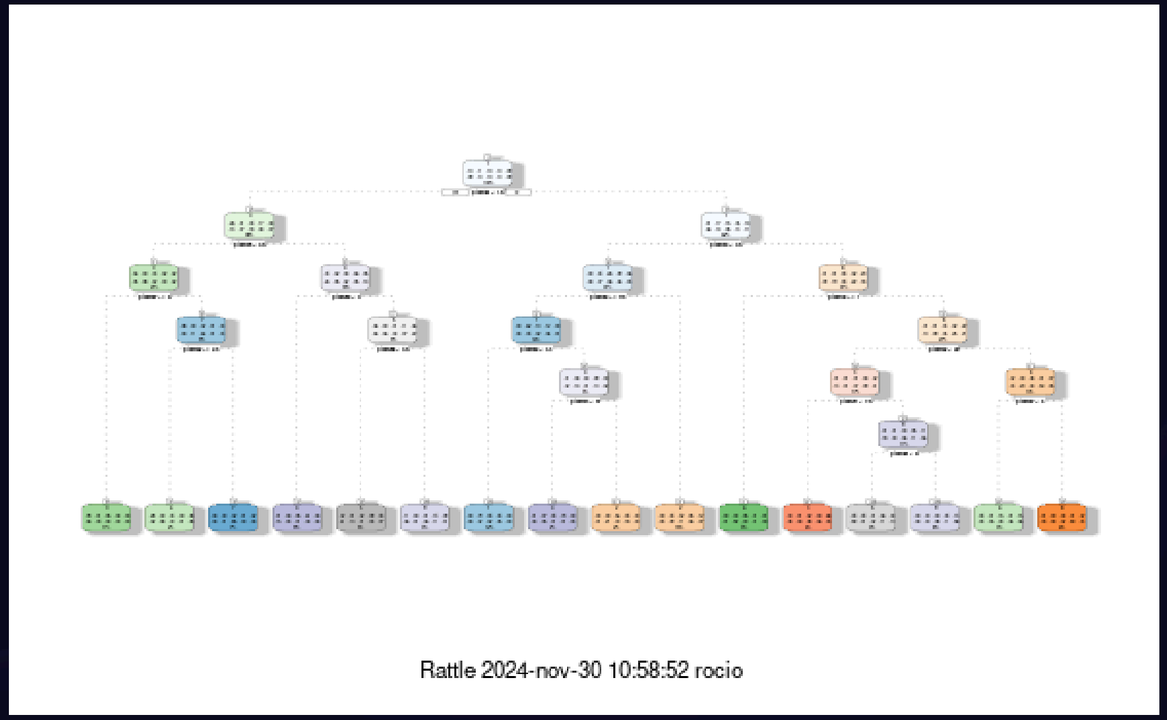
Conversión de Etiquetas

Se transforma la columna label en un tipo de dato factor.

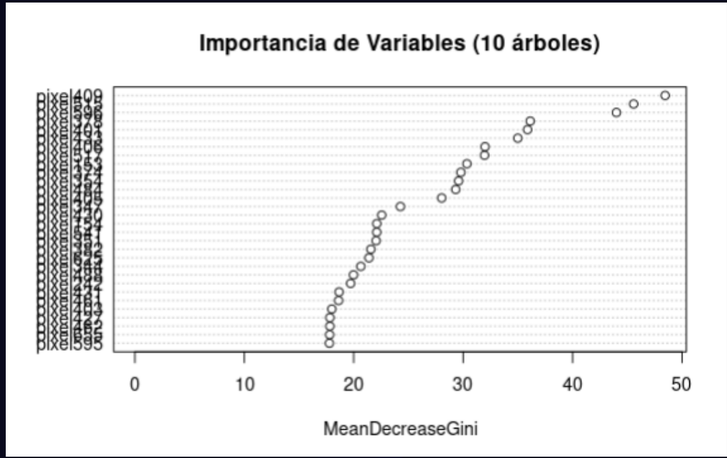
Training with Rpart

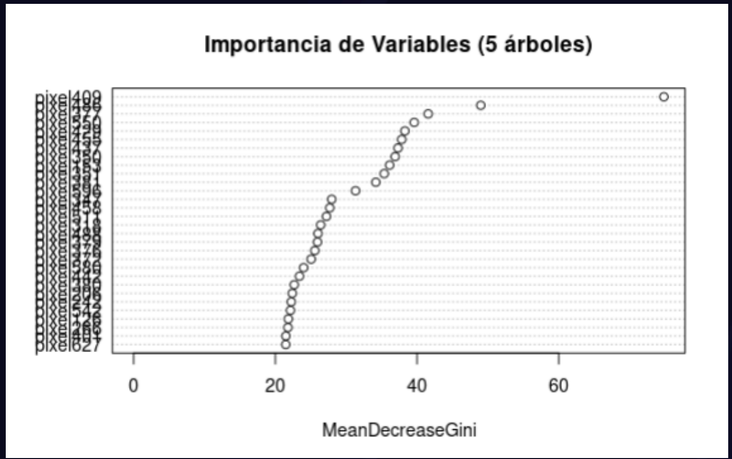
El paquete rpart de R se utiliza para construir un árbol de decisión.
Posteriormente se calculaba el modelo a través de: arbol <- rpart(label ~ ., data = dtrain, method = "class").
Evaluamos la matriz de confusión y el rendimiento del modelo para concretar que:

Precisión del árbol de decisión: 0.615.



Random Forest





1. **Entrenamiento**

   Se entrenan dos modelos de Random Forest con diferentes números de árboles (10 y 5).

2. **Predicción**

   Se realizan predicciones sobre el conjunto de prueba utilizando el modelo entrenado.

3. **Evaluación**

   Se calcula la precisión del modelo mediante la matriz de confusión.

SVM Model Training

1

Preparación de Datos

Se eliminan las columnas constantes.

2

Entrenamiento del Modelo

Se utiliza el kernel _polydot_ de grado 3.

3

Evaluación del Rendimiento

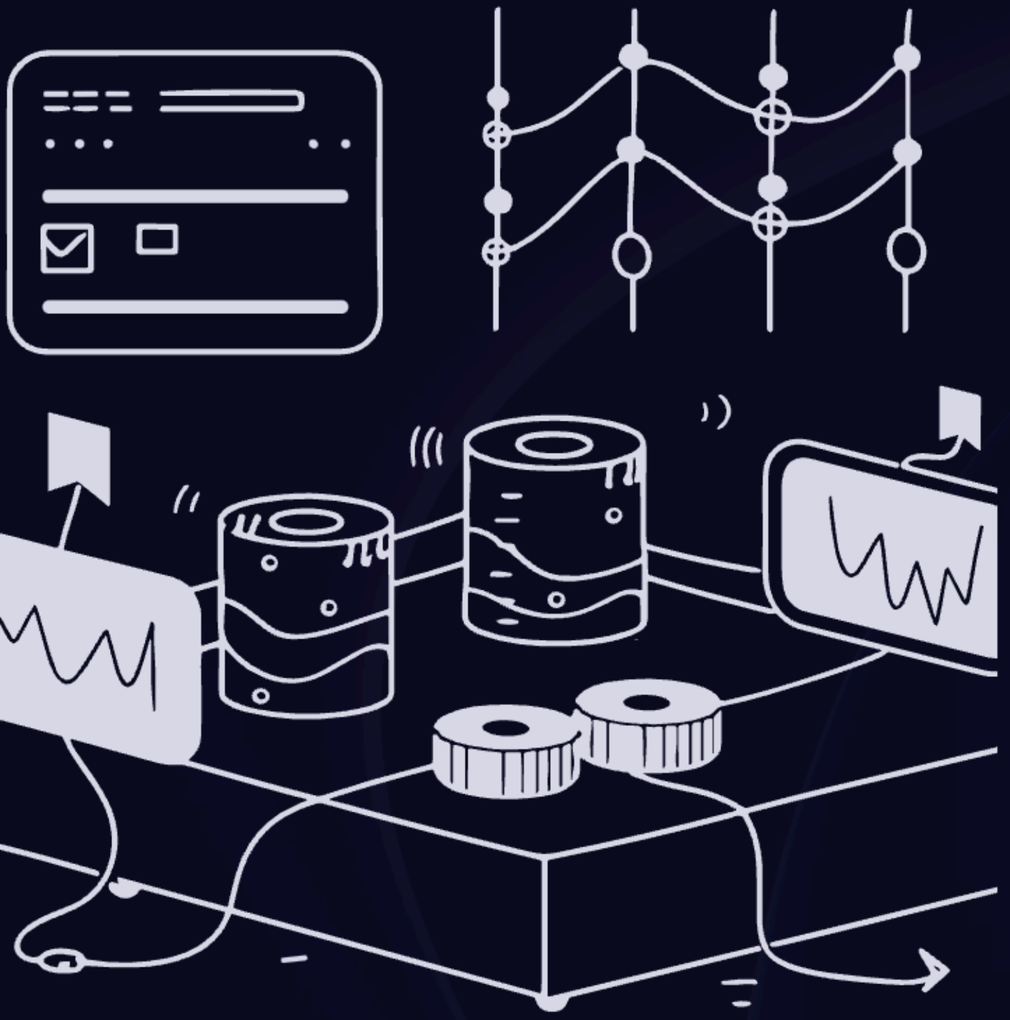
Se calcula la precisión del modelo.

4

Análisis de Resultados

Se analiza la matriz de confusión.

El modelo SVM logró una precisión del 94.1%, clasificando correctamente la mayoría de las instancias.



Bagging y K-Vecinos Más Cercanos

1

Bagging

El Bagging (Bootstrap Aggregating) combina múltiples modelos base para mejorar la precisión y reducir la varianza del modelo final.

2

KNN

El algoritmo K-Vecinos Más Cercanos (KNN) clasifica un punto de prueba según la mayoría de las etiquetas de los k vecinos más cercanos.

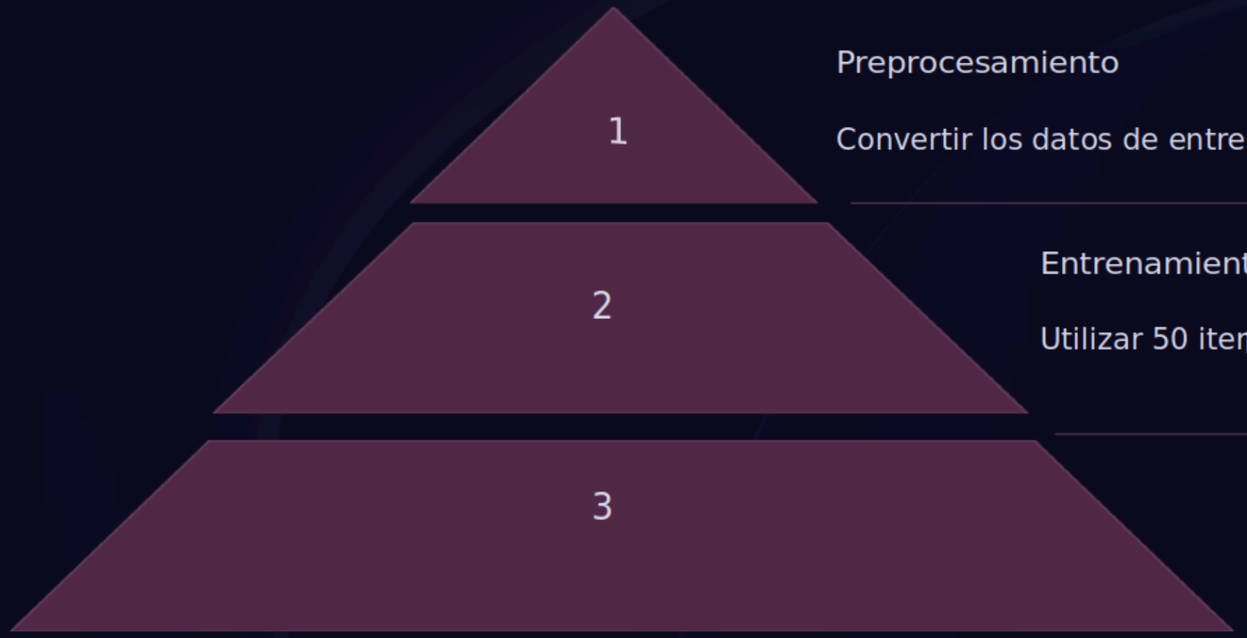
3

Resultados

La precisión del modelo de Bagging fue de 0.916, mientras que el modelo KNN logró una precisión de 0.912.

XGBoost Model and Results

El modelo XGBoost (Extreme Gradient Boosting) es una implementación eficiente y escalable de los algoritmos de gradient boosting.



Preprocesamiento

Convertir los datos de entrenamiento y prueba a formato de matriz.

Entrenamiento

Utilizar 50 iteraciones con la tarea de clasificación multiclase.

Realizar predicciones sobre el conjunto de prueba y calcular la matriz de confusión.

Evaluación

La precisión obtenida por el modelo XGBoost fue de 0.934, lo que indica que el modelo ha clasificado correctamente el 93.4 % de los dígitos en el conjunto de prueba.

Gracias