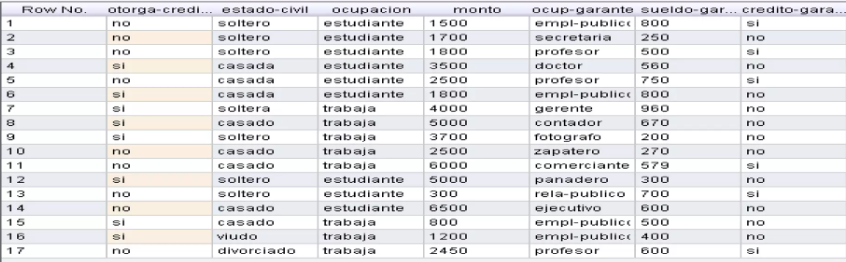
**FACULTAD DE INGENIERÍA**

**Práctica Nro. 02**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Asignatura** | **Inteligencia Artificial** | | | **NOTA** |
| **Sección** | **A61A** | | |
| **Docente** | **Ing. Juancarlos Santana Huamán** | | |  |
| **Estudiante** | Paucar Porras Miguel Angel | | |
| **Observaciones** |  | **Duración** | **Fecha** |
|  | **15/09/2025** |

1. Implemente el algoritmo ID3 y J48 de los siguientes datos:



import pandas as pd

from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier, export\_text

from sklearn.tree import DecisionTreeClassifier, export\_text

# Datos

data = {

    'estado-civil': ['soltero', 'soltero', 'soltero', 'casada', 'casada', 'casada', 'soltera', 'casado', 'soltero', 'casado', 'casado', 'soltero', 'soltero', 'casado', 'casado', 'viudo', 'divorciado'],

    'ocupacion': ['estudiante', 'estudiante', 'estudiante', 'estudiante', 'estudiante', 'estudiante', 'trabaja', 'trabaja', 'trabaja', 'trabaja', 'trabaja', 'estudiante', 'estudiante', 'estudiante', 'trabaja', 'trabaja', 'trabaja'],

    'monto': [1500, 1700, 1800, 3500, 2500, 1800, 4000, 5000, 3700, 2500, 6000, 5000, 300, 6500, 800, 1200, 2450],

    'ocup-garante': ['empl-publico', 'secretaria', 'profesor', 'doctor', 'empl-publico', 'gerente', 'contador', 'fotografo', 'zapatero', 'comerciante', 'panadero', 'rela-publico', 'ejecutivo', 'empl-publico', 'empl-publico', 'profesor', 'profesor'],

    'sueldo-gar': [800, 250, 500, 560, 750, 800, 960, 670, 200, 270, 579, 300, 700, 600, 500, 400, 600],

    'otorga-credito': ['no', 'no', 'no', 'si', 'no', 'si', 'si', 'si', 'si', 'no', 'no', 'si', 'no', 'no', 'si', 'si', 'no']

}

# Crear DataFrame

df = pd.DataFrame(data)

# Codificar variables categóricas

le = LabelEncoder()

for col in ['estado-civil', 'ocupacion', 'ocup-garante', 'otorga-credito']:

    df[col] = le.fit\_transform(df[col])

# Separar características y variable objetivo

target = 'otorga-credito'

X = df.drop(columns=[target])

y = df[target]

# Modelo ID3 con criterio Entropía

id3\_model = DecisionTreeClassifier(criterion='entropy')

id3\_model.fit(X, y)

# Extraer reglas del árbol

rules = export\_text(id3\_model, feature\_names=list(X.columns))

print(rules)

# Entrenar modelo J48 aproximado con poda mediante max\_depth para limitar complejidad

j48\_model = DecisionTreeClassifier(criterion='entropy', max\_depth=4)

j48\_model.fit(X, y)

# Extraer reglas del árbol J48 aproximado

rules\_j48 = export\_text(j48\_model, feature\_names=list(X.columns))

print(rules\_j48)

Tabla

El contenido generado por IA puede ser incorrecto.

import pandas as pd

from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier, export\_text

# Datos

data\_image = {

    'Edad': ['joven', 'joven', 'joven', 'joven', 'joven', 'adulto', 'adulto', 'adulto', 'adulto', 'adulto', 'adulto', 'adulto', 'adulto', 'joven', 'joven', 'adulto', 'adulto', 'joven', 'joven', 'adulto', 'joven'],

    'Profesional': ['si', 'si', 'no', 'si', 'no', 'si', 'no', 'si', 'no', 'si', 'no', 'si', 'no', 'si', 'si', 'no', 'no', 'no', 'no', 'si', 'si'],

    'Ingresos': ['bajos', 'altos', 'altos', 'bajos', 'medios', 'altos', 'altos', 'altos', 'medios', 'bajos', 'medios', 'medios', 'altos', 'altos', 'medios', 'medios', 'bajos', 'medios', 'bajos', 'medios', 'medios'],

    'Sexo': ['hombre', 'hombre', 'mujer', 'mujer', 'mujer', 'hombre', 'mujer', 'mujer', 'mujer', 'mujer', 'mujer', 'hombre', 'hombre', 'mujer', 'hombre', 'hombre', 'hombre', 'hombre', 'mujer', 'mujer', 'mujer'],

    'Interesado': ['si', 'si', 'no', 'si', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'si', 'si', 'si', 'no', 'no', 'no', 'no', 'no', 'si']

}

# Crear DataFrame

df\_image = pd.DataFrame(data\_image)

# Codificar variables categóricas

le\_image = LabelEncoder()

for col in ['Edad', 'Profesional', 'Ingresos', 'Sexo', 'Interesado']:

    df\_image[col] = le\_image.fit\_transform(df\_image[col])

# Separar características y variable objetivo

X\_image = df\_image.drop(columns=['Interesado'])

y\_image = df\_image['Interesado']

# Implementar ID3 con criterio 'entropy'

id3\_model\_image = DecisionTreeClassifier(criterion='entropy')

id3\_model\_image.fit(X\_image, y\_image)

# Mostrar reglas del modelo ID3

rules\_id3\_image = export\_text(id3\_model\_image, feature\_names=list(X\_image.columns))

print("Reglas ID3:\n", rules\_id3\_image)

# Implementar J48 aproximado usando poda (max\_depth)

j48\_model\_image = DecisionTreeClassifier(criterion='entropy', max\_depth=4)

j48\_model\_image.fit(X\_image, y\_image)

# Mostrar reglas del modelo J48 aproximado

rules\_j48\_image = export\_text(j48\_model\_image, feature\_names=list(X\_image.columns))

print("Reglas J48 aproximado:\n", rules\_j48\_image)