Árbol de decisiones ID3

Inteligencia Artificial

Sofía Albert Tassier 181767
José Francisco Garduño Suchil 181536
Natalia Hernández Cornejo 183420
Jorge Edgar Rodríguez Ortiz Loyola 181334

Predicción de derrame cerebral

Según la OMS, el derrame cerebral es la segunda causa de muerte a nivel global.

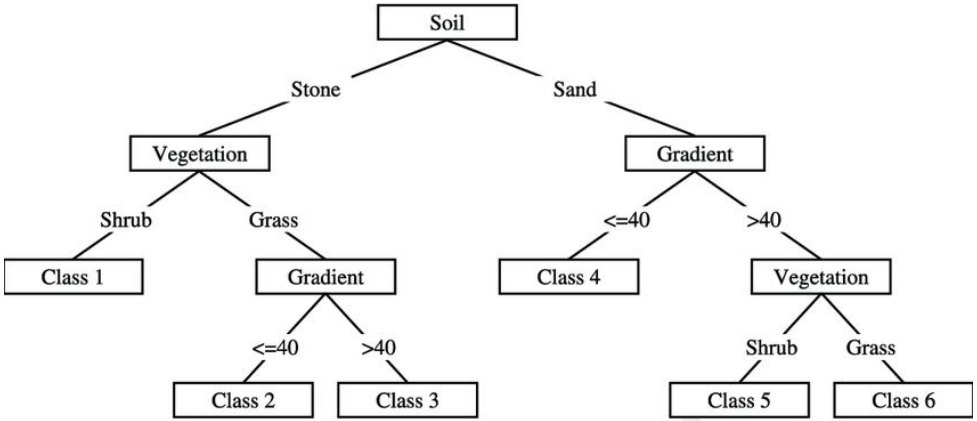
Parámetros

- Género
- Edad
- Algunas enfermedades (hipertensión, cardiopatía)
- Trabajo
- Fumador
- Zona (rural o urbana)
- Nivel de glucosa
- Índice de masa corporal
- Estado civil

Planteamiento

Un árbol de decisión es un algoritmo de clasificación que se usa para predecir el resultado de un evento con atributos dados.

Por ejemplo, ¿puedo jugar a la pelota cuando el panorama es soleado, la temperatura es alta, la humedad es alta y el viento es débil?

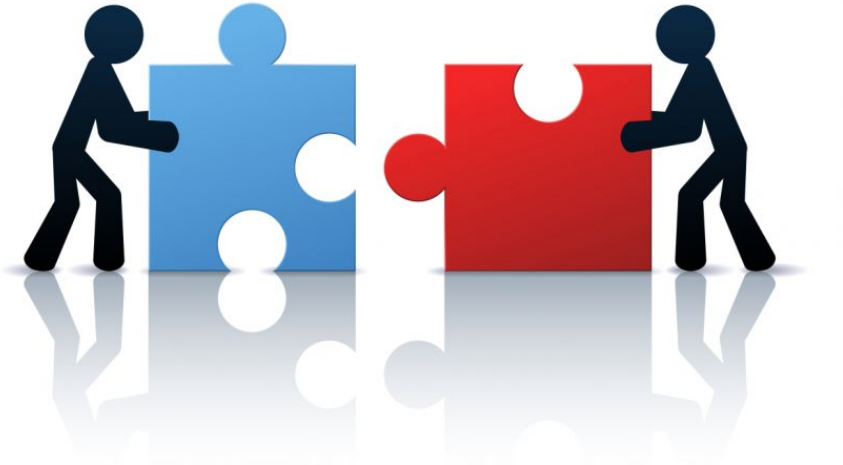


Solución

La **raíz** del árbol corresponde al descriptor con menor entropía

Conjunto de entrenamiento donde los ejemplos tienen valores de atributos y el valor del predicado meta.

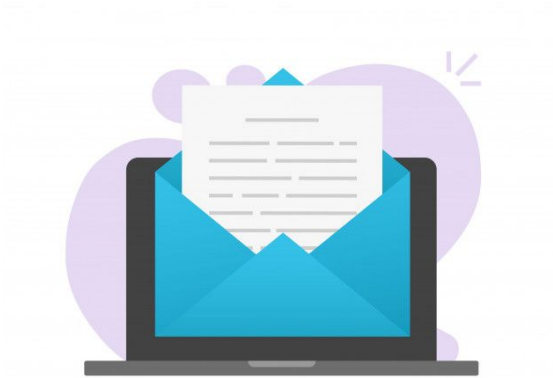
Implementación en C#



Lectura de archivos

El **conjunto de entrenamiento** se lee como un archivo **.csv,** usando **StreamReader**

El usuario escribe el *path* del archivo, lo que permite usar diferentes conjuntos de entrenamiento.

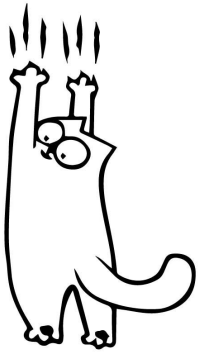


```
try
{
    using (var reader = new StreamReader(File.OpenRead(filePath)))
    {
        while (!reader.EndOfStream)
        {
            var line = reader.ReadLine();
            var values = line.Substring(0, line.Length - 1).Split(';');

            foreach (var item in values)
            {
                if (string.IsNullOrEmpty(item) || string.IsNullOrWhiteSpace(item))
                {
                    throw new Exception("No puede estar vacío el valor");
                }

                if (rows == 0)
                {
                    data.Columns.Add(item);
                }
            }
```

```
            if (rows > 0)
            {
                data.Rows.Add(values);
            }

            rows++;

            if (values.Length != data.Columns.Count)
            {
                throw new Exception("La fila no es igual de larga que la fila de titulos");
            }
        }
    }

catch (Exception ex)
{
    DisplayErrorMessage(ex.Message);
    data = null;
}
```

Calcular entropía

```
private static List<int[,]> GetAmountOfEdgesAndTotalPositivResults(DataTable data, int indexOfColumnToCheck)
{
    var foundValues = new List<int[,]>();
    var knownValues = CountKnownValues(data, indexOfColumnToCheck);

    foreach (var item in knownValues)
    {
        var amount = 0;
        var positiveAmount = 0;

        for (var i = 0; i < data.Rows.Count; i++)
        {
            if (data.Rows[i][indexOfColumnToCheck].ToString().Equals(item))
            {
                amount++;

                // Cuenta los casos positivos y los agrega a la suma para después hacer los calculos
                if (data.Rows[i][data.Columns.Count - 1].ToString().Equals(data.Rows[0][data.Columns.Count - 1]))
                {
                    positiveAmount++;
                }
            }
        }

        int[,] array = { { amount, positiveAmount } };
        foundValues.Add(array);
    }

    return foundValues;
}
```



```
private static double calcularEntropía(DataTable data)
{
    var totalRows = data.Rows.Count;
    var amountForDifferentValue = GetAmountOfEdgesAndTotalPositivResults(data, data.Columns.Count - 1);

    var stepsForCalculation = amountForDifferentValue
        .Select(item => item[0, 0] / (double)totalRows)
        .Select(division => -division * Math.Log(division, 2))
        .ToList();

    return stepsForCalculation.Sum();
}
```

Crear árbol de decisiones: decidir raíz

```
private static nodo GetRootNode(DataTable data, string edge)
{
    var attributes = new List<atributo>();
    var highestInformationGainIndex = -1;
    var highestInformationGain = double.MinValue;

    // Get all names, amount of attributes and attributes for every column
    for (var i = 0; i < data.Columns.Count - 1; i++)
    {
        var differentAttributenames = atributo.GetDifferentAttributeNamesOfColumn(data, i);
        attributes.Add(new atributo(data.Columns[i].ToString(), differentAttributenames));
    }

    // Calculate Entropy (S)
    var tableEntropy = calcularEntropía(data);

    for (var i = 0; i < attributes.Count; i++)
    {
        attributes[i].InformationGain = GetGainForAllAttributes(data, i, tableEntropy);

        if (attributes[i].InformationGain > highestInformationGain)
        {
            highestInformationGain = attributes[i].InformationGain;
            highestInformationGainIndex = i;
        }
    }

    return new nodo(attributes[highestInformationGainIndex].nombre, highestInformationGainIndex, attributes[highestInformati
}
```

Crear árbol de decisiones: calcular sub-tablas

```
private static DataTable CreateSmallerTable(DataTable data, string edgePointingToNextNode, int rootTableIndex)
{
    var smallerData = new DataTable();

    // add column titles
    for (var i = 0; i < data.Columns.Count; i++)
    {
        smallerData.Columns.Add(data.Columns[i].ToString());
    }

    // add rows which contain edgePointingToNextNode to new datatable
    for (var i = 0; i < data.Rows.Count; i++)
    {
        if (data.Rows[i][rootTableIndex].ToString().Equals(edgePointingToNextNode))
        {
            var row = new string[data.Columns.Count];

            for (var j = 0; j < data.Columns.Count; j++)
            {
                row[j] = data.Rows[i][j].ToString();
            }

            smallerData.Rows.Add(row);
        }
    }

    // remove column which was already used as node
    smallerData.Columns.Remove(smallerData.Columns[rootTableIndex]);

    return smallerData;
}
```