Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Inteligencia Artificial 1
Ing. Luis Espino
Aux. Erick Sandoval

Manual de Técnico

Nombre: Sergio Rivelino Pérez Rivera
Carnet: 201603156

Herramientas Utilizadas

JavaScript
JavaScript es un lenguaje de programación que permite agregar interactividad a las páginas web, funcionando tanto en el navegador como en el servidor para crear aplicaciones dinámicas que responden a las acciones de los usuarios.

HTML

HTML, o HyperText Markup Language, es el lenguaje de marcado que estructura y presenta contenido en la web, definiendo elementos como textos, imágenes y enlaces, y organizándolos para su visualización en navegadores.

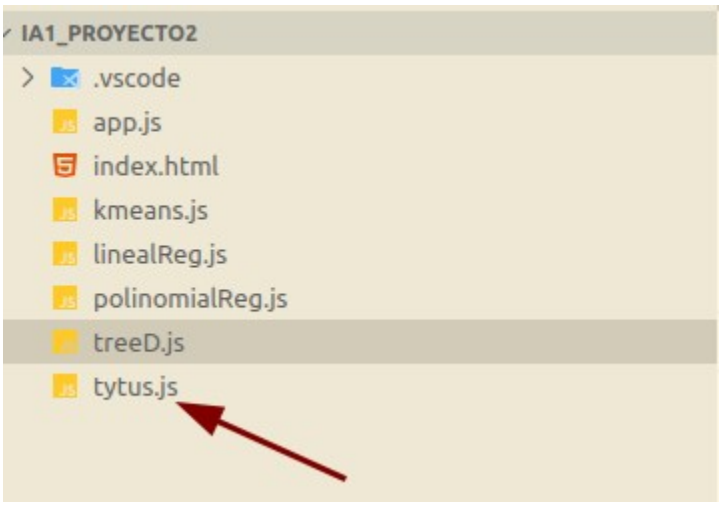
VIS.JS

Vis.js es una librería de JavaScript que facilita la creación de visualizaciones interactivas de datos, como gráficos de redes y líneas de tiempo, ofreciendo herramientas para representar relaciones y cambios de manera visual en aplicaciones que requieren gráficos dinámicos y personalizables.

Tytus.js

Es una librería de JS que nos permite utilizar métodos de Machine Learning

Directorio del Proyecto



Se creo un archivo .js para cada método de Machine Learning; también un archivo app.js para procesar los .csv y un archivo index.html que contiene todos los elementos principales de la página web. Por ultimo, el archivo tytus.js que nos permitirá utilizar las funciones de los métodos de Machine Learning.

Ejemplo de cómo instanciar una clase:

En el código se puede observar que se creo una instancia la clase LinearRegression(). De esta manera podemos acceder a funciones y métodos como **fit y predict**

```
const linear = new LinearRegression();
linear.fit(xTrain, yTrain);
const yPredict = linear.predict(xTrain);
plotDataAndRegressionLine(xTrain, yTrain, yPredict);
showPredictions(xTrain, yPredict, "Regresión Lineal");
```

Instancia de la clase PolynomialRegression de la librería tytus.js

polinomialReg.js > applyPolynomialRegression > poly

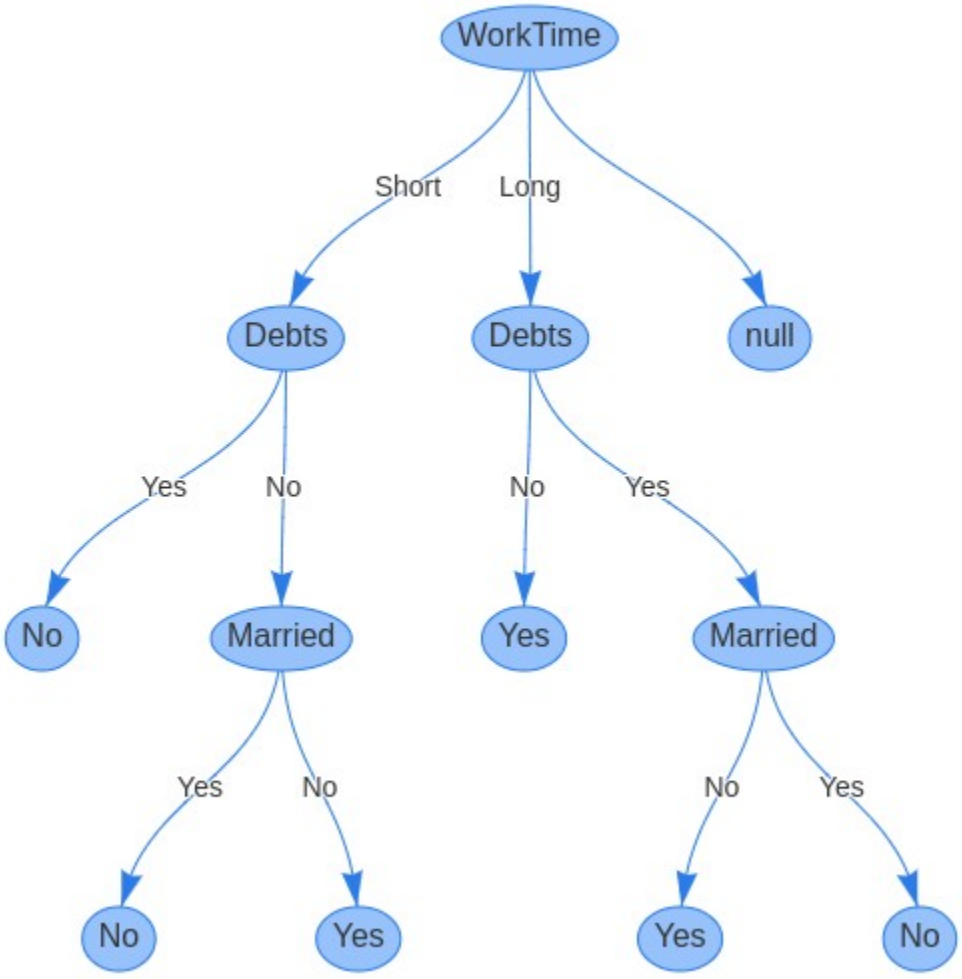
```
1  function applyPolynomialRegression(xTrain, yTrain, degree) {
2      try {
3          const poly = new PolynomialRegression();
4          poly.fit(xTrain, yTrain, degree);
5          const yPredict = poly.predict(xTrain);
6          plotDataAndPolynomialCurve(xTrain, yTrain, yPredict, degree);
7          showPredictions(xTrain, yPredict, `Regresión Polinomial (grado ${degree})`
```

Instancia de DecisionTreeID3

El método generateDotString() nos devuelve código dot para poder generar el árbol de decisión

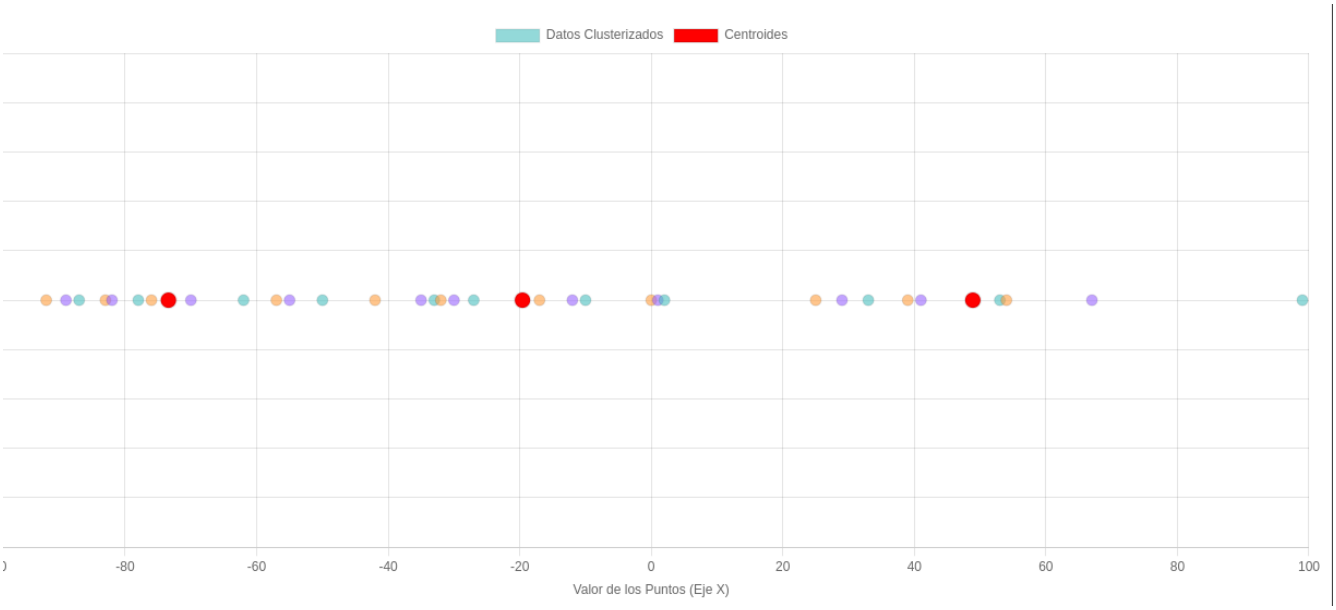
```
function initializeDecisionTree(dataset) {
    const decisionTree = new DecisionTreeID3();
    const rootNode = decisionTree.train(dataset, 0);
    return { decisionTree, rootNode };
}

function generateDotForVis(decisionTree, rootNode) {
    return decisionTree.generateDotString(rootNode);
}
```



Para generar las gráficas de la regresión lineal y polinomial, así como kmeans, se utilizó la librería:

```
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
```



Sobre los Métodos Utilizados:

K-means es un algoritmo de agrupamiento que clasifica un conjunto de datos en un número predefinido de grupos (k), asignando cada punto al cluster cuyo centroide (promedio de los puntos en el grupo) es más cercano, y ajustando los centroides iterativamente hasta que se estabilizan.

Los **árboles de decisiones** son modelos de aprendizaje automático que utilizan una estructura en forma de árbol para tomar decisiones basadas en las características de los datos. Cada nodo interno representa una pregunta sobre una característica, cada rama representa el resultado de esa pregunta, y cada hoja representa una clase o resultado final, permitiendo interpretaciones claras y visuales de cómo se toman las decisiones.

La **regresión lineal** es un modelo estadístico que busca establecer una relación lineal entre una variable dependiente y una o más variables independientes, representado en la forma de una ecuación lineal. La **regresión polinomial**, por otro lado, extiende este concepto al incluir términos de mayor grado (potencias) de la variable independiente, permitiendo captar relaciones no lineales en los datos.

Estructura de los archivos .csv para testear el método de Árbol de Decisión:

```
Income,WorkTime,Married,Debts,Children,BuyHouse
High,Short,No,Yes,No,No
High,Long,Yes,No,No,Yes
Low,Long,No,No,No,Yes
Medium,Short,Yes,No,Several,No
High,Short,No,No,No,Yes
High,Long,No,Yes,No,Yes
Medium,Long,Yes,No,One,Yes
Low,Long,Yes,Yes,One,No
Low,Short,No,Yes,One,No
Medium,Long,Yes,No,Several,Yes
Low,Short,Yes,Yes,Several,No
```

archivo .csv para el método kmeans:

```
data
99
-92
-89
-87
-83
-82
-78
-76
-70
-62
-57
-55
-50
-42
-35
-33
-32
-30
-27
-17
-12
-10
0
1
2
25
29
33
39
41
53
54
67
```

Index.html

index.html > html > body > div.content-app > input#fileInput

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Machine Learning - Regresiones</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/vis/4.21.0/vis.min.js"></script>
    <script src="linealReg.js"></script>
    <script src="polinomialReg.js"></script>
    <script src="tytus.js"></script>
</head>
<body>
    <h1>Proyecto 2 - Machine Learning</h1>
    <div class="content-app">
        <input type="file" id="fileInput" accept=".csv">

        <p>Seleccionar Método</p>
        <select id="methodSelect">
            <option value="linear">Regresión Lineal</option>
            <option value="polynomial">Regresión Polinomial</option>
            <option value="tree">Árbol de Decisón</option>
            <option value="kmeans">KMeans</option>
        </select>

        <div id="degreeContainer" style="display: none; margin-top: 0px;">
            <label for="degreeInput">Grado:</label>
            <input type="number" id="degreeInput" min="1" value="2">
        </div>

        <div>
            <label for="xColumn">Seleccionar columna X:</label>
            <select id="xColumn"></select>

            <label for="yColumn">Seleccionar columna Y:</label>
            <select id="yColumn"></select>
        </div>



        <p>Valores a predecir (separados por coma)</p>
        <input id="valoresPredecir" type="text">
        <button id="executeButton" style="background-color: #4CAF50; color: white; border: none; padding: 8px 16px; border-radius: 5px; cursor: pointer; tra
        <button id="entrenarButton" style="background-color: #2196F3; color: white; border: none; padding: 8px 16px; border-radius: 5px; cursor: pointer; tr
        <button id="predictButton" style="background-color: #FF5722; color: white; border: none; padding: 8px 16px; border-radius: 5px; cursor: pointer; tra
        <button id="patronesButton" style="background-color:  #2196F3; color: white; border: none; padding: 8px 16px; border-radius: 5px; cursor: pointer; t
        <button id="clearButton" style="background-color: #f44336; color: white; border: none; padding: 8px 16px; border-radius: 5px; cursor: pointer; trans




        <div id="kmeansContainer" style="display: none; margin-top: 10px; width: 300px; height: 70px; padding: 15px; border: 1px solid #ccc; border-radius: 8p
            <p style="margin: 0; margin-right: 10px; display: inline-block;">Ingrese K:</p>
            <input id="k" type="text" style="margin-right: 20px; width: 50px;">
            <p style="margin: 0; margin-right: 10px; display: inline-block;">Ingrese el número de iteraciones:</p>
            <input id="iteraciones" type="text" style="width: 50px;">
        </div>




        <div id="predictionResults" style="display: none; margin-top: 0px;"></div>

        <canvas id="myChart" style="display: none; margin-top: 0px;" width="400" height="150"></canvas>

        <div id="treeContainerGraph" style="display: none; margin-top: 0px;">
```

Código para implementar el árbol de decisión:

treeD.js > generateDotForVis

```
function initializeDecisionTree(dataset) {
    const decisionTree = new DecisionTreeID3();
    const rootNode = decisionTree.train(dataset, 0);
    return { decisionTree, rootNode };
}

function generateDotForVis(decisionTree, rootNode) {
    return decisionTree.generateDotString(rootNode);
}

function getVisOptions() {
    return {
        nodes: {
            shape: 'ellipse',
            font: { size: 16 }
        },
        edges: {
            arrows: { to: { enabled: true } },
            smooth: { type: 'cubicBezier' }
        },
        layout: {
            hierarchical: {
                direction: 'UD', // De arriba hacia abajo
                sortMethod: 'directed'
            }
        }
    };
}


function renderTree(dotString, options) {
    const container = document.getElementById('tree-container');
    const data = vis.network.convertDot(dotString);
    return new vis.Network(container, data, options);
}


function initDecisionTree(dataset, userInput) {
    console.log("dataset en el archivo treeD.js", dataset);


    const predictData = [
        ["Income", "WorkTime", "Married", "Debts", "Children"],
        userInput.split(",")
    ];

    const { decisionTree, rootNode } = initializeDecisionTree(dataset);
    const dotString = generateDotForVis(decisionTree, rootNode);
    const options = getVisOptions();
    renderTree(dotString, options);
    predictAndDisplayResult(decisionTree, rootNode, predictData);
}


function predictAndDisplayResult(decisionTree, rootNode, predictData) {
    const predictionResult = decisionTree.predict(predictData, rootNode);
    document.getElementById('result').textContent = `La predicción es: ${predictionResult.value}`;
}

```

```
document.addEventListener("DOMContentLoaded", function () {
    const fileInput = document.getElementById("fileInput");
    const methodSelect = document.getElementById("methodSelect");
    const degreeContainer = document.getElementById("degreeContainer");
    const degreeInput = document.getElementById("degreeInput");
    const executeButton = document.getElementById("executeButton");
    const clearButton = document.getElementById("clearButton");
    const xColumnSelect = document.getElementById("xColumn");
    const yColumnSelect = document.getElementById("yColumn");
    const valoresPredecir = document.getElementById("valoresPredecir");
    const k = document.getElementById("k");
    const iteraciones = document.getElementById("iteraciones");
    const predictButton_ = document.getElementById("predictButton");
    const predictionResults = document.getElementById("predictionResults");
    const entrenarButton = document.getElementById("entrenarButton");
    const myChart = document.getElementById("myChart");
```

Código app.js y cómo se cargaron la mayoría de elementos HTML para su respectiva manipulación: