Manual Técnico

**Proyecto 2 - Machine Learning**

**Arquitectura**

El proyecto está construido con HTML, CSS y JavaScript. Utiliza la biblioteca de Google Charts para visualizar datos y predicciones.

**Estructura de Archivos**

* index.html: Archivo principal que contiene la estructura de la página.
* styles.css: Archivo de estilos para el diseño visual.
* tytus.js: Archivo de JavaScript que contiene la lógica del modelo de machine learning.

**Lógica de Código**

1. **Carga de CSV:** La función loadCSV se encarga de leer el archivo CSV seleccionado y procesar los datos utilizando processCSV, que separa los datos en dos arreglos: xTrain y yTrain.
2. **Entrenamiento de Modelos:**
   * trainModel(): Dependiendo del modelo seleccionado, esta función entrena el modelo correspondiente:
     + **Regresión Lineal**: Se entrena con LinearRegression.
     + **Regresión Polinomial**: Se entrena usando PolynomialRegression y se realizan predicciones para grados 2, 3 y 4.
     + **Redes Neuronales**: Entrenamiento a través de trainNeuralNetworks.
3. **Predicción:**
   * predictModel(): Realiza predicciones basadas en el modelo entrenado y actualiza el log.
4. **Visualización:**
   * Las gráficas se muestran utilizando google.charts. La función showPolynomialGraph se utiliza para graficar los resultados de la regresión polinomial.
   * showTrend() y showPatterns() proporcionan análisis adicionales sobre los datos.

**Ejemplo de Uso de la Lógica**

Para entrenar un modelo de regresión polinomial:

javascript

Copiar código

polynomial.fit(xTrain, yTrain, 2);

yPredictDegree2 = polynomial.predict(xTrain);

**Conclusión**

Este proyecto es una herramienta educativa que permite a los usuarios experimentar con diferentes técnicas de machine learning. La estructura de código está diseñada para ser modular y fácil de entender, facilitando futuras mejoras y extensiones.

**Codigo**

<!DOCTYPE html>

<html lang=*"es"*>

<head>

    <meta charset=*"UTF-8"*>

    <meta name=*"viewport"* content=*"width=device-width, initial-scale=1.0"*>

    <title>Proyecto 2 - Machine Learning</title>

    <link rel=*"stylesheet"* href=*"styles.css"*>

    <script type=*"text/javascript"* src=*"tytus.js"*></script>

    <script type=*"text/javascript"* src=*"https://www.gstatic.com/charts/loader.js"*></script>

</head>

<body>

    <div class=*"container"*>

        <h2>Proyecto 2 - Machine Learning - Inteligencia Artificial 1</h2>

        <p id=*"log"*></p>

        <label for=*"modelSelect"*>Seleccione el modelo:</label>

        <select id=*"modelSelect"*>

            <option value=*"seleccionar"*>Seleccionar</option>

            <option value=*"regresionLineal"*>Regresión Lineal</option>

            <option value=*"regresionLogistica"*>Regresión Logistica</option>

            <option value=*"regresionPolinomial"*>Regresión Polinomial</option>

            <option value=*"NaiveBayes"*>Naive Bayes</option>

            <option value=*"redesNeuronales"*>Redes Neuronales</option>

            <option value=*"kmeans"*>K-means</option>

            <option value=*"kneighbor"*>K-nearest Neighbor</option>

        </select>

        <input type=*"file"* id=*"fileInput"* accept=*".csv"* />

        <button onclick=*"trainModel()"*>Entrenamiento</button>

        <button onclick=*"predictModel()"*>Predicción</button>

        <button onclick=*"showGraph()"*>Mostrar Gráficas</button>

        <button onclick=*"showTrend()"*>Tendencia</button>

        <button onclick=*"showPatterns()"*>Patrones</button>

        <div id=*"chart\_div"*></div>

    </div>

    <script type=*"text/javascript"*>

        let xTrain = [];

let yTrain = [];

let yPredict = [];

let linear;

let polynomial;  *// Nueva variable para el modelo de regresión polinomial*

let yPredictDegree2 = [];

let yPredictDegree3 = [];

let yPredictDegree4 = [];

let r2, r3, r4;

function trainNeuralNetworks() {

        loadCSV();

*// Primer red neuronal*

        let design1 = [2, 4, 3, 2];

        let brain1 = new NeuralNetwork(design1);

        for (let i = 0; i < xTrain.length; i++) {

            let input = [xTrain[i], yTrain[i]];

            let output = (xTrain[i] > yTrain[i] ? [1, 0] : [0, 1]);

            brain1.Entrenar(input, output);

        }

        let largest1 = brain1.Predecir([10, 20]);

        document.getElementById(*"logE"*).innerHTML = String(*"Probability Score for Largest: "* + brain1.Predecir([10, 20]));

        console.log(*`Probability Score for Largest (Brain 1): ${largest1}`*);

*// Segunda red neuronal*

        let design2 = [2, 4, 3, 2];

        let brain2 = new NeuralNetwork(design2);

        for (let i = 0; i < xTrain.length; i++) {

            let input = [xTrain[i], yTrain[i]];

            let output = (xTrain[i] < yTrain[i] ? [1, 0] : [0, 1]);

            brain2.Entrenar(input, output);

        }

        let largest2 = brain2.Predecir([20, 5]);

        document.getElementById(*"logEE"*).innerHTML = String(*"Probability Score for Largest: "* + brain2.Predecir([20, 5]));

        console.log(*`Probability Score for Largest (Brain 2): ${largest2}`*);

    }

function loadCSV() {

    const fileInput = document.getElementById(*'fileInput'*);

    const file = fileInput.files[0];

    if (!file) {

        alert(*'Por favor, selecciona un archivo CSV.'*);

        return;

    }

    const reader = new FileReader();

    reader.onload = function(*event*) {

        const csvData = event.target.result;

        processCSV(csvData);

    };

    reader.readAsText(file);

}

function processCSV(*data*) {

    const lines = data.split(*'\n'*);

    xTrain = [];

    yTrain = [];

    for (let i = 1; i < lines.length; i++) {

        const line = lines[i].split(*','*);

        if (line.length === 2) {

            xTrain.push(parseFloat(line[0]));

            yTrain.push(parseFloat(line[1]));

        }

    }

    document.getElementById(*"log"*).innerHTML += *'<br>Datos cargados exitosamente.'*;

}

function trainModel() {

    document.getElementById(*"log"*).innerHTML = *""*;

    const model = document.getElementById(*'modelSelect'*).value;

    if (model === *'regresionLineal'*) {

        loadCSV();

        linear = new LinearRegression();

        linear.fit(xTrain, yTrain);

        document.getElementById(*"log"*).innerHTML += *'<br>Modelo de Regresión Lineal entrenado.<br>X Train: '* + xTrain + *'<br>Y Train: '* + yTrain;

    } else if (model === *'regresionPolinomial'*) {

        loadCSV();

        polynomial = new PolynomialRegression();

*// Entrenamiento y predicción para grados 2, 3 y 4*

        polynomial.fit(xTrain, yTrain, 2);

        yPredictDegree2 = polynomial.predict(xTrain);

        r2 = polynomial.getError();

        polynomial.fit(xTrain, yTrain, 3);

        yPredictDegree3 = polynomial.predict(xTrain);

        r3 = polynomial.getError();

        polynomial.fit(xTrain, yTrain, 4);

        yPredictDegree4 = polynomial.predict(xTrain);

        r4 = polynomial.getError();

*// Redondear valores*

        [yPredictDegree2, yPredictDegree3, yPredictDegree4].forEach(*arr* => {

            for (let i = 0; i < arr.length; i++) {

                arr[i] = Number(arr[i].toFixed(2));

            }

        });

        document.getElementById(*"log"*).innerHTML += *'<br>Modelo de Regresión Polinomial entrenado.<br>X Train: '* + xTrain + *'<br>Y Train: '* + yTrain;

        document.getElementById(*"log"*).innerHTML += *'<br>Y Predicción Grado 2: '* + yPredictDegree2;

        document.getElementById(*"log"*).innerHTML += *'<br>Y Predicción Grado 3: '* + yPredictDegree3;

        document.getElementById(*"log"*).innerHTML += *'<br>Y Predicción Grado 4: '* + yPredictDegree4;

        document.getElementById(*"log"*).innerHTML += *'<br>R^2 Grado 2: '* + Number(r2.toFixed(2));

        document.getElementById(*"log"*).innerHTML += *'<br>R^2 Grado 3: '* + Number(r3.toFixed(2));

        document.getElementById(*"log"*).innerHTML += *'<br>R^2 Grado 4: '* + Number(r4.toFixed(2));

    } else if (model === *'redesNeuronales'*) {

        trainNeuralNetworks();

    } else {

        alert(*'Por favor, selecciona un modelo válido para entrenar.'*);

    }

}

function predictPolynomial() {

    if (polynomial) {

        const predictArray = xTrain;

        yPredictDegree2 = polynomial.predict(predictArray);

        yPredictDegree3 = polynomial.predict(predictArray);

        yPredictDegree4 = polynomial.predict(predictArray);

        document.getElementById(*"log"*).innerHTML += *'<br>Predicción Polinomial realizada.'*;

    } else {

        alert(*'Primero entrena el modelo de regresión polinomial antes de predecir.'*);

    }

}

function joinArrays() {

    var a = [];

    if (*arguments*.length > 0) {

*// Agrega los encabezados*

        var headers = [];

        for (var i = 0; i < *arguments*.length; i += 2) {

            headers.push(*arguments*[i]); *// Tomar los nombres de las series*

        }

        a.push(headers);

*// Asumiendo que todas las series tienen la misma longitud*

        var maxLength = *arguments*[1].length; *// Longitud de la primera serie*

        for (var i = 0; i < maxLength; i++) {

            var row = [];

            for (var j = 1; j < *arguments*.length; j += 2) {

                row.push(*arguments*[j][i]); *// Agregar los valores de cada serie*

            }

            a.push(row);

        }

    }

    return a;

}

function showPolynomialGraph() {

    if (yPredictDegree2.length > 0 || yPredictDegree3.length > 0 || yPredictDegree4.length > 0) {

        var dataArray = joinArrays(

*'x'*, xTrain,

*'Y Train'*, yTrain,

*'Predicción Grado 2'*, yPredictDegree2,

*'Predicción Grado 3'*, yPredictDegree3,

*'Predicción Grado 4'*, yPredictDegree4

        );

        google.charts.load(*'current'*, { *'packages'*: [*'corechart'*] });

        google.charts.setOnLoadCallback(() => {

            var data = google.visualization.arrayToDataTable(dataArray);

            var options = {

                seriesType: *'scatter'*,

                series: {

                    1: { type: *'line'* },

                    2: { type: *'line'* },

                    3: { type: *'line'* }

                }

            };

            var chart = new google.visualization.ComboChart(document.getElementById(*'chart\_div'*));

            chart.draw(data, options);

        });

    } else {

        alert(*'Primero realiza una predicción polinomial para mostrar la gráfica.'*);

    }

}

        function predictModel() {

            const model = document.getElementById(*'modelSelect'*).value;

            if(model === *'regresionPolinomial'*){

                predictPolynomial();

            } else {

                if (linear) {

                   yPredict = linear.predict(xTrain);

                   document.getElementById(*"log"*).innerHTML += *'<br>Predicción realizada.<br>Y Predict: '* + yPredict;

                } else {

                   alert(*'Primero entrena el modelo antes de predecir.'*);

                }

            }

        }

        function showGraph() {

            const model = document.getElementById(*'modelSelect'*).value;

            if(model === *'regresionPolinomial'*){

                showPolynomialGraph();

            } else {

                if (yPredict.length > 0) {

                    const dataArray = joinArrays(*'x'*, xTrain, *'yTrain'*, yTrain, *'yPredict'*, yPredict);

                    google.charts.load(*'current'*, {*'packages'*: [*'corechart'*]});

                    google.charts.setOnLoadCallback(() => drawChart(dataArray));

                } else {

                    alert(*'Primero realiza una predicción para mostrar la gráfica.'*);

                }

            }

        }

        function showTrend() {

            if (xTrain.length > 1 && yTrain.length > 1) {

                const trendData = [];

                for (let i = 0; i < xTrain.length; i++) {

                    trendData.push([xTrain[i], yTrain[i]]);

                }

*// Calcular dirección de tendencia*

                const slope = (yTrain[yTrain.length - 1] - yTrain[0]) / (xTrain[xTrain.length - 1] - xTrain[0]);

                const trendText = slope > 0 ? *"La tendencia es ascendente."* : *"La tendencia es descendente."*;

                document.getElementById(*"log"*).innerHTML += *`<br>${trendText}`*;

                google.charts.load(*'current'*, {*'packages'*: [*'corechart'*]});

                google.charts.setOnLoadCallback(() => drawTrendChart(trendData));

            } else {

                alert(*'Primero carga y entrena un modelo con datos suficientes.'*);

            }

        }

        function showPatterns() {

            if (xTrain.length > 1 && yTrain.length > 1) {

                const avgData = xTrain.map((*x*, *index*) => [x, yTrain[index]]);

*// Análisis de patrones: detectar picos y valles*

                let peaks = 0, valleys = 0;

                for (let i = 1; i < yTrain.length - 1; i++) {

                    if (yTrain[i] > yTrain[i - 1] && yTrain[i] > yTrain[i + 1]) peaks++;

                    if (yTrain[i] < yTrain[i - 1] && yTrain[i] < yTrain[i + 1]) valleys++;

                }

                const patternText = *`Patrones detectados: ${peaks} picos y ${valleys} valles en los datos.`*;

                document.getElementById(*"log"*).innerHTML += *`<br>${patternText}`*;

                google.charts.load(*'current'*, {*'packages'*: [*'corechart'*]});

                google.charts.setOnLoadCallback(() => drawPatternChart(avgData));

            } else {

                alert(*'Primero carga y entrena un modelo con datos suficientes.'*);

            }

        }

        function drawChart(*dataArray*) {

            var data = google.visualization.arrayToDataTable(dataArray);

            var options = {

                seriesType: *'scatter'*,

                series: {1: {type: *'line'*}}

            };

            var chart = new google.visualization.ComboChart(document.getElementById(*'chart\_div'*));

            chart.draw(data, options);

        }

        function drawChart2(*dataArray*) {

            var data = google.visualization.arrayToDataTable(dataArray);

            var options = {

                seriesType: *'scatter'*,

                series: {

                    1: { type: *'line'* },

                    2: { type: *'line'* },

                    3: { type: *'line'* }

                }

            };

            var chart = new google.visualization.ComboChart(document.getElementById(*'chart\_div'*));

            chart.draw(data, options);

        }

        function drawTrendChart(*dataArray*) {

            const data = google.visualization.arrayToDataTable([[*'X'*, *'Y'*], ...dataArray]);

            const options = {

                title: *'Tendencia de los datos'*,

                trendlines: { 0: {} },

                legend: { position: *'bottom'* }

            };

            const chart = new google.visualization.LineChart(document.getElementById(*'chart\_div'*));

            chart.draw(data, options);

        }

        function drawPatternChart(*dataArray*) {

            const data = google.visualization.arrayToDataTable([[*'X'*, *'Y'*], ...dataArray]);

            const options = {

                title: *'Patrones en los datos'*,

                hAxis: { title: *'X'* },

                vAxis: { title: *'Y'* },

                legend: *'none'*

            };

            const chart = new google.visualization.ColumnChart(document.getElementById(*'chart\_div'*));

            chart.draw(data, options);

        }

    </script>

</body>

</html>