Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Inteligencia Artificial 1

Aux. Erick Eden Sandoval Ramirez

MANUAL TECNICO

Nombre: Luis Enrique Garcia Gutierrez

Carnet: 202010814

PRESENTACION

El siguiente manual describe los pasos necesarios que cualquier persona que tenga cierta base de programación en HTML, JS y en modelos de Machine Learning puede hacer uso y entender el funcionamiento del programa, a su vez guiara a los usuarios que harán soporte al sistema, el cual les dará a conocer los requerimientos y la estructura para la construcción del sistema.

PROCESOS

Procesos de entrada:

- Carga de archivo .csv
- Ingreso de parámetros para el modelo
- Selección del modelo a entrenar

Procesos de salida:

- Graficas de predicción para cada modelo
- Graficas de tendencia para cada modelo
- Resultados de los modelos

DESCRIPCION DE FUNCIONES UTILIZADAS

Estas dos funciones se utilizan para unir arrays que se utilizaran para poder realizar la gráfica de los modelos.

```
function joinArrays() {
    var a = []

    if (arguments.length == 6) {
        a.push([arguments[0], arguments[2], arguments[4]])
        for (var i = 0; i < arguments[1].length; i++) {
            a.push([arguments[1][i], arguments[3][i], arguments[5][i]])
        }
    }
    return a
}


function joinArraysPolynomial() {
    var a = []
    if (arguments.length == 10) {
        a.push([arguments[0], arguments[2], arguments[4], arguments[6], arguments[8]]);
        for (var i = 0; i < arguments[1].length; i++) {
            a.push([arguments[1][i], arguments[3][i], arguments[5][i], arguments[7][i], arguments[9][i]]);
        }
    }
    return a;
}
```

La función parseCSV() recibe el contenido del dataset para extraer cada fila del archivo.

```
function parseCSV(csv) {
    const rows = csv.split('\n').map(row => row.split(','));
    const headers = rows[0];
    const data = rows.slice(1).map(row => {
        let obj = {};
        row.forEach((value, index) => {
            obj[headers[index]] = parseFloat(value) || value;
        });
        return obj;
    });
    return data;
}
```

La función populateVariableSelects() almacena cada variable del dataset para almacenarlos en el select.

```
function populateVariableSelects(columns) {
    const inputSelect = document.getElementById('input-variable');
    const outputSelect = document.getElementById('output-variable');

    inputSelect.innerHTML = '';
    outputSelect.innerHTML = '';

    columns.forEach(col => {
        const option = document.createElement('option');
        option.value = col;
        option.textContent = col;

        inputSelect.appendChild(option.cloneNode(true));
        outputSelect.appendChild(option);
    });
}
```

La función Entrenar() se ejecuta al momento de darle al botón Entrenar, en esta función se verifica el modelo seleccionado y después se muestran los componentes necesarios para el modelo y hacer el entrenamiento

```
function Entrenar() {

    const inputSelect = document.getElementById('modelos').value;
    console.log(inputSelect)

    if (inputSelect === "LinealRegression") {
        document.getElementById('tree-data').style.display = 'none'; //o
        document.getElementById("chart_div").style.display = "block"; //
        document.getElementById("log1").innerHTML = "";
        document.getElementById("log2").innerHTML = "";
        document.getElementById("log3").innerHTML = "";
        document.getElementById("log4").innerHTML = "";
        document.getElementById("log5").innerHTML = "";
        document.getElementById("log6").innerHTML = "";
        document.getElementById("log7").innerHTML = "";
        document.getElementById("log8").innerHTML = "";
        document.getElementById("log9").innerHTML = "";
        LinealRegresion()
    } else if (inputSelect === "PolynomialRegression") {
        document.getElementById('tree-data').style.display = 'none'; //o
        document.getElementById("chart_div").style.display = "block"; //
        Polynomial()
    } else if (inputSelect === "DecisionTree") {
        document.getElementById('tree-data').style.display = 'block'; //
        document.getElementById("chart_div").style.display = "none"; //o
        document.getElementById("log1").innerHTML = "";
        document.getElementById("log2").innerHTML = "";
        document.getElementById("log3").innerHTML = "";
        document.getElementById("log4").innerHTML = "";
        document.getElementById("log5").innerHTML = "";
        document.getElementById("log6").innerHTML = "";
        document.getElementById("log7").innerHTML = "";
        document.getElementById("log8").innerHTML = "";
        document.getElementById("log9").innerHTML = "";
        DecisionTree()
    }

}
```

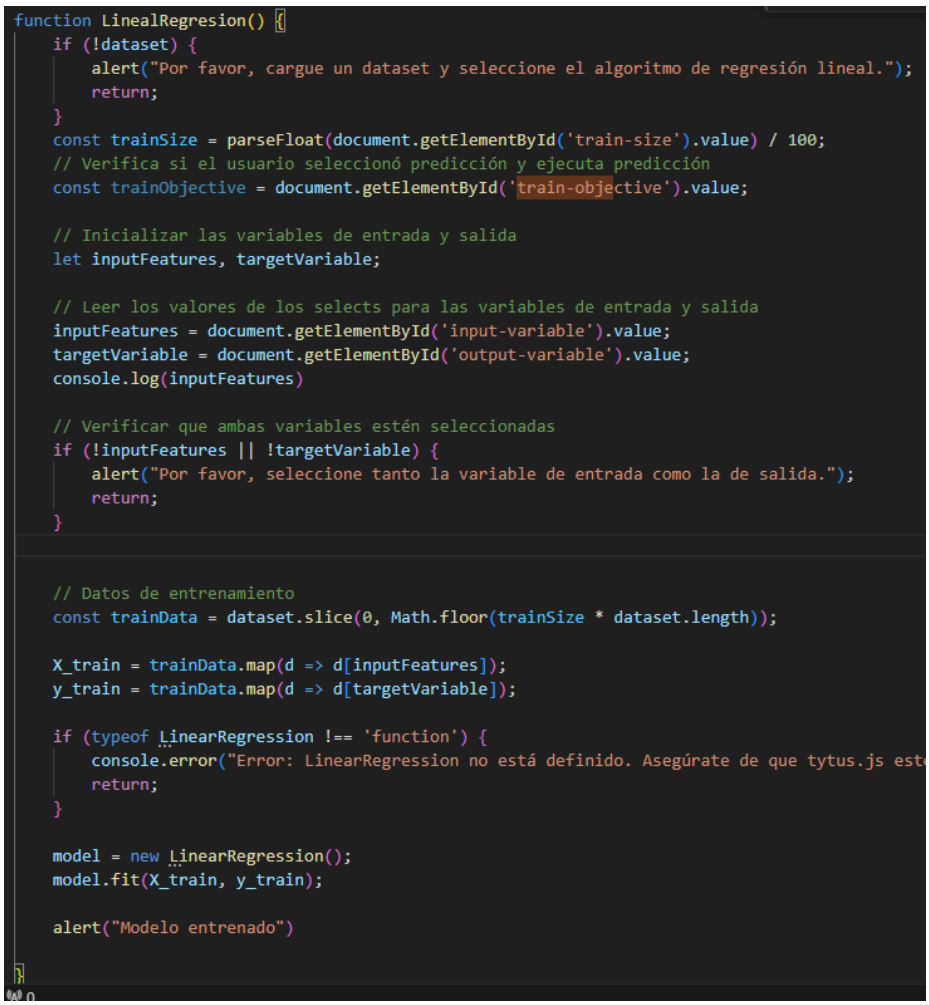
La funciion Predecir() se ejecuta al darle al botón Predecir el cual verifica el modelo seleccionado para mostrar los componentes necesarios y hacer la predicción.

```
function Predecir() {
    const inputSelect = document.getElementById('modelos').value;
    console.log(inputSelect)

    if (inputSelect === "LinealRegression") {
        document.getElementById('tree-data').style.display = 'none';
        document.getElementById("chart_div").style.display = "block";
        document.getElementById("log1").innerHTML = "";
        document.getElementById("log2").innerHTML = "";
        document.getElementById("log3").innerHTML = "";
        document.getElementById("log4").innerHTML = "";
        document.getElementById("log5").innerHTML = "";
        document.getElementById("log6").innerHTML = "";
        document.getElementById("log7").innerHTML = "";
        document.getElementById("log8").innerHTML = "";
        document.getElementById("log9").innerHTML = "";
        PrediccionLinealRegresion()
    } else if (inputSelect === "PolynomialRegression") {
        document.getElementById('tree-data').style.display = 'none';
        document.getElementById("chart_div").style.display = "block";

        PrediccionPolynomial()
    } else if (inputSelect === "DecisionTree") {
        document.getElementById('tree-data').style.display = 'block';
        document.getElementById("chart_div").style.display = "none";
        document.getElementById("log1").innerHTML = "";
        document.getElementById("log2").innerHTML = "";
        document.getElementById("log3").innerHTML = "";
        document.getElementById("log4").innerHTML = "";
        document.getElementById("log5").innerHTML = "";
        document.getElementById("log6").innerHTML = "";
        document.getElementById("log7").innerHTML = "";
        document.getElementById("log8").innerHTML = "";
        document.getElementById("log9").innerHTML = "";

        DecisionTree()
    }

}
```

La función Tendencia() se ejecuta después de que dar click al botón Tendencia, el cual verifica el modelo para poder mostrar los componentes necesarios y realizar la tendencia que tiene el modelo entrenado.

```
function Tendencia() {
    const inputSelect = document.getElementById('modelos').value;

    if (inputSelect === "LinealRegression") {
        document.getElementById('tree-data').style.display = 'none'; /
        document.getElementById("chart_div").style.display = "block";
        document.getElementById("log1").innerHTML = "";
        document.getElementById("log2").innerHTML = "";
        document.getElementById("log3").innerHTML = "";
        document.getElementById("log4").innerHTML = "";
        document.getElementById("log5").innerHTML = "";
        document.getElementById("log6").innerHTML = "";
        document.getElementById("log7").innerHTML = "";
        document.getElementById("log8").innerHTML = "";
        document.getElementById("log9").innerHTML = "";
        Tendencias()
    } else if (inputSelect === "PolynomialRegression") {
        document.getElementById('tree-data').style.display = 'none'; /
        document.getElementById("chart_div").style.display = "block";

        Tendencias()
    } else if (inputSelect === "DecisionTree") {
        document.getElementById('tree-data').style.display = 'block';
        document.getElementById("chart_div").style.display = "none"; /
        document.getElementById("log1").innerHTML = "";
        document.getElementById("log2").innerHTML = "";
        document.getElementById("log3").innerHTML = "";
        document.getElementById("log4").innerHTML = "";
        document.getElementById("log5").innerHTML = "";
        document.getElementById("log6").innerHTML = "";
        document.getElementById("log7").innerHTML = "";
        document.getElementById("log8").innerHTML = "";
        document.getElementById("log9").innerHTML = "";

        alert("No se puede aplicar tendencias")
    }
}
```

La función LinealRegresion() se encarga de hacer el entrenamiento para el modelo LinealRegresion. Primero se verifica que se haya cargado el dataset. Después se obtiene el porcentaje de datos a ser entrenados. Luego se obtiene las variables de entrada y salida para almacenar esos datos para el entrenamiento

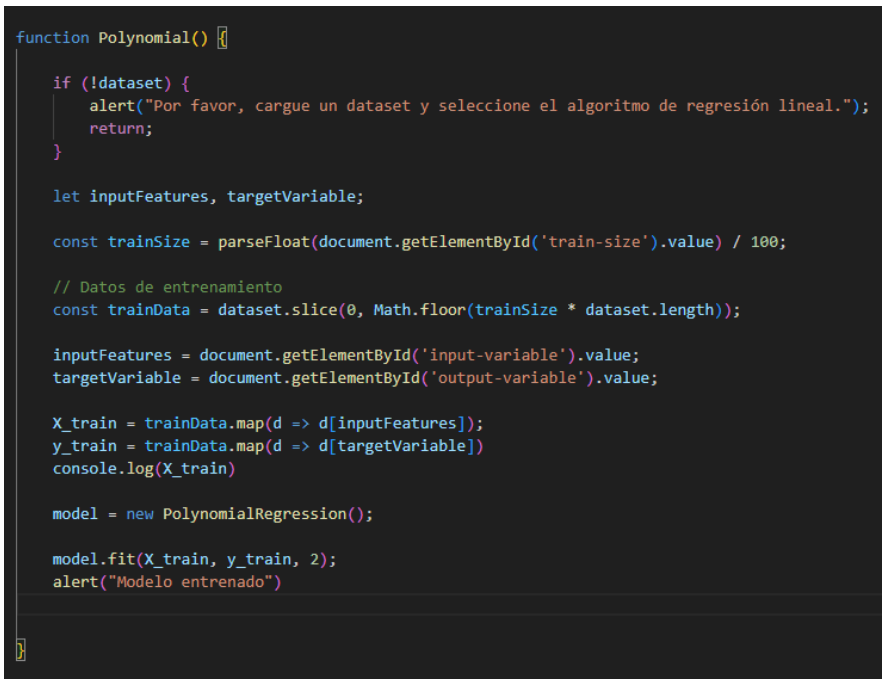


```
function LinealRegresion() {
    if (!dataset) {
        alert("Por favor, cargue un dataset y seleccione el algoritmo de regresión lineal.");
        return;
    }
    const trainSize = parseFloat(document.getElementById('train-size').value) / 100;
    // Verifica si el usuario seleccionó predicción y ejecuta predicción
    const trainObjective = document.getElementById('train-objective').value;

    // Inicializar las variables de entrada y salida
    let inputFeatures, targetVariable;

    // Leer los valores de los selects para las variables de entrada y salida
    inputFeatures = document.getElementById('input-variable').value;
    targetVariable = document.getElementById('output-variable').value;
    console.log(inputFeatures)

    // Verificar que ambas variables estén seleccionadas
    if (!inputFeatures || !targetVariable) {
        alert("Por favor, seleccione tanto la variable de entrada como la de salida.");
        return;
    }


    // Datos de entrenamiento
    const trainData = dataset.slice(0, Math.floor(trainSize * dataset.length));

    X_train = trainData.map(d => d[inputFeatures]);
    y_train = trainData.map(d => d[targetVariable]);

    if (typeof LinearRegression !== 'function') {
        console.error("Error: LinearRegression no está definido. Asegúrate de que tytus.js est
        return;
    }

    model = new LinearRegression();
    model.fit(X_train, y_train);

    alert("Modelo entrenado")

}
```

Esta función se encarga de realizar la predicción para el modelo Linear, primero verifica si el objetivo es predicción, si es así obtiene los valores del input para realizar la predicción, después se muestran en un log y se grafica. Caso contrario se toman los valores de la variable de entrada para realizar la predicción.

```
function PrediccionLinealRegresion() {

    const trainObjective = document.getElementById('train-objective').value;

    if (trainObjective === 'prediccion') {
        const predictionValue = document.getElementById('prediction-input').value;

        let cadena = predictionValue.toString()
        let numeros = cadena.split(",")
        let predict_values = []
        numeros.forEach((value) => {
            predict_values.push(value)
        })
        console.log(predict_values)

        //valor_predict = model.predict([numeroMasCercano]); // Predicción para el nuevo va
        let yPredict = model.predict(predict_values)

        document.getElementById("log1").innerHTML = yPredict

        console.log("Predicción para el valor ingresado:", yPredict);
        const a = joinArrays('x', X_train, 'yTrain', y_train, 'yPredict', yPredict);

        google.charts.load('current', { 'packages': ['corechart'] });
        google.charts.setOnLoadCallback(() => drawChart(a));

    } else {

        yPredict = model.predict(X_train); // Predicción para los datos de entrenamiento
        console.log("Predicción para el valor ingresado:", yPredict);
        document.getElementById("log1").innerHTML = yPredict
        const a = joinArrays('x', X_train, 'yTrain', y_train, 'yPredict', yPredict);

        google.charts.load('current', { 'packages': ['corechart'] });
        google.charts.setOnLoadCallback(() => drawChart(a));

    }
}
```

Esta función se encarga de entrenar el modelo de Polynomial, primero verifica que el dataset se haya cargado. Después se obtienen los datos para el entrenamiento en base al porcentaje de datos que se quieran entrenar. Después se obtiene la variable de entrada y salida para almacenar esos datos para el entrenamiento.



```
function Polynomial() {

    if (!dataset) {
        alert("Por favor, cargue un dataset y seleccione el algoritmo de regresión lineal.");
        return;
    }

    let inputFeatures, targetVariable;

    const trainSize = parseFloat(document.getElementById('train-size').value) / 100;

    // Datos de entrenamiento
    const trainData = dataset.slice(0, Math.floor(trainSize * dataset.length));

    inputFeatures = document.getElementById('input-variable').value;
    targetVariable = document.getElementById('output-variable').value;

    X_train = trainData.map(d => d[inputFeatures]);
    y_train = trainData.map(d => d[targetVariable])
    console.log(X_train)

    model = new PolynomialRegression();

    model.fit(X_train, y_train, 2);
    alert("Modelo entrenado")


}
```

Esta función se encarga de realizar la predicción para el modelo Polynomial. Primero verifica si el objetivo del modelo es predicción. Si es predicción se obtienen los valores que se desea predecir, caso contrario los valores de predicción serán los mismos que los datos de la variable de entrada. Luego se realizan las predicciones para

```
function PrediccionPolynomial() {

    const trainObjective = document.getElementById('train-objective').value;

    let predict_values = []

    if (trainObjective === "prediccion") {
        const predictionValue = document.getElementById('prediction-input').value;


        let cadena = predictionValue.toString()
        let numeros = cadena.split(",")
        numeros.forEach((value) => {
            predict_values.push(value)
        })
        console.log(predict_values)

    } else {
        predict_values = X_train
    }
    console.log(predict_values)

    model.fit(X_train, y_train, 2);
    yPredict = model.predict(predict_values);
    r2 = model.getError();

    model.fit(X_train, y_train, 3);
    let yPredict2 = model.predict(predict_values);
    r22 = model.getError();

    model.fit(X_train, y_train, 4);
    let yPredict3 = model.predict(predict_values);
    r23 = model.getError();

    for (let i = 0; i < predict_values.length; i++) {
        yPredict[i] = Number(yPredict[i].toFixed(2));
        yPredict2[i] = Number(yPredict2[i].toFixed(2));
        yPredict3[i] = Number(yPredict3[i].toFixed(2));
    }
```

Esta función se encarga de realizar el árbol de decisión y la predicción. Si no se elige la predicción tomara el primero registro del dataset para realizarlo.

```
function DecisionTree() {
    if (!dataset) {
        alert("Por favor, cargue un dataset y seleccione el algoritmo de regresión lineal.");
        return;
    }

    //mostrar .csv en html
    const codeElement = document.getElementById("text-info");
    codeElement.innerHTML = "[<br />"


    //Conseguir cada fila del dataset
    console.log("rows")
    const rows = csv.split("\n");
    rows.forEach((row, index) => {
        console.log(row)
        codeElement.innerHTML += "    " + "[" + row + "],<br/>"
    })

    codeElement.innerHTML += "]<br />"
    console.log(header)
    let value2 = header[header.length - 1]
    console.log(value2)
    let header2 = []

    header.forEach((row, index) => {
        if (index != header.length - 1) {
            header2.push(row)
        }

    })

    testWithChart = () => {

        let dtSt = []
        //Insertar header que ya es un array de elementos


        rows.forEach((row, index) => {
            let row_array = []
            value = row.split(",")
```

```
    testWithChart = () => {

        rows.forEach((row, index) => {
            let row_array = []
            value = row.split(",")
            value.forEach((row2) => {
                row_array.push(row2.replace(/\r/g, ''))
            })

            dtSt.push(row_array)
        })

        console.log(dtSt)


        const trainObjective = document.getElementById('train-objective').value;
        let predict_values = []
        if (trainObjective === 'prediccion') {
            const predictionValue = document.getElementById('prediction-input').value;
            predict_values = predictionValue.toString().split(",")

        } else {
            rows.forEach((row, index) => {
                if (index == 1) {
                    value = row.split(",")
                    predict_values = value

                }
            })

        }
        console.log(header2)
        console.log(predict_values)

        let dTree = new DecisionTreeID3(dtSt);
        console.log(dTree)
        console.log(dTree.dataset)
        let root = dTree.train(dTree.dataset);
        let predict = dTree.predict([
            header2,
            predict_values
```

Esta función realiza la tendencia de los datos. Primero se calcula la pendiente para verificar si es mayor a cero, si así la tendencia es ascendente, caso contrario la tendencia es descendente. Luego se realiza la grafica para observar el resultado.

```
const Tendencias = () => {
    //xvalues y yvalues son del .csv
    if (X_train.length > 1 && y_train.length > 1) {
        const trendData = [];

        const pendiente = (y_train[y_train.length - 1] - y_train[0]) / (X_train[X_train.length - 1] - X_train[0]);
        const trendText = pendiente > 0 ? "La tendencia es ascendente." : "La tendencia es descendente.";

        alert(trendText)

        // Preparar los datos para el gráfico
        for (let i = 0; i < X_train.length; i++) {
            trendData.push([X_train[i], y_train[i]]);
        }

        google.charts.setOnLoadCallback(() => drawChartTendencia(trendData));
    } else {
        alert('No hay suficientes datos para determinar la tendencia.', 'error');
    }
};
```