Manual Técnico

INTELIGENCIA ARTIFICIAL 1

INDICE

RESPONSABLES

No.	Nombre	Carnet
2	Saúl Jafet Menchú Recinos	201906444

ACERCA DE

En este proyecto se detalla la implementación y funcionamiento de un programa en `JS` usando la libreria Tytusjs para el uso de Machine Learning.

CODIGO

A continuación se detalla parte del código y las funciones utlizadas dentro del mismo. Para está práctica todo el código fue realizado en lenguaje `JS` .

Carga de CSV

```
function procesarCSV(data) {
    const lines = data.split('\n');
    xData = [];
    yData = [];
    lines.forEach(line => {
        const [x, y] = line.split(',').map(Number);
        if (!isNaN(x) && !isNaN(y)) {
            xData.push(x);
            yData.push(y);
        }
    });
    document.getElementById("xTrain").value = xData.join(',');
    document.getElementById("yTrain").value = yData.join(',');
    console.log("Datos de X:", xData);
    console.log("Datos de Y:", yData);
}
```

Esta función permite carga los datos para su uso en los modelos *Lineal* y *Polinomial*. Los almacena en variables temporales y luego son usados para el entrenamiento y ejecución del mismo.

Regresión Lineal

```
function entrenarLineal() {
    let xTrain = document.getElementById("xTrain").value.split(',').map(Number);
    let yTrain = document.getElementById("yTrain").value.split(',').map(Number);
    let porcentaje = (Math.round(document.getElementById("porcentaje").value/10) * 10)/100;
    console.log(xTrain.length*porcentaje)

    let newXTrain = xTrain.slice(0,Math.round(xTrain.length*porcentaje))
    let newYTrain = yTrain.slice(0,Math.round(yTrain.length*porcentaje))

    console.log(newXTrain,newYTrain)

    linear = new LinearRegression();
    linear.fit(newXTrain, newYTrain);

    document.getElementById("log").innerHTML = 'Modelo entrenado exitosamente!';
}
```

En esta sección del codgio encontramos el entrenamiento del modelo de regresión lineal.

A continuación se muestra la función para la predicción lineal y cómo está se muestra en una gráfica.

```
function predecirLineal() {
    if (!linear) {
        document.getElementById("log").innerHTML = 'Por favor, entrene el modelo primero.';
        return;
    }
    let xTrain = document.getElementById("xTrain").value.split(',').map(Number);
    let porcentaje = (Math.round(document.getElementById("porcentaje").value/10) * 10)/100;
    let newXTrain = xTrain.slice(0,Math.round(xTrain.length*porcentaje))

    let yPredict = linear.predict(newXTrain);

    let chartData = [['X', 'Y Train', 'Y Predict']];
    let yTrain = document.getElementById("yTrain").value.split(',').map(Number);
    let newYTrain = yTrain.slice(0,Math.round(yTrain.length*porcentaje))
    for (let i = 0; i < newXTrain.length; i++) {
        chartData.push([newXTrain[i], newYTrain[i], yPredict[i]]);
    }
    google.charts.load('current', {'packages': ['corechart']});
    google.charts.setOnLoadCallback(function () {
        drawChart(chartData);
    });
}
```

Regresión Polinomial

En esta función encontramos el proceso de entreno para el modelo de regresión polinomial y la selección del grado.

```
function PolinomialEntrenar() {
            let xTrain = document.getElementById("polinomialXTrain").value.split(',').map(Numbe
            let yTrain = document.getElementById("polinomialYTrain").value.split(',').map(Numbe
            let degree = parseInt(document.getElementById("polinomialDegree").value);

            polinomialModel = new PolynomialRegression();
            polinomialModel.fit(xTrain, yTrain, degree);

            document.getElementById("polinomialLog").innerHTML = 'Modelo entrenado exitosamente
        }
```

Seguidamente el proceso para la selección de la predicción

```
function PolinomialPredecir() {
    if (!polinomialModel) {
        document.getElementById("polinomialLog").innerHTML = 'Por favor, entrene el modelo prime
        return;
    }

    let predictArray = document.getElementById("polinomialXTrain").value.split(',').map(Number);
    let yPredict = polinomialModel.predict(predictArray);

    // Redondear valores para mostrar en log
    for (let i = 0; i < yPredict.length; i++) {
        yPredict[i] = Number(yPredict[i].toFixed(2));
    }

    document.getElementById("polinomialLog").innerHTML += '<br>Predicciones: [' + yPredict + ']
    PolinomialGenerarGrafica(predictArray, yPredict);
}
```

K Means Neighbor

Por último, tenmos la creación de la tabla para el modelo K Means Neighbor

```
function KllenarTabla() {
    x2 = document.getElementById("x2").value.split(",").map(Number);
    y2 = document.getElementById("y2").value.split(",").map(Number);
    group = document.getElementById("group").value.split(",");
    point = [parseFloat(document.getElementById("pointX").value), parseFloat(document.getElement
}
```

Y el proceso de ejecución para el mismo.

```
function KejecutarKNN() {
    var individuals = zip([x2, y2, group]);
    var knn = new KNearestNeighbor(individuals);
    var euc = knn.euclidean(point);
    var man = knn.manhattan(point);

    document.getElementById("logE").innerHTML = "Distancia Euclidiana: " + euc;
    document.getElementById("logM").innerHTML = "Distancia Manhattan: " + man;
}
```