Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de ciencias Y sistemas
Laboratorio inteligencia artificial
Tutor académico: Erick Eden
Sandoval Ramirez

Manual Tecnico

Nombre	Carné
Jonatan Leonel Garcia Arana	202000424

1. Introducción al Proyecto

Este proyecto consiste en una plataforma web interactiva diseñada para implementar y evaluar distintos modelos de clasificación, como Regresión Logística, Redes Neuronales y Máquinas de Vectores de Soporte (SVM). La aplicación permite a los usuarios importar datasets, entrenar los modelos, hacer predicciones y analizar los resultados a través de gráficos. Está dirigida a académicos y profesionales que desean explorar técnicas de análisis de datos de manera accesible.

2. Requerimientos del Sistema

- **Hardware:**
    - Computadora de escritorio o portátil con procesador quad-core o superior.
    - Mínimo 8 GB de RAM.
- **Software:**
    - Navegador moderno (Safari, Opera, Chrome).
    - Librerías JavaScript requeridas:
        - Chart.js para visualizaciones gráficas.
        - TensorFlow.js para la implementación de modelos de aprendizaje automático.
        - Materialize CSS para el diseño de la interfaz.
        - D3.js para la visualización avanzada de datos.

3. Arquitectura del Sistema

La arquitectura consiste en un único documento HTML que integra la interfaz de usuario y la lógica para los modelos de clasificación en JavaScript. Los modelos son:

- **Regresión Lineal**
- **Regresión Polinomial**
- **Arboles de decisiones**

4. Descripción de Funcionalidades

- **Selección y Entrenamiento de Modelos:** Permite al usuario elegir el modelo y cargar datos desde un archivo Excel o introducirlos manualmente.
- **Predicción de Resultados:** Facilita la predicción de nuevos datos.
- **Visualización de Resultados:**
    - Gráficas de Comparación: Muestra las predicciones contra los datos originales.
    - Análisis de Datos: Gráficas que identifican patrones y tendencias.
    - Visualización de Redes Neuronales: Representación gráfica de la estructura de la red.
    -

5. Interfaz de Usuario

- **Panel de Entrenamiento:** Opción para seleccionar el modelo, ingresar datos y un botón para iniciar el entrenamiento.

- **Panel de Predicción:** Campos para introducir nuevos datos y un botón para realizar la predicción.
- **Panel de Resultados:** Sección que muestra las gráficas generadas y los resultados obtenidos.

6. Flujo del Usuario

1. **Carga de Datos:** El usuario puede cargar un archivo Excel o ingresar datos manualmente.
2. **Selección del Modelo:** Escoge el modelo a utilizar.
3. **Entrenamiento:** El sistema entrena el modelo seleccionado.
4. **Predicción:** El usuario ingresa nuevos datos para realizar predicciones.
5. **Visualización de Resultados:** Se presentan las gráficas y resultados de los modelos.

7. Manual de Instalación y Configuración

1. **Descargar Dependencias:**
   - Asegúrese de tener el archivo tensorflow.js en el mismo directorio que el archivo HTML.
2. **Abrir el Archivo HTML:**
   - Cargue el archivo HTML en un navegador compatible.
   - Acepte los permisos necesarios si hay advertencias de seguridad.

8. Casos de Uso

- **Educación:** Herramienta para estudiantes que desean aprender sobre modelos de clasificación.
- **Investigación Rápida:** Ideal para la creación de prototipos de modelos de clasificación con conjuntos de datos pequeños.

9. Solución de Problemas Comunes

- **No carga el Excel:** Verifique que el archivo esté en formato válido y que el navegador tenga acceso permitido.
- **Problemas de visualización:** Asegúrese de estar conectado a internet para cargar Chart.js y D3.js.
- **No se entrena el modelo:** Asegúrese de que todos los datos necesarios estén ingresados antes de iniciar el entrenamiento.

10. Mantenimiento

- **Actualización de Dependencias:** Mantenga actualizadas las librerías (tensorflow.js, Chart.js, D3.js) para asegurar la funcionalidad.
- **Seguridad del Navegador:** Configure adecuadamente los permisos de acceso a archivos en su navegador.

Lógica del sistema

Carga de archivos csv

```
function processCSV(data) {
        const rows = data.trim().split("\n");
        const trainingDataTable = document.getElementById("trainingDataTable");
        trainingDataTable.innerHTML = ""; // Clear existing data
        xTrain = [];
        yTrain = [];

        for (let i = 1; i < rows.length; i++) {
            const cols = rows[i].split(";");
            xTrain.push(parseFloat(cols[0]));
            yTrain.push(parseFloat(cols[1]));

            // Populate training data table
            const row = document.createElement("tr");
            row.innerHTML = `<td>${cols[0]}</td><td>${cols[1]}</td>`;
            trainingDataTable.appendChild(row);
        }
    }
```

Función Polinomial

```
function fitPolynomialModel() {
          var polynomial = new PolynomialRegression();

          // Fit and predict with different degrees
          polynomial.fit(window.xTrain, window.yTrain, 2);
          var yPredict = polynomial.predict(window.predictArray);
          var r2 = polynomial.getError();

          polynomial.fit(window.xTrain, window.yTrain, 3);
          var yPredict2 = polynomial.predict(window.predictArray);
          var r22 = polynomial.getError();

          polynomial.fit(window.xTrain, window.yTrain, 4);
          var yPredict3 = polynomial.predict(window.predictArray);
          var r23 = polynomial.getError();

          // Display predictions in the table
          const displayTable = document.getElementById("displayTable").getElementsByTagName("tbody")[0];
          displayTable.innerHTML = ""; // Clear existing predictions
          for (let i = 0; i < window.predictArray.length; i++) {
              const row = document.createElement("tr");
              const error = Math.abs((yPredict[i] - window.yTrain[i]) / window.yTrain[i]) * 100; // Calculate error percentage
              row.innerHTML = `<td>${window.predictArray[i]}</td><td>${yPredict[i]}</td><td>${error.toFixed(2)}%</td>`;
              displayTable.appendChild(row);
          }

          // Draw the trend charts
          drawTrendCharts(yPredict, yPredict2, yPredict3);
      }
```

Función Lineal

```
document.getElementById("processButton").addEventListener("click", function() {
    // Create and train the linear regression model
    const linear = new LinearRegression();
    linear.fit(xTrain, yTrain);
    yPredict = linear.predict(xTrain);

    // Update log and predicted data table
    document.getElementById("log").innerHTML = `
        <br>X Train: [${xTrain.join(", ")}]
        <br>Y Train: [${yTrain.join(", ")}]
        <br>Y Predicted: [${yPredict.join(", ")}]
    `;

    const predictedTable = document.getElementById("displayTable").getElementsByTagName("tbody")[0];
    predictedTable.innerHTML = ""; // Clear existing predicted data
    for (let i = 0; i < xTrain.length; i++) {
        const error = Math.abs(yTrain[i] - yPredict[i]) / yTrain[i] * 100;
        const row = document.createElement("tr");
        row.innerHTML = `<td>${xTrain[i]}</td><td>${yPredict[i]}</td><td>${error.toFixed(2)}%</td>`;
        predictedTable.appendChild(row);
    }

    document.getElementById("drawButton").style.display = 'inline'; // Show draw button
});
```