Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Ingeniería en Ciencias y Sistemas
Inteligencia Artificial I

Machine Learning con tytus.js

MANUAL TÉCNICO

Nombre: Cristian Aramis Lopez Bautista
Carnet: 201904042

ÍNDICE

Introducción

Este proyecto tiene como objetivo aplicar y profundizar en conceptos clave de Machine Learning mediante la implementación de algoritmos variados utilizando la biblioteca tytus.js. Se desarrolla como un sitio web alojado en GitHub Pages, construido únicamente con HTML y JavaScript, siguiendo el lineamiento de no usar frameworks externos, lo cual permite un mejor entendimiento de la lógica subyacente a cada modelo.

La interfaz de la aplicación facilita el proceso de carga y manipulación de datasets en formato CSV, permitiendo que los usuarios seleccionen diferentes modelos de Machine Learning para entrenar y probar los datos, con un enfoque visual que incluye gráficos interactivos generados con Google Charts y vis.js. Entre los modelos de aprendizaje automático disponibles se encuentran métodos de regresión (lineal y polinomial), modelos de clasificación y predicción como el árbol de decisión y Naive Bayes, y algoritmos de clustering como KMeans.

Cada uno de estos modelos ofrece parametrización ajustable, como el porcentaje de datos destinados a entrenamiento y prueba, la selección de variables de entrada y salida para aprendizaje supervisado, y otros parámetros específicos de cada algoritmo. Además, la aplicación permite realizar predicciones y visualizar resultados en tiempo real, lo que proporciona una herramienta práctica y educativa tanto para estudiantes como para entusiastas de la inteligencia artificial que desean experimentar con distintos enfoques de Machine Learning en un entorno web accesible.

Requisitos del Sistema

Lenguajes y Bibliotecas:

- HTML5
- JavaScript puro (sin frameworks)
- Bootstrap (para el diseño visual)
- Biblioteca tytus.js (para algoritmos de Machine Learning)
- Google Charts y vis.js (para visualización de gráficos)

Entorno de Desarrollo: GitHub Pages

Formato de Archivos: Soporte para archivos CSV como datasets de entrada.

Estructura del Proyecto

- **index.html**: Página principal que contiene todos los elementos de configuración y ejecución de los modelos de Machine Learning.
- **tytus.js**: Como librería que contiene toda la lógica para la resolución de predicciones y entrenamientos de inteligencia artificial.
- **sidebar**: Incluye enlaces para seleccionar el modelo de ML deseado.
- **header-container**: Contiene el título y encabezado de la aplicación.
- **main-container**: Sección principal donde se cargan y muestran los modelos, configuraciones y visualización de datos.

Descripción de Componentes

Selección de Modelo:

El sistema ofrece una interfaz intuitiva a través de un menú lateral desde el cual el usuario puede seleccionar entre diversos modelos de Machine Learning, incluidos: Regresión Lineal, Regresión Polinomial, Árbol de Decisión, Naive Bayes, Redes Neuronales y KMeans. Cada modelo implementado permite al usuario ajustar una serie de parámetros específicos de entrada, lo cual facilita la manipulación personalizada tanto para el entrenamiento como para la predicción.

1. **Regresión Lineal y Polinomial**:
    - Estos modelos están diseñados para tareas de predicción de tendencias. El usuario puede seleccionar el tipo de regresión deseada (lineal o polinomial) y ajustar el porcentaje de datos dedicados a entrenamiento versus prueba.
    - Para la regresión polinomial, el usuario puede especificar el grado del polinomio, permitiendo una mayor flexibilidad en la modelación de relaciones no lineales.

2. **Árbol de Decisión**:
    - Utilizando el algoritmo ID3, el Árbol de Decisión permite al usuario cargar un dataset en formato CSV y visualizar la estructura del árbol.
    - Se incluyen campos específicos para ingresar los valores de prueba, lo que permite hacer predicciones basadas en datos específicos proporcionados por el usuario.
    - La visualización del árbol se genera dinámicamente, facilitando una comprensión visual de las decisiones de cada nodo y cómo se estructuran las ramas.
3. **Naive Bayes**:
    - Este modelo es ideal para tareas de clasificación. El usuario selecciona las columnas relevantes y especifica valores de entrada para realizar predicciones de clase basadas en las probabilidades calculadas.
    - Los datos de entrenamiento pueden ser cargados desde un archivo CSV y se muestra una tabla con los valores previos para referencia.
4. **Redes Neuronales**:
    - Este modelo permite realizar predicciones en función de operaciones probabilísticas. El usuario puede cargar un dataset y seleccionar columnas específicas para el análisis.
    - Se proporcionan opciones para ajustar la estructura de la red neuronal, como la elección de columnas de entrada, y el tipo de operación (mayor, menor o diferente) que se desea modelar.
5. **KMeans (Clustering)**:
    - El modelo KMeans está disponible en dos variantes: KMeans Lineal y KMeans 2D.
    - Para KMeans Lineal, el usuario puede ingresar datos numéricos y ajustar la cantidad de clusters y el número de iteraciones para el proceso de agrupamiento.
    - En el caso de KMeans 2D, se permite la entrada de coordenadas 2D para analizar datos en dos dimensiones, con controles para la cantidad de clusters y las iteraciones necesarias.

Cada modelo se presenta con opciones específicas de parametrización y visualización de datos, adaptadas para proporcionar una experiencia de usuario dinámica. Además, la interfaz muestra los datos cargados, la selección de variables de entrada y salida, y las gráficas resultantes, lo que facilita una experimentación más comprensible y permite un análisis visual de los resultados de cada modelo.

Entrenamiento y Predicción:

- Cada modelo incluye botones específicos para realizar el entrenamiento y efectuar predicciones sobre el dataset cargado.

Visualización de Datos:

- **Google Charts** y **vis.js** son utilizados para visualizar los datos y resultados generados por cada modelo.
- Los gráficos están diseñados para mostrar tanto los datos originales como las predicciones generadas por los modelos.

Estructura del Código

JavaScript:

- **Función `mostrarVista()`**: permite mostrar/ocultar el contenido de cada modelo según la selección del usuario.

```javascript
// Función para mostrar y ocultar vistas según el modelo seleccionado
function mostrarVista(vistaId) {
    document.getElementById('lineal').style.display = 'none';
    document.getElementById('polinomial').style.display = 'none';
    document.getElementById('parametrizacion').style.display = 'none';
    document.getElementById('lineal').style.display = 'none';
    document.getElementById('polinomial').style.display = 'none';
    document.getElementById('decision-tree').style.display = 'none';
    document.getElementById('naive-bayes').style.display = 'none';
    document.getElementById('newronal').style.display = 'none';
    document.getElementById('seccionkmeans').style.display = 'none';

    if (vistaId === 'lineal') {
        if (document.getElementById(vistaId)) {
            document.getElementById(vistaId).style.display = 'block';
            document.getElementById('parametrizacion').style.display = 'block';
        }
    } else if (vistaId === 'polinomial') {
        if (document.getElementById(vistaId)) {
            document.getElementById(vistaId).style.display = 'block';
            document.getElementById('parametrizacion').style.display = 'block';
        }
    } else if (document.getElementById(vistaId)) {
        document.getElementById(vistaId).style.display = 'block';
    }


}
```

- **Función `cargarDataset()`**: gestiona la carga y preprocesamiento de archivos CSV.

```
function cargarDataset() {
    const csvFile = document.getElementById('csvFile').files[0];
    if (!csvFile) {
        alert("Por favor, selecciona un archivo CSV.");
        return;
    }

    const reader = new FileReader();
    reader.onload = function (event) {
        const text = event.target.result;
        procesarCSV(text);
    };
    reader.readAsText(csvFile);
}
```

- **Funciones de entrenamiento y predicción**: cada modelo tiene una función dedicada para entrenar y predecir, como `entrenarModelo()` para entrenamiento y `realizarOperacion()` para predicción.

```
function entrenarModelo(tipo) {
    const xData = modelo.variablesEntrada;
    const yData = modelo.variableSalida;
    const trainSize = Math.floor((modelo.porcentajeTrain / 100) * xData.length);
    const xTrain = xData.slice(0, trainSize);
    const yTrain = yData.slice(0, trainSize);

    if (tipo === 'lineal') {
        modelo.entrenado = new LinearRegression();
        modelo.entrenado.fit(xTrain, yTrain);
    } else if (tipo === 'polinomial') {
        modelo.entrenado = new PolynomialRegression();
        modelo.entrenado.fit(xTrain, yTrain, modelo.gradoPolinomio);
    } else if (tipo === 'decisionTree') {
        modelo.entrenado = new DecisionTreeID3([xData, yData]);
        modelo.entrenado.train();
    } else if (tipo === 'naiveBayes') {
        modelo.entrenado = new NaiveBayes();
        modelo.entrenado.train(xTrain, yTrain);
    }

    alert(`Modelo de ${tipo} entrenado con éxito.`);
}
```

```
function realizarOperacion(tipo) {
    let yPred;

    if (tipo === 'lineal' || tipo === 'polinomial') {
        const xValores = modelo.objetivo === 'prediccion' && document.getElementById('rangoPrediccion').value
            ? document.getElementById('rangoPrediccion').value.split(',').map(Number)
            : modelo.variablesEntrada;

        yPred = modelo.entrenado.predict(xValores);
        dibujarGrafica(tipo === 'lineal' ? 'graficaLineal' : 'graficaPolinomial', modelo.variablesEntrada, modelo.variableSalida, yPred, tipo === 'lineal' ? 'Re
    } else if (tipo === 'decisionTree') {
        const xValores = modelo.variablesEntrada;
        yPred = modelo.entrenado.predict(xValores);
        document.getElementById('graficaDecisionTree').innerHTML = `Predicción: ${yPred}`;
    } else if (tipo === 'naiveBayes') {
        const xValores = modelo.variablesEntrada;
        yPred = modelo.entrenado.predict(xValores);
        document.getElementById('resultadoNaiveBayes').innerHTML = `Predicción: ${yPred}`;
    }
}
```

- **Visualización de gráficos**: `dibujarGrafica()` crea gráficos con Google Charts para mostrar datos de entrada y predicciones.

```
function dibujarGrafica(elementId, xData, yData, yPred, title) {
    google.charts.load('current', { 'packages': ['corechart'] });
    google.charts.setOnLoadCallback(() => {
        const data = new google.visualization.DataTable();
        data.addColumn('number', 'X');
        data.addColumn('number', 'Original');
        data.addColumn('number', 'Predicción');

        const filas = xData.map((x, i) => [x, yData[i], yPred[i]]);
        data.addRows(filas);

        const options = {
            title: title,
            hAxis: { title: 'X' },
            vAxis: { title: 'Y' },
            seriesType: 'scatter',
            series: { 1: { type: 'line' } }
        };

        const chart = new google.visualization.ComboChart(document.getElementById(elementId));
        chart.draw(data, options);
    });
}
```

Configuración de Modelos

Cada modelo incluye configuraciones específicas:

- **Regresión Lineal y Polinomial**: permite configurar el grado del polinomio y otros parámetros.
- **Árbol de Decisión**: usa el algoritmo ID3 para construir y visualizar el árbol de decisión.
- **Naive Bayes**: permite la selección de variables para predecir clases basadas en probabilidades.
- **KMeans**: soporta tanto KMeans Lineal como KMeans 2D, con opciones de configuración para iteraciones y cantidad de clusters.

Conclusión

Este manual técnico describe en detalle los elementos y funcionalidades del proyecto de Machine Learning, proporcionando una referencia completa para comprender y utilizar cada componente de la aplicación. Desde la selección de modelos y parametrización hasta la visualización y análisis de resultados, cada sección ha sido diseñada para ofrecer al usuario una experiencia práctica e intuitiva en la exploración de algoritmos de aprendizaje automático.

Además de facilitar el uso efectivo de la biblioteca tytus.js, este manual guía al usuario en el ajuste de los modelos y en la interpretación de sus salidas, permitiéndole experimentar con configuraciones personalizadas y visualizar el comportamiento de los algoritmos en tiempo real. Con esta herramienta, se espera que el usuario pueda profundizar en su comprensión de los modelos de Machine Learning y sus aplicaciones, promoviendo un aprendizaje práctico y accesible.