

# Intérprete DSL Deep Learning

Un intérprete completo para un lenguaje específico de dominio (DSL) orientado al aprendizaje automático y análisis de datos.

## Características

- **Operaciones matemáticas:** Aritmética básica, funciones trigonométricas, logaritmos
- **Matrices:** Transposición, multiplicación, suma, resta, inversión
- **Machine Learning:** Regresión lineal, clasificadores MLP, K-means clustering
- **Visualización:** Gráficos de líneas, dispersión, histogramas
- **Control de flujo:** Condicionales (if-else), bucles (while)
- **Archivos:** Lectura y escritura de archivos CSV y texto
- **Interfaz interactiva:** REPL con comandos especiales

## Requisitos

- Python 3.6+
- ANTLR4 para Python
- Archivos generados por ANTLR4:
  - `DeepLearningDSLLexer.py`
  - `DeepLearningDSLParser.py`
  - `DeepLearningDSLVisitor.py`

## Instalación

1. Instala ANTLR4 para Python:

```
bash
```

```
pip install antlr4-python3-runtime
```

2. Genera los archivos ANTLR4 (si no los tienes):

```
bash
```

```
antlr4 -Dlanguage=Python3 -visitor DeepLearningDSL.g4
```

3. Coloca todos los archivos en el mismo directorio:

- `main.py` (intérprete principal)
- `DSLInterpreterVisitor.py` (visitor personalizado)
- Archivos generados por ANTLR4
- `ejemplo.dsl` (ejemplos de código)



## Uso

### Modo Interactivo

```
bash
```

```
python main.py
```

Esto iniciará el REPL interactivo donde puedes escribir comandos DSL:

```
🚀 DSL Deep Learning - Intérprete Interactivo
Escribe '.help' para ver ayuda o '.exit' para salir
```

```
>>> x = 5;
```

```
📝 x = 5
```

```
✅ Ejecutado correctamente
```

```
>>> y = x * 2 + 3;
```

```
📝 y = 13
```

```
✅ Ejecutado correctamente
```

```
>>> .vars
```

```
📝 Variables definidas:
```

```
-----
x = 5
```

```
y = 13
-----
```

### Ejecutar Archivo

```
bash
```

```
python main.py ejemplo.dsl
```

### Comandos Especiales del REPL

- `.help` - Muestra ayuda completa
- `.vars` - Lista todas las variables definidas
- `.history` - Muestra historial de comandos
- `.clear` - Limpia variables e historial
- `.exit` - Sale del intérprete

## Sintaxis del Lenguaje

### Variables y Asignaciones

javascript

```
x = 5;
nombre = "Juan";
lista = [1, 2, 3, 4, 5];
```

### Operaciones Matemáticas

javascript

```
resultado = x + 5 * 2;
seno = sin(3.14159);
raiz = sqrt(25);
potencia = x ^ 2;
```

### Matrices

javascript

```
matriz = [[1, 2], [3, 4]];
transpuesta = transpose(matriz);
inversa = inverse(matriz);
producto = matmul(matriz, matriz);
suma_mat = matsum(matriz, matriz);
```

### Estructuras de Control

javascript

```
if x > 0 then
    resultado = "Positivo";
else
    resultado = "No positivo";
fi
```

```
contador = 0;
while contador < 10 do
    contador = contador + 1;
done
```

## Machine Learning

javascript

```
// Regresión lineal
X = [1, 2, 3, 4, 5];
y = [2, 4, 6, 8, 10];
modelo = linearRegression(X, y);

// Clasificación
features = [[1, 2], [2, 3], [3, 4]];
labels = [0, 1, 1];
clasificador = mlpClassifier(features, labels, [10, 5]);

// Clustering
datos = [1, 2, 3, 8, 9, 10];
clusters = kmeans(datos, 2);
```

## Visualización

javascript

```
plot(x_values, y_values); // Gráfico de líneas
scatter(x_data, y_data); // Gráfico de dispersión
hist(data); // Histograma
```

## Operaciones con Archivos

javascript

```
datos = readFile("archivo.csv");
writeFile("salida.csv", matriz);
```

## Ejemplos Completos

### Análisis de Datos Básico

javascript

```
// Cargar datos
datos = [[1, 2], [2, 4], [3, 6], [4, 8], [5, 10]];
x_vals = [1, 2, 3, 4, 5];
y_vals = [2, 4, 6, 8, 10];

// Entrenar modelo
modelo = linearRegression(x_vals, y_vals);

// Visualizar
plot(x_vals, y_vals);
scatter(x_vals, y_vals);
```

## Procesamiento de Matrices

javascript

```
// Crear matrices
A = [[1, 2, 3], [4, 5, 6]];
B = [[1, 4], [2, 5], [3, 6]];

// Operaciones
C = matmul(A, B);
A_T = transpose(A);

// Matriz cuadrada para inversión
cuadrada = [[2, 1], [1, 2]];
inversa = inverse(cuadrada);
```

## Clustering de Datos

javascript

```
// Datos de ejemplo
datos_1d = [1, 2, 3, 10, 11, 12, 20, 21, 22];
resultado = kmeans(datos_1d, 3);

// Visualizar distribución
hist(datos_1d);
```

## Extensión del Intérprete

El intérprete está diseñado para ser extensible. Para agregar nuevas funcionalidades:

1. **Nuevas operaciones:** Modifica `DSLInterpreterVisitor.py`
2. **Nuevas palabras clave:** Actualiza la gramática `DeepLearningDSL.g4`
3. **Nuevos tipos de datos:** Extiende los métodos auxiliares del visitor







## Manejo de Errores

El intérprete maneja varios tipos de errores:

- **Errores de sintaxis:** Detectados por ANTLR4
- **Variables no definidas:** Verificación en tiempo de ejecución
- **Errores de tipo:** Conversiones automáticas cuando es posible
- **Errores matemáticos:** División por cero, raíces negativas, etc.
- **Errores de archivos:** Archivos no encontrados, permisos, etc.

## Salida del Intérprete

El intérprete proporciona salida informativa:

-  Confirmación de ejecución exitosa
-  Asignaciones de variables
-  Resultados de expresiones
-  Información de modelos ML entrenados
-  Detalles de visualizaciones
-  Confirmación de operaciones de archivos

## Personalización

Puedes personalizar el comportamiento del intérprete modificando:

- Formatos de salida en `_format_value()`
- Precisión numérica en las operaciones matemáticas
- Implementaciones de algoritmos ML
- Estilos de visualización ASCII

## Modo Debug

Ejecuta con `--debug` para ver información detallada de errores:

```
bash
```

```
python main.py --debug
```

## Notas Importantes

- Todos los statements deben terminar con (;)
- Las variables son dinámicamente tipadas
- Los algoritmos de ML son implementaciones simplificadas para demostración
- Las visualizaciones se muestran como texto ASCII
- Los archivos se leen/escriben en formato UTF-8

¡Disfruta explorando el mundo del machine learning con este DSL! 🚀