Documentación: Creacion de proyectos de inteligencia artificial utilizando diferentes lenguajes

La inteligencia artificial es un campo que día a día va aumentado y mejorando cada vez más, al desarrollar estos proyectos se puede comprender como funcionan algunos lenguajes de programación a la hora de utilizar funciones para inteligencia artificial. En este proyecto se han creado dos clasificadores de vinos utilizando dos lenguajes de programación muy interesantes que son: Python y Julia. La finalidad del clasificador es predecir la clase de vino en base a sus características químicas, utilizando el dataset Wine que contiene alrededor de 178 muestras de vino cada una con 13 características químicas diferentes y a su vez están divididas en 3 clases. A continuación, podremos explorar los resultados de ambas implementaciones.

Implementación en Python

El primer lenguaje utilizado para este proyecto fue Python. Es un lenguaje de alto nivel, interpretado y de código abierto, se centra en la simplicidad y legibilidad del código lo que permite que desarrolladores sin practica o profesionales puedan implementar sus soluciones de forma sencilla.

Para este proyecto, se utilizo la biblioteca “scikit-learn”, que brinda un conjunto completo de herramientas para el aprendizaje automático y es la que hará posible la realización del clasificador en Python. Las funciones utilizadas fueron las siguientes:

- **Wine**: Importa el conjunto de datos que vamos a utilizar.
- **Train_test_split:** Permite que se puedan organizar los datos en función de para que se utilizara, ya sea entrenamiento o prueba.
- **DecisionTreeClassifier:** Utilizada para crear el modelo de árbol de decisiones que se entrenara.
- **Accuray_score y Classification_report:** Utilizadas para la evolución y realización y realización de un reporte con los resultados de las predicciones.

En la siguiente foto podemos observar el proceso de importar las herramientas necesarias:

Proyectos IA > Clasificador_IA.py > ...

```
1  #Primero importamos los recursos necesarios para realizar el proyecto
2  from sklearn.datasets import load_wine
3  from sklearn.model_selection import train_test_split
4  from sklearn.tree import DecisionTreeClassifier
5  from sklearn.metrics import accuracy_score,classification_report
```

Una vez terminamos de importar las herramientas necesarias, comencé con el procesamiento de los datos que, se dividen en datos de entrenamiento y datos de prueba en términos de 70 – 30, luego se crea una instancia del árbol de decisiones y se entrena el modelo usando los datos de entrenamiento y para finalizar se realiza una pequeña función que se encargue de medir que tan efectivo fue el entrenamiento y se imprimen los resultados obtenidos.

En la siguiente imagen podemos apreciar esto:

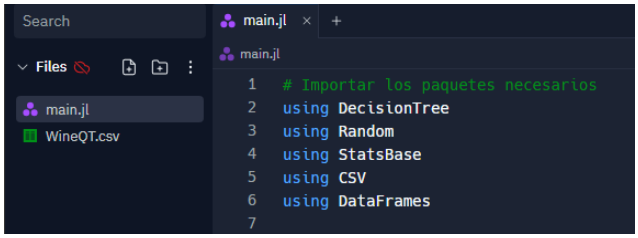
```
#Cargamos los datos
vinos = load_wine()
x_entrenamiento, x_prueba, y_entrenamiento, y_prueba = train_test_split(vinos.data, vinos.target, test_size=0.3, random_state=40)

#Creamos el modelo de clasificacion, que sera un arbol de decisiones y empezamos el entrenamiento
modeloClasificador = DecisionTreeClassifier()
modeloClasificador.fit(x_entrenamiento, y_entrenamiento)

#Realizamos la prediccion con los datos de prueba y luego vemos la precision del mismo
prediccion = modeloClasificador.predict(x_prueba)
print("Presicion del modelo:", accuracy_score(y_prueba,prediccion))
print("\n", classification_report(y_prueba, prediccion, target_names=vinos.target_names)) #Muestra un reporte del entrenamiento
```

Implementación en Julia

El siguiente lenguaje que utilizamos para la implementación de este proyecto fue Julia. Es un lenguaje de programación de alto nivel, diseñado especialmente para realizar calculo científico y análisis de datos. Destaca por tener una gran capacidad para manejar cálculos numéricos complejos y grandes volúmenes de datos de forma eficiente. Caracterizado por su alto rendimiento, sintaxis fácil de aprender y parecida a la de Python y es de tipado dinámico, por lo que, permite definir tipos de datos de forma flexible.

Para la realización del clasificador de vinos en Julia se intento replicar la misma lógica utilizada en el proyecto de Python, para que la comparación fuera lo mas acercada posible. Para ello se implementaron varias librerías que fueron de mucha utilidad a la hora de realizar la creación del código, además de que para este proyecto el dataset se descargó de forma externa (Google).

En la siguiente imagen se puede observar como se asignan todas las librerías a utilizar y vemos el dataset que ha sido colocado en el espacio de trabajo:



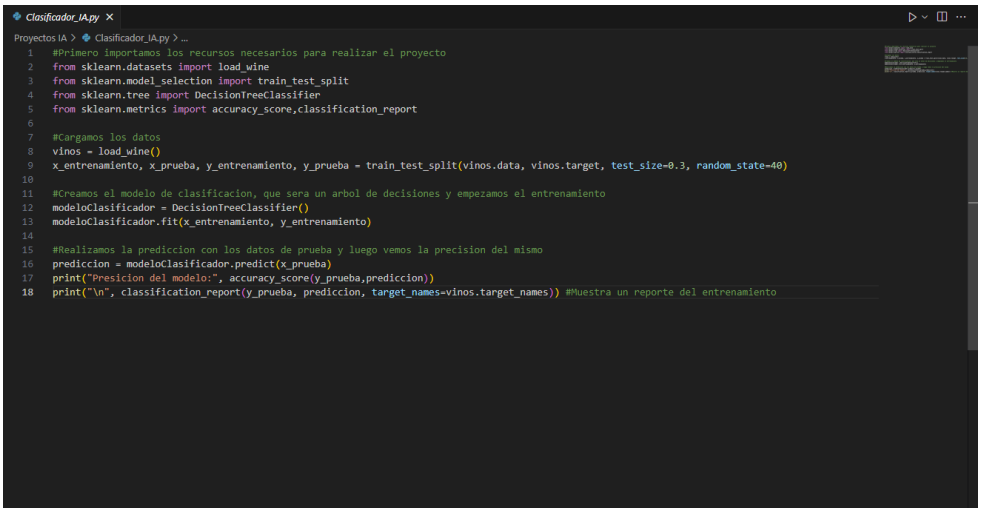
Luego se realiza el mismo proceso que en Python, se cargan los datos y se preparan para el entrenamiento. Luego se dividen en datos de entrenamiento y datos de prueba con una proporción de 70 – 30 respectivamente. Luego se realiza la creación del modelo de árbol de decisiones para que este sea entrenado por medio de los datos de entrenamiento y luego de este proceso se realiza una función la cual se encarga de realizar predicciones en base a los datos de prueba para determinar la precisión del modelo y estos resultados se imprimen en consola.

Estos procedimientos se pueden observar en la siguiente imagen:

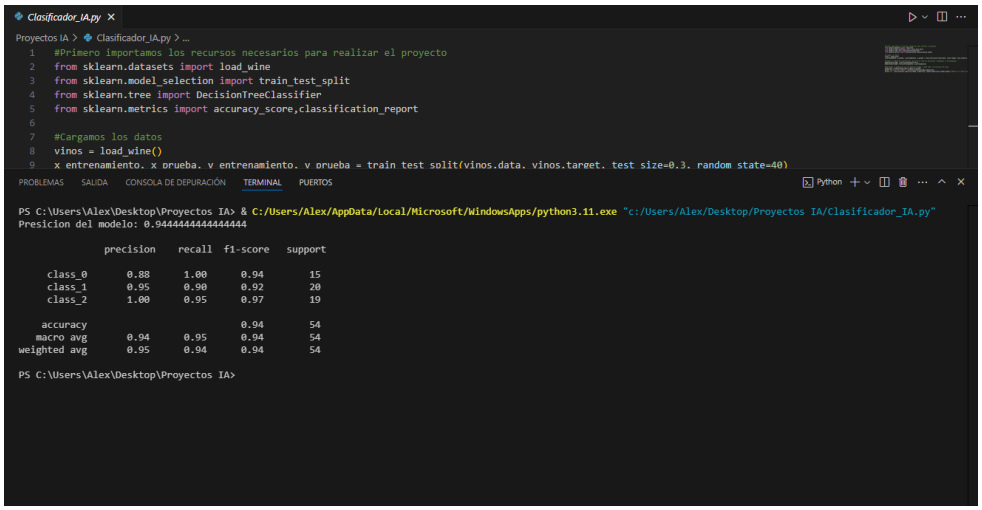
```
# Cargar el conjunto de datos desde un archivo CSV
vinos = CSV.read("WineQT.csv", DataFrame)
x = Matrix(vinos[:, 2:end])
y = Vector(vinos[:, 1])

# Dividir los datos en conjuntos de entrenamiento y prueba
Random.seed!(40)
n = size(x, 1)
train_ind = sample(1:n, round(Int, 0.7 * n), replace=false)
test_ind = setdiff(1:n, train_ind)

x_entrenamiento = x[train_ind, :]
y_entrenamiento = y[train_ind]

x_prueba = x[test_ind, :]
y_prueba = y[test_ind]

# Crear y entrenar el modelo de árbol de decisiones
modeloClasificador = DecisionTreeClassifier()
DecisionTree.fit!(modeloClasificador, x_entrenamiento, y_entrenamiento)

# Realizar predicciones y calcular la precisión
prediccion = DecisionTree.predict(modeloClasificador, x_prueba)

# Evaluar la precisión del modelo
precision = mean(prediccion .== y_prueba)
println("Precisión del modelo: $precision")
```

Imágenes del código y resultados

Código del proyecto en Python

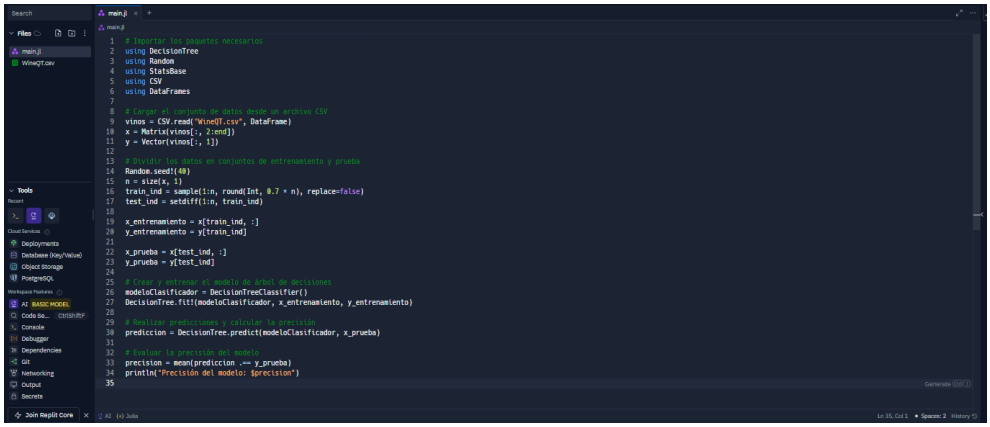


Resultados obtenidos



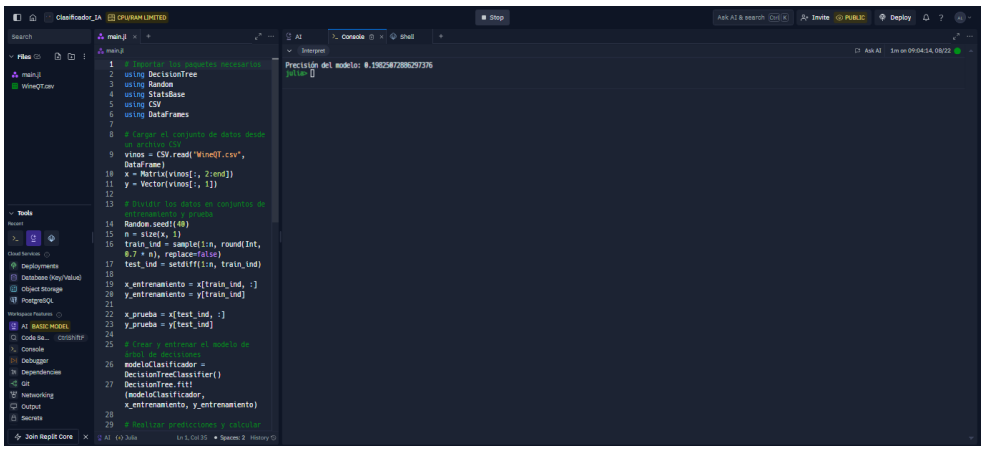
```
Clasificador_IA.py

#Primero importamos los recursos necesarios para realizar el proyecto
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,classification_report

#Cargamos los datos
vinos = load_wine()
x_entrenamiento, x_prueba, y_entrenamiento, y_prueba = train_test_split(vinos.data, vinos.target, test_size=0.3, random_state=40)

PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS

PS C:\Users\Alex\Desktop\Proyectos IA> & C:/Users/Alex/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/Alex/Desktop/Proyectos IA/Clasificador_IA.py"
Presicion del modelo: 0.9444444444444444

              precision    recall  f1-score   support

     class_0       0.88      1.00      0.94        15
     class_1       0.95      0.90      0.92        20
     class_2       1.00      0.95      0.97        19

    accuracy                           0.94        54
   macro avg       0.94      0.95      0.94        54
weighted avg       0.95      0.94      0.94        54

PS C:\Users\Alex\Desktop\Proyectos IA>
```

Código del proyecto en Julia



```
# Importar los paquetes necesarios
using DecisionTree
using Random
using StatsBase
using CSV
using DataFrames

# Cargar el conjunto de datos desde un archivo CSV
vinos = CSV.read("WineQT.csv", DataFrame)
x = Matrix(vinos[:, 2:end])
y = Vector(vinos[:, 1])

# Dividir los datos en conjuntos de entrenamiento y prueba
Random.seed!(40)
n = size(x, 1)
train_ind = sample(1:n, round(Int, 0.7 * n), replace=false)
test_ind = setdiff(1:n, train_ind)

x_entrenamiento = x[train_ind, :]
y_entrenamiento = y[train_ind]

x_prueba = x[test_ind, :]
y_prueba = y[test_ind]

# Crear y entrenar el modelo de árbol de decisiones
modeloClasificador = DecisionTreeClassifier()
DecisionTree.fit!(modeloClasificador, x_entrenamiento, y_entrenamiento)

# Realizar predicciones y calcular la precisión
prediccion = DecisionTree.predict(modeloClasificador, x_prueba)

# Evaluar la precisión del modelo
precision = mean(prediccion .== y_prueba)
println("Precisión del modelo: $precision")
```

Resultados obtenidos



```
Precisión del modelo: 0.9629629629629629
julia>
```

En teoría la lógica implementada para ambos proyectos es la misma, esto para lograr una comparación mas clara y como se puede observar el resultado en Python es mas aceptable que el de julia, en Python se obtuvo una precisión del 94% y en Julia una del 19%. Pero esto no hace a Python superior o no, simplemente puede que la lógica implementada en Julia no fuera la correcta o los datos están siendo mal administrados, lo que puede provocar un sobreajuste y por esta razón los resultados son tan bajos.

Comparación del proceso en cada lenguaje

La implementación del proyecto en el lenguaje Python fue bastante sencilla, ya que Python ofrece buena documentación sobre los procesos necesarios para utilizar las diferentes librerías y las herramientas que esta ofrece, además de que su sintaxis es bastante sencilla de entender y muy intuitiva. Por otra parte, estoy algo familiarizado con el funcionamiento de Python por lo que, resulto ser un proceso bastante sencillo y rápido.

En cambio, con julia fue algo diferente, este lenguaje ofrece un alto rendimiento y es ideal para cálculos numéricos, pero al no tener mucha experiencia con este lenguaje, presento una dificultad extra para la realización del proyecto en este lenguaje. Sin embargo, la curva de aprendizaje que presenta se puede reducir un poco debido a que su sintaxis es parecida a la de Python. La implementación del proyecto costo un poco mas puesto que se presentaron algunos problemas con una librería, encargada de importar el dataset wine, al momento de utilizarla no encontraba el archivo en la ruta especificada y para solucionar eso pues descargue el dataset de forma externa, esta implementación si tomo un poco más de tiempo del que se esperaba.

Reflexión personal

Desde mi punto de vista prefiero utilizar el lenguaje de Python para la realización de proyectos de inteligencia artificial, puesto que estoy familiarizado con el lenguaje de programación y algunas de las librerías, las cuales brindan herramientas super importantes para la realización de estos proyectos. Además de que la sintaxis de Python es muy intuitiva por lo que no es tan difícil acostumbrarse o aprender como utilizarla. Por otra parte, cuenta con bastante documentación que facilita la realización de las diferentes actividades o proyectos de inteligencia artificial, considero que es un lenguaje bastante completo a la hora de ser utilizado para implementaciones de inteligencia artificial.

Para finalizar, luego de la realización de este proyecto hemos explorado algunos lenguajes y se han utilizado dos lenguajes interesantes para el desarrollo de inteligencia artificial, tanto Python como Julia son lenguajes muy buenos y que se pueden utilizar para estos proyectos de forma efectiva, pero se debe tener en cuenta el contexto en el que se vaya a utilizar cada lenguaje. Python es un buen lenguaje para uso general y para realizar algunas operaciones de Inteligencia artificial de forma efectiva. Por otra parte, Julia es un lenguaje que ofrece potencia a la hora de realizar diferentes cálculos matemáticos, por lo que se debe tener en cuenta el contexto en el que se vaya a utilizar cada implementación y tomar la decisión de cual utilizar en base a los requerimientos del proyecto.