Curso de Especialización en Inteligencia Artificial y BigData

Programación de Inteligencia Artificial

Tarea Evaluable 3.1

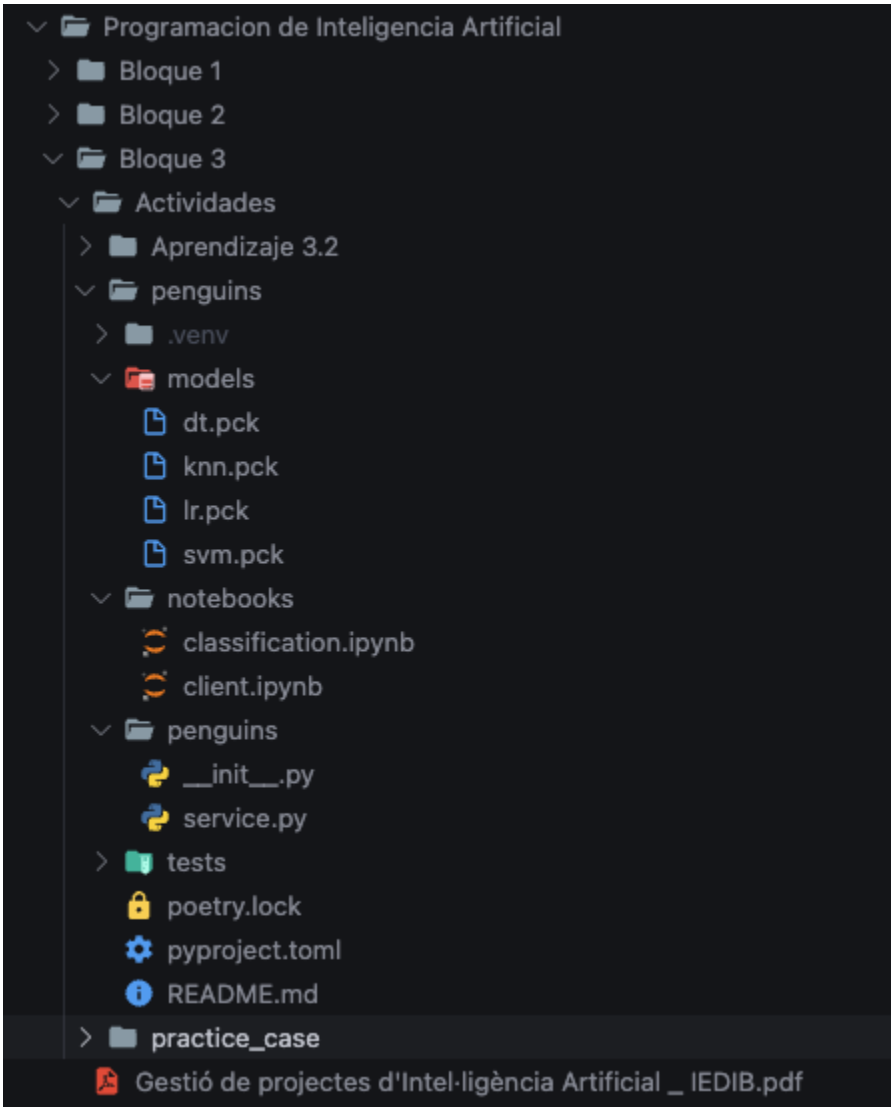
Autor: Carlos Sánchez Recio.
30 / 11 / 2024

Enlace del repositorio de GitHub con el proyecto.

https://github.com/CharlyMech/IEDIB_CEIA_2024-25/tree/main/Programacion%20de%20Inteligencia%20Artificial/Bloque%203/Actividades/penguins

Como se puede ver por la ruta al proyecto en GitHub, éste se encuentra alojado en un repositorio destinado a alojar todo lo realizado durante el curso de especialización.

Contenido del proyecto



Contenido del archivo `pyproject.toml`

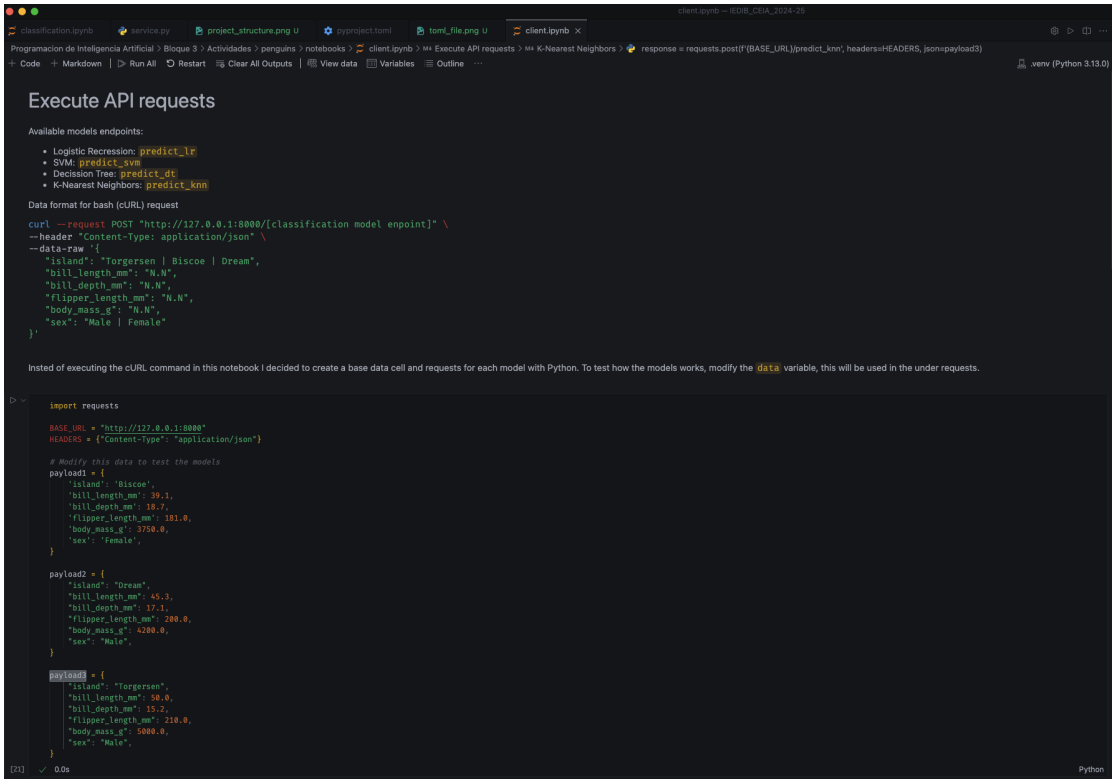
Programacion de Inteligencia Artificial > Bloque 3 > Actividades > penguins > pyproject.toml

You, 1 hour ago | 1 author (You)

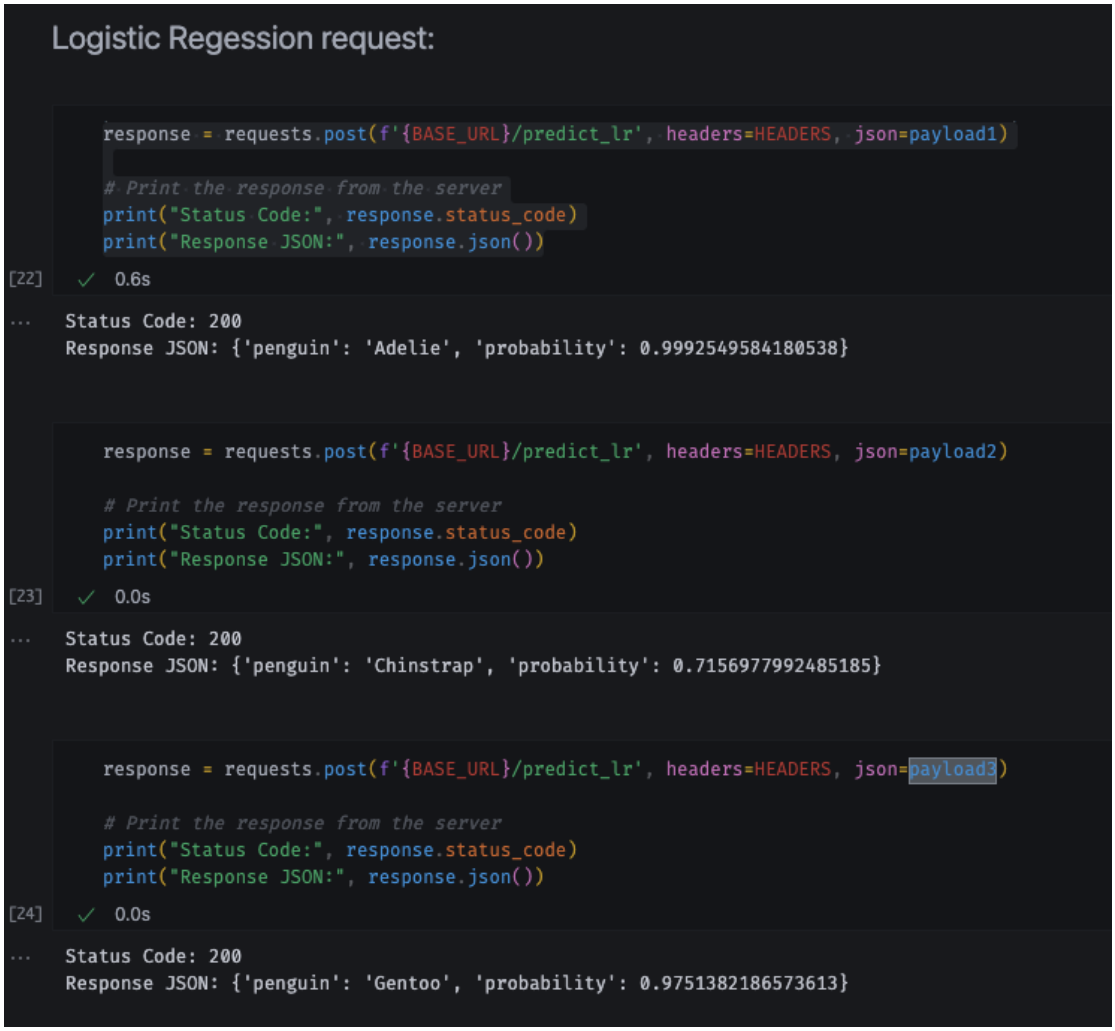
```toml
[tool.poetry]
name = "penguins"
version = "0.1.0"
description = ""
authors = ["CharlyMech <sanchezreciocarlos99@gmail.com>"]
readme = "README.md"

[tool.poetry.dependencies]
python = "^3.13"
flask = "^3.1.0"
scikit-learn = "^1.5.2"
ipykernel = "^6.29.5"
seaborn = "^0.13.2"
requests = "^2.32.3"
    You, 1 hour ago • feat: penguins evaluated task v1, pending revis…

[build-system]
requires = ["poetry-core"]
build-backend = "poetry.core.masonry.api"
```

Peticiones a cada modelo

Para las peticiones he decidido utilizar el paquete de **`requests`** de Python ya que mediante el uso de **cURL** en los notebooks siempre aparecía un error aunque se ejecutara correctamente. Además que de esta forma puedo centralizar los datos de la siguiente forma (consultar en el repositorio para mejor contexto):

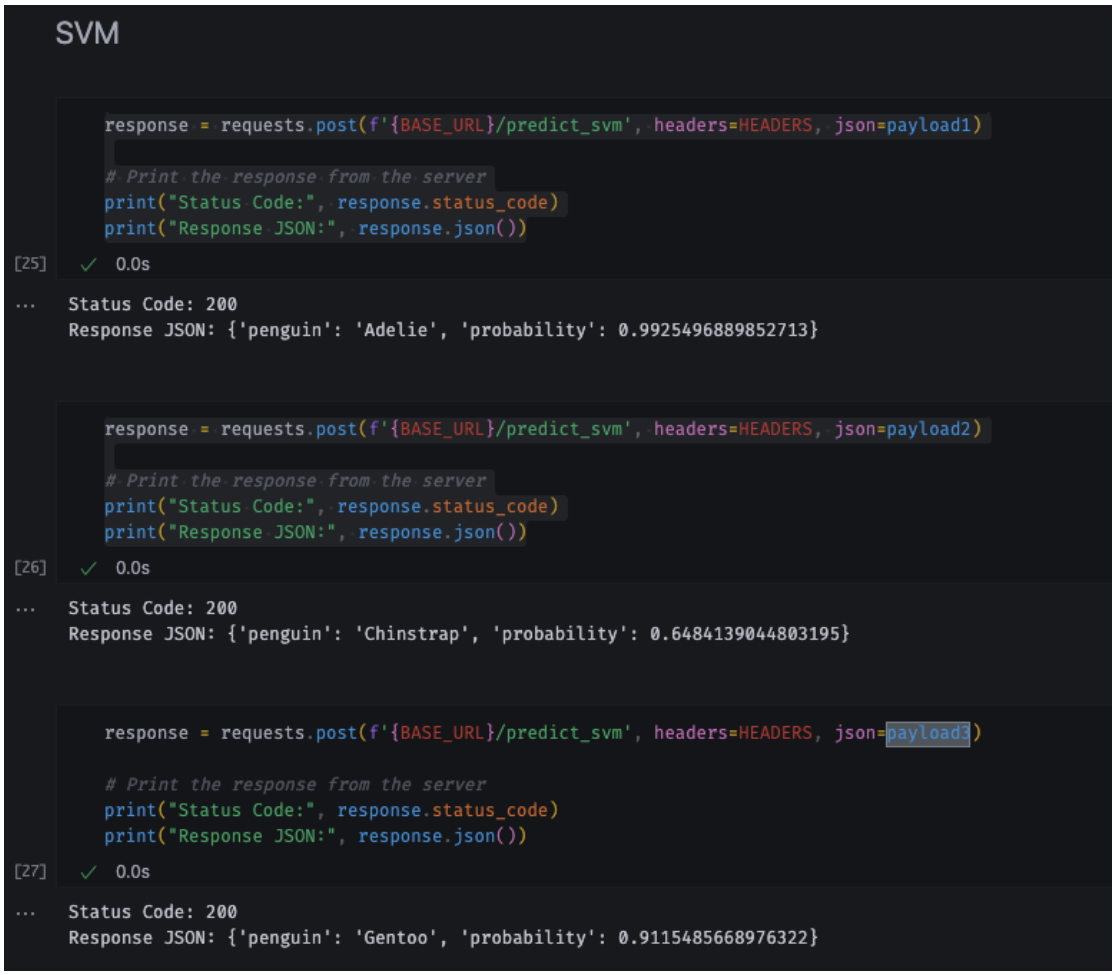


Logistic Regression



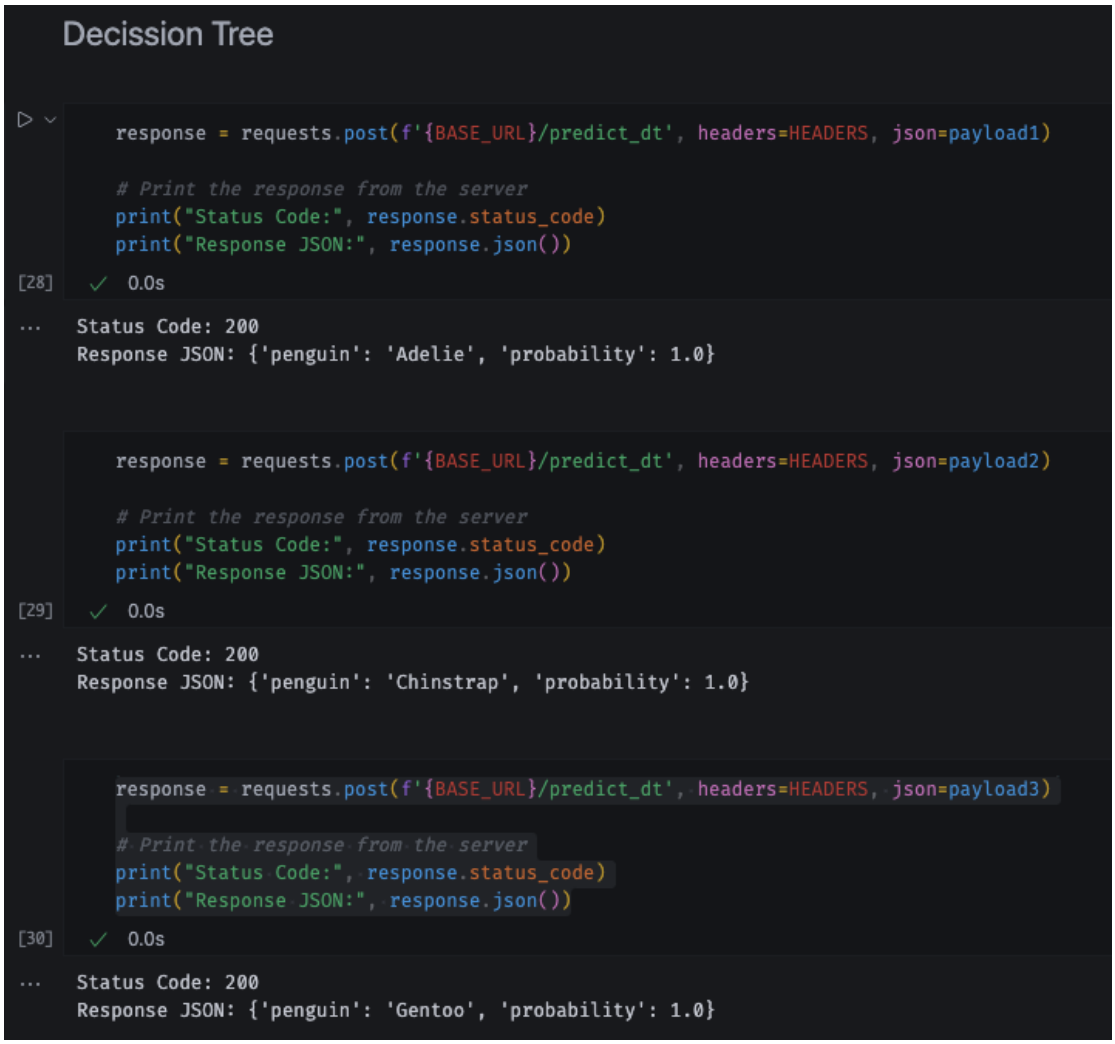
Como se puede ver, los porcentajes de probabilidad son bastante parejos, con el segundo difiriendo un poco pero aún así siendo relevante.

SVM



Como sucedía con el modelo anterior, el primero y el tercero tienen probabilidades altas, mientras que el segundo no tan alto.

Decission Tree



En este caso se puede ver que el modelo es resolutivo al 100% en los 3 casos planteados.

K-Nearest Neighbors

K-Nearest Neighbors

```
response = requests.post(f'{BASE_URL}/predict_knn', headers=HEADERS, json=payload1)

# Print the response from the server
print("Status Code:", response.status_code)
print("Response JSON:", response.json())
```

[31] ✓ 0.0s

```
Status Code: 200
Response JSON: {'penguin': 'Adelie', 'probability': 1.0}
```

```
response = requests.post(f'{BASE_URL}/predict_knn', headers=HEADERS, json=payload2)

# Print the response from the server
print("Status Code:", response.status_code)
print("Response JSON:", response.json())
```

[32] ✓ 0.0s

```
Status Code: 200
Response JSON: {'penguin': 'Gentoo', 'probability': 0.6666666666666666}
```

```
response = requests.post(f'{BASE_URL}/predict_knn', headers=HEADERS, json=payload3)

# Print the response from the server
print("Status Code:", response.status_code)
print("Response JSON:", response.json())
```

[33] ✓ 0.0s

```
Status Code: 200
Response JSON: {'penguin': 'Gentoo', 'probability': 1.0}
```

De nuevo como con los dos primeros modelos, se vuelve a plantear la misma tesitura con los resultados.

Comentario general sobre los resultados

Como se ha ido mencionando, el primer y tercer resultado tenían un porcentaje bastante significativo de acierto para los 3 casos planteados de datos, mientras que el segundo caía entre un 60% y 70% aproximadamente, siendo aún ciertamente relevante pero significativamente menos que los otros dos. Esto puede sugerir un problema con el caso planteado en concreto, que los datos no estén bien ajustados a la realidad[1].

Por otra parte, el tercer modelo (*Decission Tree*) aporta un 100% de probabilidades en los 3 casos. Dado el problema planteado, podría ser factible que este modelo pueda llegar a ser el más preciso de los 4 pero sigue siendo un tanto extraño un acierto absoluto en todos los casos.

[1] Los datos de ejemplo han sido proveídos por ChatGPT, habiéndole añadido el contexto del uso de `seaborn` y los datos sobre pingüinos para un mejor resultado del prompt.