Deploy do primeiro modelo de Machine Learning (FastAPI)#datascience

Passo a passo de como fazer o deploy de um modelo de machine learning usando *FastAPI* ⚡

iddhardhan

How to Deploy Machine Learning Model as an API in Python - FastAPI
youtube.com

Salve o modelo que você treinou

- Após feita toda etapa de análise, visualização e entendimento dos dados (preferencialmente usando algum framework, como o CRISP-DM), vamos salvar o modelo usando a biblioteca pickle

Vamos para o código! 👨🏻‍💻

1. Para o coding vamos precisar fazer os seguintes imports:

```
from fastapi import FastAPI
from pydantic import BaseModel
import pickle
import json
```

2. Apos feito o import, vamos instanciar a classe do *FastAPI* e modelar nossa classe *model_input*, que vai ser responsável por padronizar o input dos dados na nossa *API*, vai ficar mais ou menos assim:

```
app = FastAPI()

class model_input(BaseModel):
    PassangerId: int
```

```
    Pclass: int
    Name: str
    Sex: str
    Age: int
    SibSp: int
    Parch: int
    Ticket: str
    Fare: float
    Cabin: int
    Embarked: str
```

3. Agora vamos instanciar o modelo que salvamos no formato *.pkl* usando a biblioteca *pickle* e também vamos definir nossa função *titanic_pred*, responsável pelo envio dos dados para a nossa *API*, que recebe como parâmetro a nossa classe *model_input*.

```
titanic_model = pickle.load(open("modelo.pkl","rb"))

@app.post("/titanic_prediction")
def titanic_pred(input_params: model_input):
    pass
```

4. Implementação da função *titanic_pred:*

```
@app.post("/titanic_prediction")
def titanic_pred(input_params: model_input):
    input_data = input_params.json()

    # converts json input to python dict
    input_dictionary = json.loads(input_data)

    # now we convert our dict to a list
    passanger_id = input_dictionary["PassangerId"]
    pclass = input_dictionary["Pclass"]
    name = input_dictionary["Name"]
    sex = input_dictionary["Sex"]
    age = input_dictionary["Age"]
    sibsb = input_dictionary["SibSp"]
    parch = input_dictionary["Parch"]
    ticket = input_dictionary["Ticket"]
    fare = input_dictionary["Fare"]
    cabin = input_dictionary["Cabin"]
    embarked = input_dictionary["Embarked"]

     input_list = [
        passanger_id,
        pclass,
```

```
        name,
        sex,
        age,
        sibsb,
        parch,
        ticket,
        fare,
        cabin,
        embarked,
    ]
```

- Primeiramente nos iremos converter o input da função para json, o formato ideal para trocar dados entre nossa aplicação e a API

- O proximo passo é fazer um dicionário para em seguida extrair em forma de variável para cada atributo presente na classe model_input, para que seja mais fácil lidar com estes dados no código. Feito isso nos vamos juntar tudo em uma única lista.

```
prediction = titanic_model.predict([input_list])

if prediction[0] == 0:
    return "Essa pessoa sobreviveu ao Titanic"
else:
    return "Essa pessoa não sobreviveu ao titanic"
```

- Agora, devemos fazer a previsão usando a nossa lista como input

- Fix uma condicional que pode retornar diferentes valores, caso o passageiro seja um possível sobrevivente (afinal, o modelo não é 100% preciso) ou não.

5. Deploy do modelo

- Para realizar o deploy, vamos no terminal e digitar:

```
uvicorn [nome_do_arquivo.py]:app
```

- O :*app* é nada mais nada menos do que a instanciação da classe do *FastAPI* que definimos no código

6. Testando a nossa API

- Nossa API já esta no localhost e agora devemos testa-la, e para isso criei um novo arquivo em python para fazer este teste. Ele ficou assim

```
import json
import requests

url = "http://127.0.0.1:8000/titanic_prediction"

input_data_for_model = {
    "PassangerId": 710,
    "Pclass": 3,
    "Name": "Moubarek, Master. Halim Gonios ('William
George')",
    "Sex": "male",
    "Age": -9999.0,
    "SibSp": 1,
    "Parch": 1,
    "Ticket": "2661",
    "Fare": 15.2458,
    "Cabin": -9999,
    "Embarked": "C",
}

# transformando o dict em json para mandar os dados para
nossa api

input_json = json.dumps(input_data_for_model)
response = requests.post(url,data=input_json)
print(response.text)
```

- A url que vamos fazer a requisição post deve conter também nosso endpoint definido na função no outro arquivo. Que no caso foi */titanic_prediction*

- Iremos pegar os dados em forma de dict e depois fazer o json.dump para transforma-los em json

- Apos isso iremos enviar a request no método post e armazenar na variável response

- Por fim iremos printar a predição do nosso modelo no terminal.