DigitalHouse > Coding School

DATA SCIENCE

Modelos: Persistencia y Desarrollo de una API

OBJETIVOS DE LA CLASE

1. **Serialización**
2. **Utilización de pickle para persistencia.**
3. **Servicio Web**
4. **Implementación en Python con Framework Flask**

INTRODUCCIÓN

Para poder disponibilizar nuestro modelo es necesario cumplir con una serie de pasos:

- Entrenar nuestro modelo
- Guardarlo de alguna forma para poder transmitirlo a un entorno distinto al de desarrollo.
- Cargar el modelo en memoria dentro de un sistema en producción.
- Hacer que el modelo reciba los datos del exterior para hacer una predicción.
- Tomar esta predicción y transformarla para responder la petición.

¿Como se realizarían estas tareas de forma local?

Serialización

Serialización

En algunas circunstancias nos encontramos con la necesidad de enviar un objeto python de un programa a otro, o a otra computadora, o guardarlo para utilizarlo en algún otro momento o ir grabando sus diferentes estados.

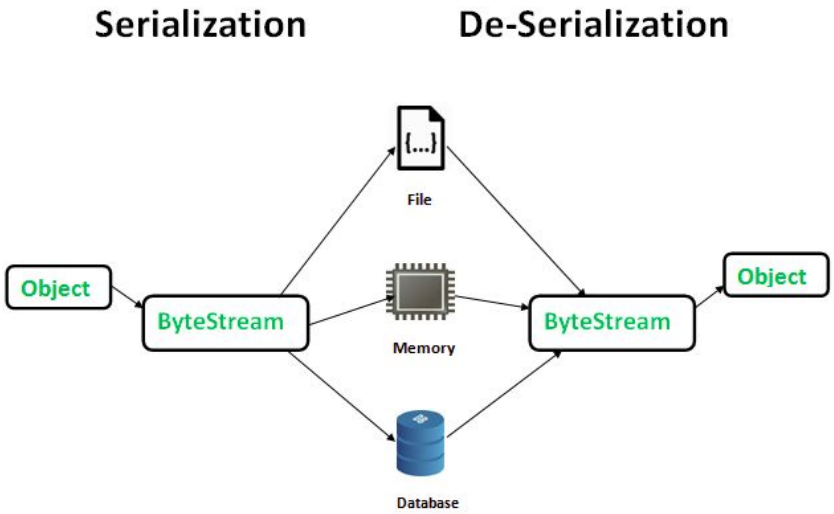
- Existen diferentes soluciones para realizar estas acciones, y dependen de lo que se quiera realizar.

- Por ejemplo: si tenemos un texto, lo podemos representar dentro de una especificación json o xml, por otra parte un dataframe de pandas, se puede volcar en un csv o un excel.

En el caso de objetos más complicados, como un modelo de regresión lineal o un knn, ¿como se les podría representar?

Serialización

- La serialización es una técnica en donde se representan objetos por medio de una codificación binaria.
- Permite realizar la codificación sobre objetos y estructuras de datos.
- Es posible enviar a través de la red un objeto serializado como un conjunto de bytes para eventualmente guardarlo, cargarlo en memoria y volverlo a utilizar con la posibilidad de poder transmitirlo.



Pickle

Para serializar objetos python vamos a utilizar la librería pickle.

Pickle es un estandar python, por lo que no es compatible con otros lenguajes de programación.

¿ Que cosas puede serializar pickle?

- Enteros
- Float
- Tuplas
- Listas
- Diccionarios
- Conjuntos
- Funciones
- Clases

- **Instancias de Clases**

Diferencias con json:

- Json viene en formato texto (en su mayoria UTF-8)
- Json es comprensible por humanos
- Json puede ser interpretados fuera de python
- Json puede representar sólo algunos tipos de python

Pickle no es seguro contra data errónea o maliciosa, se recomienda no deserializar data desde fuentes desconocidas.

Serializamos una tupla con 3 elementos numéricos

```
import pickle
file = open('tupla.pkl','wb')
pickle.dump( (-1,2,3), file )
file.close()
```

```
python3 -m pickletools tupla.pkl -a
```

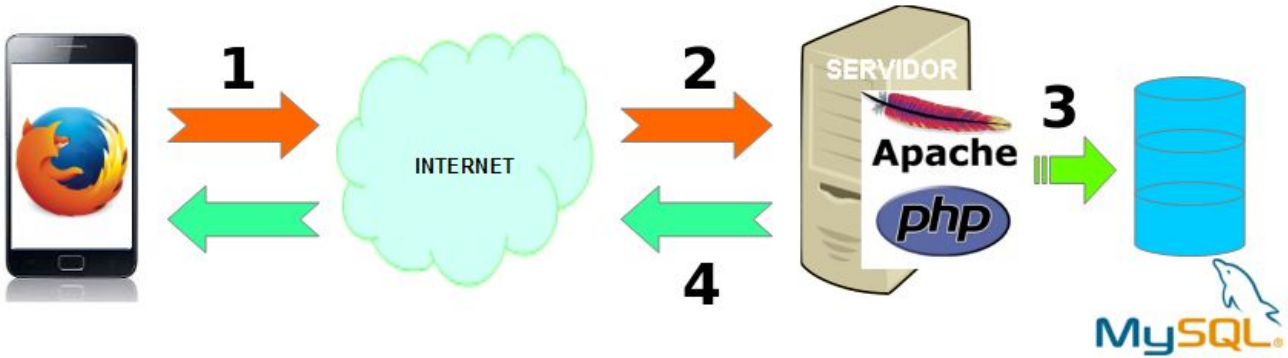
Usamos el módulo pickletools para interpretar el archivo.

```
 0: \x80  PROTO     3          Protocol version indicator.
 2: J     BININT    -1         Push a four-byte signed integer.
 7: K     BININT1   2          Push a one-byte unsigned integer.
 9: K     BININT1   3          Push a one-byte unsigned integer.
11: \x87  TUPLE3              Build a three-tuple out of the top three items on the stack.
12: q     BINPUT    0          Store the stack top into the memo.  The stack is not popped.
14: .     STOP                 Stop the unpickling machine.
```

Es importante señalar que los módulos y dependencias deben existir en el entorno de deserializacion para los tipos de datos que no son nativos de python.

Servicios Web

Las aplicaciones del lado del servidor son programas que corren en los servidores web,, procesan las solicitudes que el servidor recibe y generan respuestas que espera el cliente.



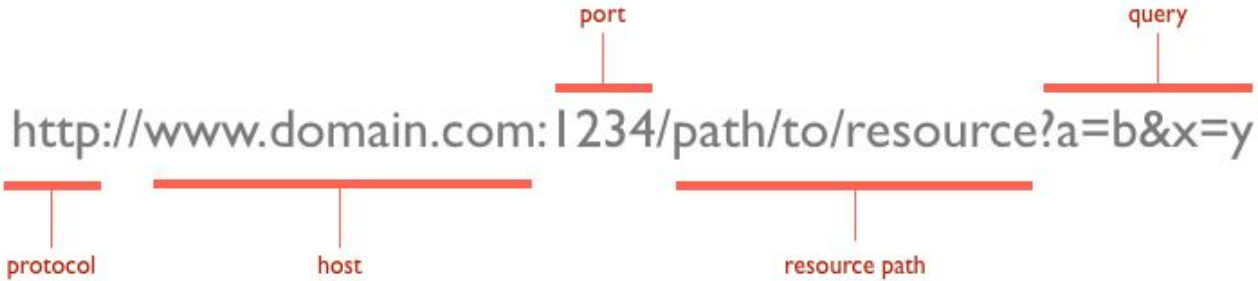
1. El cliente envía una solicitud, HTTP Request, al Servidor Web .
2. El Servidor Web interpreta la solicitud y ejecuta la Aplicación correspondiente.
3. La Aplicación del lado del servidor accede a la base de datos y genera la respuesta.
4. El Servidor Web envía la respuesta, HTTP Response, al cliente

REST es un estilo de arquitectura de software en donde se enuncian una serie de reglas para el diseño de un sistema distribuido en una red (como internet)

- **Identificador único:** Tiene que existir una URI única
- **Interfaz uniforme :** se aplican acciones concretas (POST, GET, PUT y DELETE) sobre los recursos, recursos que están identificados con una URI.
- **Exposición de recursos a través de representaciones:** Los recursos son accedidos a través de su URI (identificador de recursos uniforme) en forma de una representación (text/csv, application/json, audio/midi), las representaciones pueden ser múltiples.
- **Relación Cliente/Servidor sin estado:** La comunicación no presenta estado, la información provista debería ser lo suficientemente autodescriptiva para configurar un mensaje http. (cache, login, cookies), siempre devuelve el mismo estado final..

- ¿Como hace nuestra aplicación para saber qué es lo que se está pidiendo en la solicitud?

  Esto es especificado en la **URI** (Uniform Resource Identifier), una ruta que indica donde se puede encontrar un recurso.



- Sobre este recurso se puede definir un método http.

Interface uniforme, métodos http:

- GET : Solicita una representación de un recurso específico. Sólo se deben recuperar datos.
- PUT : Reemplaza la representación con los datos en la petición.
- POST : El método POST se utiliza para enviar una entidad a un recurso (puede generar un cambio).
- DELETE : Borra la representación de un recurso.

REST : Es el estilo de arquitectura de software.

RESTful : Es el servicio web que la implementa.

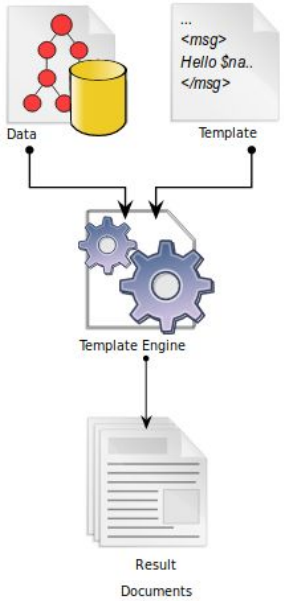
Flask

- Flask es un microframework o sea, un conjunto de herramientas para desarrollar aplicaciones web.

- Esta basado en WSGI (Web Server Gateway Interface), una especificación para desarrollo de servidores web y aplicaciones web python (Python Enhancement Proposal 333).

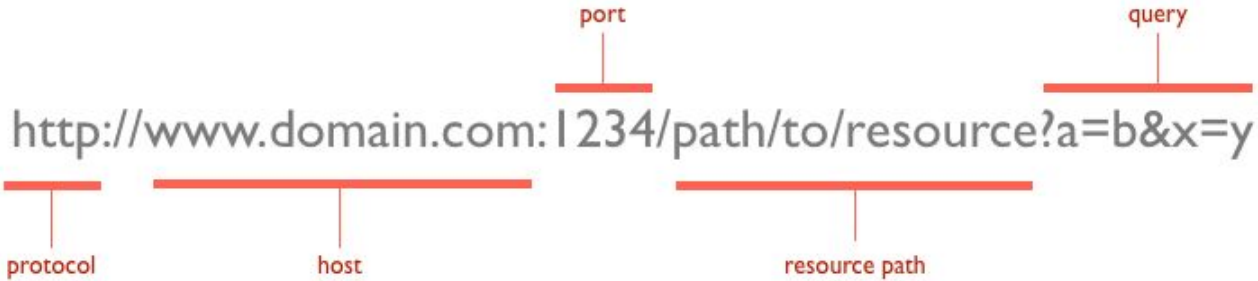
- Utiliza Jinja2 sobre templates (software para hacer modificaciones sobre un html y generar documentos)

- Flask permite tomar una función python y asociarla a una URI a través del routing utilizando el diseño REST.



- ¿Como hace nuestra aplicación para saber qué es lo que se está pidiendo en la solicitud?

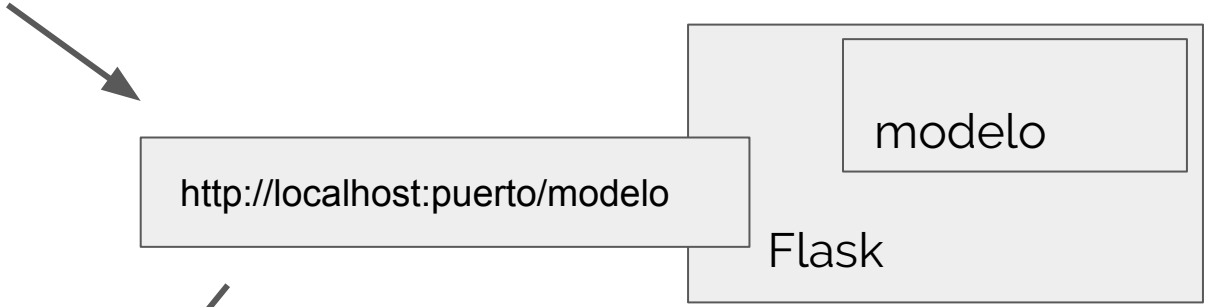
  Esto es especificado en la **URI** (Uniform Resource Identifier), una ruta que indica donde se puede encontrar un recurso.



- Sobre este recurso se puede definir un método http.

REQUEST POST

```
{ feature1. : '1'
  feature2  : '0'
  feature3  : '1'
     .
     .
   fearureN: '0'}
```



RESPONSE BODY

```
{ predict: 2.83 }
```

Vamos a posicionar la representación del recurso "modelo" en una URI.

Mediante un request post tomaremos los datos desde un json, entregamos esos datos al modelo.

Finalmente devolveremos la predicción en el cuerpo del response.

```python
from flask import Flask
app = Flask(__name__)
```

```python
@app.route('/')
def index():
    return 'Index Page'

@app.route('/hello')
def hello():
    return 'Hello, World'
```

```python
from flask import request

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        return do_the_login()
    else:
        return show_the_login_form()
```

PRACTICA GUIADA

CONCLUSIÓN

- Hemos aprendido sobre Serialización y representación de recursos en una aplicación web con python utilizando Flask y Pickle.

- Esto es importante para entender cómo se realiza la puesta en producción de un modelo.