Maestría
en Inteligencia
Analítica de Datos

Taller Empaquetamiento

Instrucciones:

En este taller emplearemos algunas herramientas para el empaquetamiento de modelos de analítica en python.

La entrega de este taller consiste en un reporte en formato de documento de texto, donde pueda incorporar fácilmente capturas de pantalla, textos y elementos similares. Puede utilizar formatos como Word, LibreOffice, Markdown, u otros.

Parte 1: desarrollo de modelos

1. Junto a este enunciado encontrará el cuaderno *bank-churners-train-test.ipynb* y los datos *Bank-Churn.csv*.
2. Abra estos archivos localmente, ejecute el cuaderno y explore en detalle su ejecución.
3. Este cuaderno puede tomarse como el resultado final de un proceso de exploración que nos ha llevado al modelo que allí se entrena y prueba.
   Incluya en su reporte una descripción de qué hace este cuaderno en detalle.

Parte 2: preparación del empaquetamiento

1. Junto a este enunciado encontrará el archivo *package-src.zip*.
2. Descomprima este archivo localmente. Encontrará una estructura de carpetas con múltiples archivos.
3. Abra el archivo setup.py y familiarícese con su estructura y contenido.
4. Abra el archivo tox.ini y familiarícese con su estructura y contenido. En este archivo se definen dos ambientes clave que usaremos hacia adelante: test_package y train. En cada uno de estos ambientes se define un directorio para el ambiente, un conjunto de dependencias y los comandos a ejecutar.
5. En la carpeta model encontrará el archivo train_pipeline.py, que ejecuta el entrenamiento. Allí se emplea el pipeline de datos definido en el archivo pipeline.py y se entrena el modelo y guarda el modelo.

6. Note que en éste y otros scripts se hace referencia al archivo config.yml, el cual define el valor que toman algunos parámetros clave. Explore este archivo con detenimiento para entender los parámetros definidos.**Incluya en su reporte una descripción de qué hace el archivo *config.yml* en detalle.**
7. En la carpeta model encontrará el archivo predict.py, que realiza predicciones. Note que para esto se emplea también el pipeline de datos definido en el archivo pipeline.py y la validación de datos disponible en el archivo processing/validation.py.
8. Revise el archivo processing/validation.py, donde encontrará los esquemas de un dato DataInputSchema y de múltiples datos MultipleDataInputs.
9. Explore la carpeta requirements, que contiene las dependencias del paquete (requirements.txt), las de prueba (test_requirements.txt) y las tipos (typing_requirements.txt).
10. En la carpeta tests encontrará los archivos de prueba. En *conftest.py* la configuración inicial que define el archivo de prueba. Y en *test_prediction.py* una prueba unitaria del método de predicción. **Incluya en su reporte una descripción de qué hace el archivo *test_prediction.py* en detalle.**
11. Note que en este paquete contamos con esencialmente los mismos pasos que en el cuaderno de jupyter, pero estructurados de una manera adecuada para empaquetamiento.

Parte 3: empaquetamiento

1. Lance una instancia en AWS EC2. Se recomienda una máquina t2.small, con sistema operativo Ubuntu y 20GB de disco. Incluya un pantallazo de la consola de AWS EC2 con la máquina en ejecución en su reporte. Su usuario de AWS debe estar visible en el pantallazo.
2. Suba el archivo package-src.zip a la máquina con un comando como

```
scp -i llave.pem package-src.zip ubuntu@IP:/home/ubuntu
```

3. Conéctese a la máquina:
   a) Abra una terminal: En windows, escriba *cmd* y *Enter*. En macOS, abra la aplicación llamada *Terminal*.
   b) En la terminal emita el comando

```
ssh -i /path/to/llave.pem ubuntu@IP
```

   donde /path/to/ se refiere a la ubicación del archivo llave.pem que descargó, e IP es la dirección IP de la instancia EC2 que lanzó. Si prefiere, en la terminal puede navegar a la ubicación del archivo llave.pem y emitir el comando

```
ssh -i llave.pem ubuntu@IP
```

Note que usamos en este caso ubuntu, pues éste es el usuario creado por defecto como administrador con sistema operativo Ubuntu server. Incluya en su reporte un *screenshot* de la conexión a la máquina virtual.

4. Actualice las ubicaciones de los paquetes con el siguiente comando:

```
sudo apt update
```

5. Instale pip para python3

```
sudo apt install python3-pip
```

6. Instale unzip para manipular los archivos .zip

```
sudo apt install zip unzip
```

7. Instale vemv para crear ambientes virtuales con el siguiente comando:

```
sudo apt install python3.12-venv
```

8. Cree un ambiente virtual con el nombre env-tox

```
python3 -m venv /home/ubuntu/env-tox
```

9. Active el ambiente con el comando:

```
source env-tox/bin/activate
```

10. Descomprima el archivo package-src.zip

```
unzip package-src.zip
```

Verifique que haya quedado la carpeta package-src creada

11. Instale tox, una librería de automatización y pruebas

```
pip install tox
sudo apt-get install tox
```

Incluya la dirección /home/ubuntu/.local/bin en el PATH para facilitar su ejecución

```
PATH=$PATH:/home/ubuntu/.local/bin
```

12. Ingrese a la carpeta package-src

```
cd package-src
```

y ejecute el ambiente de entrenamiento de tox

```
tox run -e train
```

Incluya en su reporte un *screenshot* de la salida de este comando y una breve explicación. Verifique que se pasen todas las pruebas.

13. Ejecute ahora el ambiente de prueba del paquete de tox

```
tox run -e test_package
```

Incluya en su reporte un *screenshot* de la salida de este comando y una breve explicación. Verifique que se pasen todas las pruebas.

14. Con todas las pruebas pasadas, procederemos a construir el paquete, para esto necesitamos instalar el paquete build

```
python3 -m pip install --upgrade build
```

y construimos el paquete con el comando

```
python3 -m build
```

Note que se crea una carpeta **dist** en donde se tienen archivos para la instalación del paquete (uno tar.gz y uno whl). **Incluya en su reporte un *screenshot* donde se evidencien los archivos creados en esta carpeta.**

15. En su carpeta home en la máquina (/home/ubuntu) cree una nueva carpeta test.

```
cd /home/ubuntu
mkdir test
```

16. Ingrese a esta carpeta y copie allí el archivo whl del paquete creado

```
cd test
cp ../package-src/dist/model_abandono-0.0.1-py3-none-any.whl .
```

Note el punto al final del comando.

17. Instale el paquete

```
pip install model_abandono-0.0.1-py3-none-any.whl
```

Note que se instalan todas las dependencias del paquete.

18. Ejecute el siguiente comando para mostrar los paquetes instalados:

```
pip freeze
```

Incluya en su reporte un *screenshot* donde se evidencien los paquetes.

19. A la carpeta test suba los archivos test-package.py y bankchurn_test.csv.
20. Ejecute el archivo test-package.py

```
python3 test-package.py
```

Incluya en su reporte un *screenshot* donde se evidencie la ejecución de este comando.

21. Descargue el archivo whl en su equipo usando un comando como (desde una terminal local)


```
scp -i llave.pem ubuntu@IP:/home/ubuntu/package-src/dist/model_abandono-0.0.1-
    py3-none-any.whl .
```

Note el punto al final del comando (indica que el archivo remoto se copia localmente en la carpeta actual).

Parte 4: modificando el paquete

1. Localmente, modifique su modelo para usar todas las características excepto una de su elección. Para ello, edite el archivo *config.yml* eliminando uno de los elementos en features y agregándolo a temp_feautres. **Incluya en su reporte un pantallazo de la modificación realizada.**
2. Modifique el archivo VERSION de tal forma que ahora corresponda a la versión 0.0.2. **Incluya en su reporte un pantallazo de la modificación realizada.**
3. Comprima la carpeta con las modificaciones. Adicionalmente, renombre la carpeta como "package-src2.zip".
4. Utilizando un repositorio en Git, realice el commit correspondiente para subir la nueva carpeta comprimida. Para ello, localmente deberá crear una carpeta nueva que contenga a "package-src2.zip", utilizar git para agregar el archivo, realizar el respectivo commit, enlazar este repositorio con un nuevo repositorio en línea en GitHub, enlazarlo con su máquina local y realizar el push para que quede en la rama master remota. Las líneas de código se muestran a continuación asumiendo que abre la terminal local desde la carpeta local donde tiene el archivo comprimido y el URL del repositorio en GitHub es URL:

   ```
   git init
   git add package-src2.zip
   git commit -m "paquete modificado"
   git remote add origin URL
   git push origin master
   ```

   Incluya en su reporte un pantallazo del archivo package-src2.zip en GitHub.
5. Desde su máquina virtual, asegúrese de que Git esté instalado utilizando el siguiente comando:

   ```
   sudo apt install git
   ```

6. Clone el repositorio con la nueva versión del paquete. Para ello utilice el siguiente comando, donde URL corresponde al enlace HTTPS del respositorio en GitHub:

   ```
   git clone URL
   ```

7. Ingrese a la carpeta clonada utilizando el siguiente comando asumiendo que su repositorio en GitHub tiene el nombre my-repo:

```
cd my-repo
```

8. Descomprima el archivo utilizando el siguiente comando:

```
unzip package-src2.zip
```

9. Con este nuevo paquete, realice nuevamente desde el paso 9 hasta el 17 de la parte 3 para verificar que se pasan adecuadamente los pasos de prueba y empaquetamiento.

```
cd package-src2
tox run -e train
tox run -e test_package
python3 -m build
cd /home/ubuntu
mkdir test2
cd test2
cp ../package-src2/dist/model_abandono-0.0.2-py3-none-any.whl .
pip install model_abandono-0.0.2-py3-none-any.whl
pip freeze
python3 test-package.py
```

Incluya en su reporte un *screenshot* donde se evidencie la instalación del paquete y explique brevemente el resultado.

10. Descargue en su máquina local el archivo de empaquetamiento generado. **Incluya en su reporte un pantallazo del archivo model_abandono-0.0.2-py3-none-any.whl en su máquina local.**