* **Parte 1: desarrollo de modelos**

Incluya en su reporte una descripción de qué hace este cuaderno en detalle.

Este cuaderno inicialmente carga nuestra base de datos llamada BankChurn.csv. Despues se explora nuestro set de datos observando métricas numéricas y para algunas variables categóricas se cuenta el numero de valores únicos existentes. Adicionalmente se calculan métricas de churn, valores duplicados, valores faltantes y tipos de datos del dataframe. Seguidamente se realiza un preprocesamiento de nuestras variables convirtiendo variables categóricas en numéricas (ejemplo la variable Gender) y se hace escalamiento de las variables numéricas. Despues se dividen los datos en X y Y para poder realizar el entrenamiento de nuestro modelo. Se seleccionan las variables de interés, en este caso en total se tienen 9 variables de interés. Seguidamente se define el modelo a manejar el cual será Random Forest clasifier, se realiza el entrenamiento del modelo y se calcula la precisión del modelo usando métricas como Accuracy, F1 Score, Recall, etc. Para complementar la evaluación del desempeño del modelo se usa cross validation con 8 pliegues y se evaluan las métricas de (f1', 'accuracy' y'roc\_auc'.

**Parte 2: preparación del empaquetamiento**

Incluya en su reporte una descripción de qué hace el archivo config.yml en detalle.

El archivo config.yaml define parámetros que son requeridos por lso diferentes scripts del proyecto. Se tienen parámetros para Nombre del paquete, Archivos de datos (mis datos para entrenar y testear mi modelo), Variables (mis variables de interés para entrenar mi modelo), pipelines, fracción del conjunto de prueba (porcentaje que se va a asignar para los datos de prueba), semilla (para la reproducibilidad de mi experimento), hiperparametros de mi modelo, y variables categóricas que deben mapearse para convertirlas a numéricas.

Incluya en su reporte una descripción de qué hace el archivo test\_prediction.py en detalle.

El archivo test\_prediction.py esta importando del archivo model.predict la funcion make\_prediction. Luego se define una función llamada test\_make\_prediction que recibe como parametros un sample\_input\_data. La función define dos variables, expected\_first\_prediction\_value y expected\_no\_predictions con valores por defecto. Despues llama la función make\_prediction pasando los datos como parámetro. La función make\_prediction realiza la predicción usando un modelo guardado. Luego realiza la validación revisando que no existan errores y que se reciba el numero de predicciones esperado.

**Parte 3: empaquetamiento**

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**A screenshot of a computer

Description automatically generated**

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.

A screenshot of a computer

Description automatically generated

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.

A screen shot of a computer

Description automatically generated

Se puede observar que primero se instalan las ependencias usando el comando de pip install. Seguidamente se ejecuta el comando python model/train\_pipeline.py que ejecuta el script train\_pipeline.py que va a generar nuestro modelo de acuerdo al código definido.

Ejecute ahora el ambiente de prueba del paquete de tox: tox run -e test\_packag. Incluya en su reporte un screenshot de la salida de este comando y una breve explicación.

A screen shot of a computer

Description automatically generated

Podemos observar que igualmente se instalan las librerías y se ejecuta el script model/train\_pipeline.py. Esto genera a su vez la ejecución del script tests/test\_prediction.py::test\_make\_prediction el cual como se puede observar en la imagen genera un resultado de pruebas exitosas.

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



Ejecute el siguiente comando para mostrar los paquetes instalados: pip freeze. Incluya en su reporte un screenshot donde se evidencien los paquetes.

A screenshot of a computer

Description automatically generated

Ejecute el archivo test-package.py python3 test - package .py Incluya en su reporte un screenshot donde se evidencie la ejecución de este comando.

A black and white background

Description automatically generated

**Parte 4: modificando el paquete**

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.

A screenshot of a computer

Description automatically generated

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.

A screenshot of a computer

Description automatically generated

Incluya en su reporte un pantallazo del archivo package-src2.zip en GitHub.

A screenshot of a computer

Description automatically generated

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

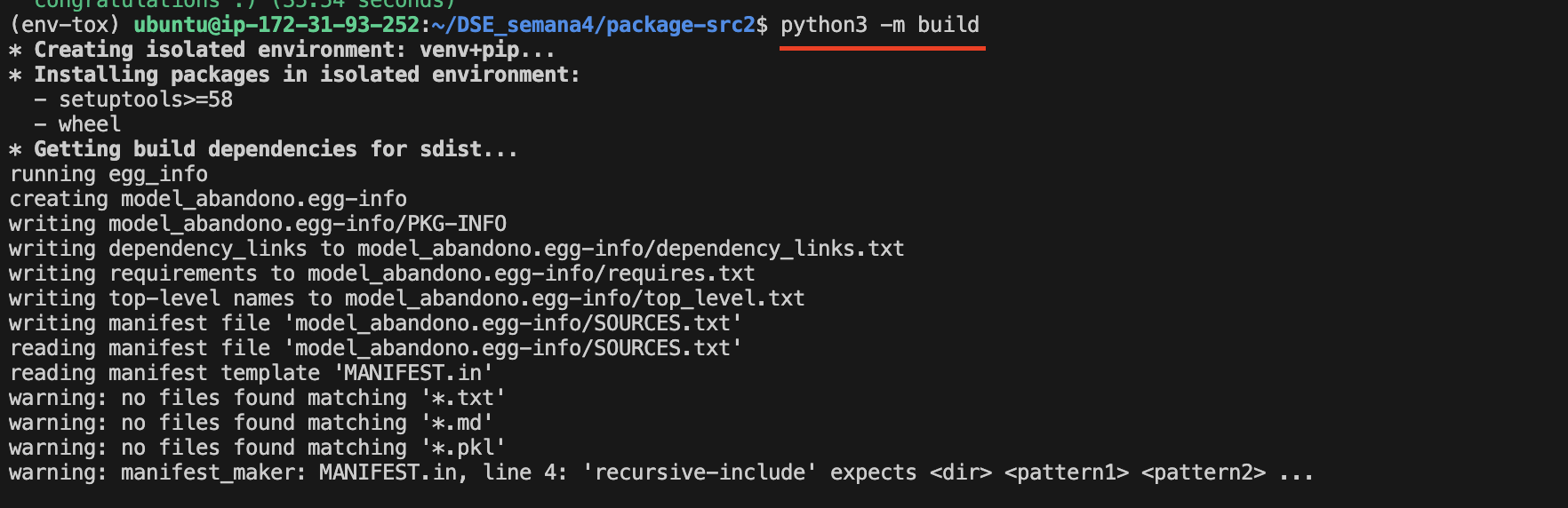
pip freeze

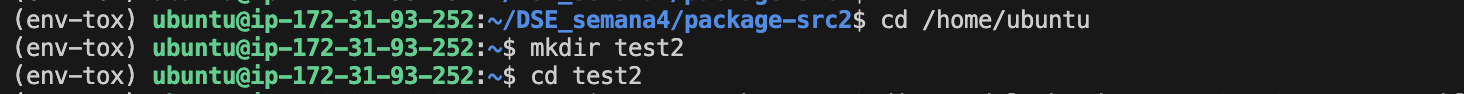
python3 test - package .py

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

A screenshot of a computer screen

Description automatically generated







A screen shot of a computer screen

Description automatically generated

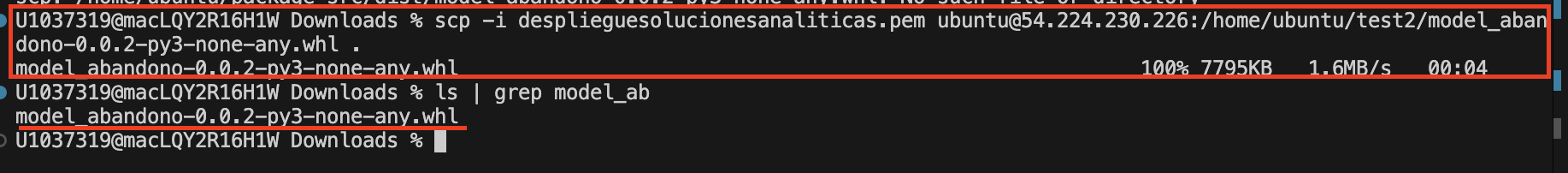


A black and white background

Description automatically generated with medium confidence

Podemos observar que se ha ejecutado el nuevo modelo incluyendo las cambios que habíamos especificado en un inicio, como fue una variable que no tuvimos en cuenta y el cambio del versionamiento en el archivo. El resultado final nos muestra la ejecución de nuestro modelo usando nuestro empaquetamiento del modelo.

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.



A screenshot of a computer

Description automatically generated