Reto datos

Herramientas y tecnologías

Las herramientas y tecnologías que vamos a utilizar para este proyecto son las siguientes

Etapa	Herramientas
#1	Ninguna
#2	• Oracle Cloud • Jupyter Notebook • Visual Studio Code • Python • Numpy • Pandas • Ubuntu • Spark • Windows
#3	• Oracle Cloud • Jupyter Notebook • Python • Numpy • Pandas • Open Street Map • Windows • GitHub • Visual Studio Code
#4	• Oracle Cloud • Jupyter Notebook • Python • Numpy • Pandas • Windows • GitHub • Visual Studio Code
#5	• Oracle Cloud • Jupyter Notebook • Python • Numpy • Pandas • Windows • GitHub • Visual Studio Code
#6	Ninguna

- Jupyter Notebook
- Visual Studio Code
- Python

- Spark con PySpark
- Pandas
- Numpy

- Una instancia de Oracle Cloud
- Ubuntu 20.04
- Windows 10/11
- Open Street Map
- Git + GitHub
- Google Colab

Usaremos estas tecnologías ya que Oracle Cloud + Jupyter Notebook nos ayudarán a tener muchos recursos disponibles y velocidad buena en procesamiento para trabajar. En caso de buscar una velocidad muy rápida usaremos Google Colab pero la RAM será más limitada que en Oracle Cloud.

Ubuntu en Oracle Cloud y en nuestras máquinas nos facilitarán el procesado de datos, en Cloud cuando se trate de algo muy grande y que tarde mucho tiempo y en local cuando se trate de algo pequeño.

Windows lo usaremos como sistema operativo principal, ya que no todos tenemos Ubuntu de forma nativa, la ventaja es que podemos usar los recursos de la nube desde ahí, así como otras aplicaciones sin mayor complicación.

Con la ayuda de Visual Studio Code nos podemos conectar a la instancia de Oracle Cloud que contiene el entorno de Jupyter Notebook.

Spark + Pandas + Numpy nos ayudarán a realizar operaciones de forma rápida a los datos con los que estemos trabajando. Spark es muy bueno en velocidad de lectura de datos una vez que está cargada a la RAM, Pandas y Numpy tomarán más el rol de realizar distintos tipos de operaciones.

Las características del servidor que estamos utilizando son:

- Procesador Ampere, 4 núcleos, 3.0 GHz
- 24 GB de RAM, DDR4 3200 MHz
- SSD: 200 GB

Openstreetmap nos dará datos de ubicaciones para extraer la dirección completa de coordenadas, así obtendremos las comunas para identificar dónde se encuentran las antenas.

Git + GitHub será nuestro sistema de control de versiones, en caso de que algo llegue a pasar, podemos acceder a lo último y más estable que hayamos hecho en esa plataforma.

La combinación de todas estas tecnologías nos hará trabajar de forma más rápida y con más seguridad de que lo que hagamos pueda correr sin problemas de que se nos acabe la memoria RAM o que nuestros procesadores sean muy lentos a la hora de realizar diversas operaciones con el dataset.

Modelo de almacenamiento de los datos

En un principio, pensamos usar varias instancias virtuales en la nube por los beneficios que nos ofrece Spark, pues lo que hace es leer todo el dataset del Disco Duro o Unidad de Estado Sólido, lo carga a la memoria RAM y el acceso se vuelve más rápido.

Pyspark nos da más ventajas, entre ellas el poder usar múltiples núcleos de forma automatizada para realizar operaciones de manera más rápida.

Pero debido a limitaciones, dejamos la idea de usar un sistema de clusters ya que Oracle no lo permite en su free-tier. Sólo se puede tener 1 máquina virtual con 24 GB de RAM con 4 núcleos de procesamiento o 4 máquinas, 1 core cada una y 6GB.

Por otro lado, conectarlas de cuenta a cuenta no sería muy bueno a pesar de que seleccionemos el mismo centro de datos de Oracle. Así mismo dicha plataforma a veces suele limitar las transferencias de información con los planes free-tier y les da más prioridad de procesado a las personas que pagan por una instancia, por lo que puede que dos cuentas caigan en un mismo recurso y todo se haga lento de repente.

Como modelo de almacenamiento, estamos usando el sistema de archivos ZFS de Linux para almacenar todo nuestro dataset original y archivos derivados en la nube. Si se requiere realizar procesamiento, ahí mismo hacemos todo para no estar moviendo archivos fuera del recurso, evitar pérdida o corrupción de datos.

Evitamos subir y descargar muchos archivos desde nuestras computadoras personales al servidor. Si es necesario meter algo dentro de nuestro servidor lo hacemos a través de la propia herramienta de Jupyter Notebook ya que nos asegura que los archivos lleguen bien haciendo un checksum. Si el recurso es descargable de internet, preferimos usar la consola y usar wget para obtener los datos.

Tener los datos almacenados de esta manera nos permite un acceso rápido y sencillo sin comprometer la seguridad, hablaremos de eso más adelante.

Respecto al acceso a los datos en temas de seguridad, el acceso al sistema entero y a los archivos está restringido por varias capas:

1. Login de Oracle Cloud:
    a. Para acceder a la instancia completamente, es necesario tener el acceso a la cuenta de Oracle Cloud para realizar modificaciones importantes al sistema.
2. SSH
    a. Para acceder de forma completa al sistema operativo y de forma parcial a los archivos, se necesita de una clave privada para autenticarse y realizar cambios críticos al entorno de trabajo.

3. FTP(S)
    a. Para obtener acceso a administración, subida y descarga de archivos por completo, tenemos un acceso especial por FTPS protegido bajo un nombre de usuario, contraseña y certificado SSL, así es como evitamos el robo de datos de alguna persona externa.
4. Jupyter Notebook:
    a. Jupyter Notebook cuenta con varias herramientas: nos permite administrar los archivos, tener acceso a terminales (con acceso limitado) y correr código de forma remota para lo que tengamos que hacer.
    b. Para proteger el entorno, montamos un servidor Apache en el puerto 80, este contiene una página estática que no hace nada realmente, usamos un puerto diferente a 80 para evitar que encuentren fácilmente la página real e intenten hacer algo.
    c. El entorno real se encuentra protegido por HTTPS en el puerto 25565.
    d. Finalmente, para acceder al entorno, se necesita una contraseña para comenzar a usarlo, así que nos sentimos totalmente protegidos de cualquier cosa.

Para asegurarnos de cualquier cosa, en Oracle Cloud se cuenta con una unidad de almacenamiento adicional de respaldo en caso de que ocurra algo, además para estar más protegidos contra cualquier incidente, hacemos copias manuales de seguridad, las descargamos a través de FTP a nuestras computadoras y las subimos a OneDrive y a Google Drive al mismo tiempo.

Por último, para mayor seguridad contamos con un certificado SSL/TLS para los servicios HTTP que no tienen que ver con el entorno de Jupyter, adicionalmente revisamos constantemente los intentos de conexión que somos nosotros y si hay alguna anomalía o inicios de sesión no autorizados que pongan en riesgo la integridad de los datos y el proyecto en general, tenemos listos respaldos para volver a reestablecer todo de la manera más rápida posible, siempre con alternativas sencillas y fáciles de utilizar.

Extracción de datos, limpieza y carga a Oracle Cloud

Para la extracción de datos, sólo tuvimos que descargar el dataset, los datos ya venían previamente limpios, listos para comenzar a trabajar con ellos. Sin señal de datos varios en una sola columna o bien, datos nulos o faltantes.

El único problema fue que desde la terminal de Ubuntu en la nube de Oracle, el programa de wget en Google Drive no funcionaba, pero se solucionó una vez montado todo el entorno de Jupyter, ya que te permite subir, descargar, modificar y crear archivos desde ahí mismo manteniendo la integridad con el uso de checksums. Además, ofrece acceso directo a la terminal si es que necesitamos hacer algo dentro de esa instancia en la nube. Todo está bajo una contraseña y certificados de seguridad para obtener el acceso.

Scripts de configuración utilizados para la instancia en la nube

Los scripts / comandos que usamos para montar nuestro entorno de trabajo fueron los siguientes una vez creada la instancia de Oracle Cloud:

```
# Verificar memoria RAM
free -m

# Verificar espacio de Almacenamiento
df -h

# Actualizar los repositorios de Ubuntu
sudo apt-get update
sudo apt-get upgrade

# Instalar Java 8 (intentamos con el 17 y no funcionó Spark)
sudo apt-get install openjdk-8-jdk-headless -y


# Crear nueva carpeta, descargar Spark y descomprimirlo
mkdir
wget https://downloads.apache.org/spark/spark-3.2.1//spark-3.2.1-bin-hadoop3.2.tgz
tar xf spark-3.2.1-bin-hadoop3.2.tgz

# Instalar Python3, pip y virtualenv
sudo apt-get install python3-pip python3-dev
sudo -H pip3 install --upgrade pip
sudo -H pip3 install virtualenv

# Salirse de la carpeta de Spark y crear un entorno virtual de Python
# Los entornos virtuales permiten correr varias aplicaciones de Python a la vez
cd /
cd /home/
mkdir mc
cd mc
virtualenv mc

# Instalar screen para que el entorno y lo demás no muera el cerrar Putty
sudo apt-get install screen

# Crear una nueva screen
screen

# Correr el entorno virtual
cd /home/
source mc/bin/activate

# Cuando se inicie el entorno aparecerá el nombre del entorno entre paréntesis.
# (mc) ubuntu@fox-arm:/home/mc/DataSet$

# Ahora instalamos Jupyter en ese environment
pip install jupyter

# Duplicamos un archivo de configuración de Spark y lo renombramos
cd /home/mc/spark/spark-3.2.2-bin-hadoop.3.2/conf/
sudo cp spark-env.sh.template spark-env.sh
```

```
# Modificamos el archivo en la parte de hasta abajo y agregamos
# Permite que Spark pueda ser llamado sin problema desde el entorno de Jupyter de forma remota, sin exponer a Spark directamente al internet
sudo nano spark-env.sh
SPARK_MASTER_IP=0.0.0.0
SPARK_LOCAL_IP=0.0.0.0

# Presionamos CTRL + X, Y, Enter

# Regresamos a nuestra carpeta de entorno donde queremos iniciar a Jupyter
# Recomendamos crear una nueva carpeta para no tocar por accidente los archivos del environment de Python y de Spark
mkdir /home/mc/root
cd /home/mc/root

# Oracle Cloud usa IPTables como firewall en sus instancias
# Después de haber desbloqueado el puerto de nuestra preferencia el firewall desde la página de Oracle Cloud, ahora desbloquearemos ese mismo puerto en la instancia.
# xxxx es el puerto, en este caso usaremos 25565
sudo iptables -A INPUT -p tcp --dport xxxx -j ACCEPT
sudo netfilter-persistent save
sudo systemctl restart iptables

# Ahora configuramos el entorno de Jupyter con alguna contraseña, se pedirán los datos de contraseña dos veces y dará una salida indicando que se cambió o si ocurrió algún problema
jupyter notebook password

# Iniciamos el servidor / entorno de Jupyter con
# La IP en 0.0.0.0 es para aceptar cualquier conexión de cualquier puerto
jupyter notebook --no-browser --ip=0.0.0.0 --port=25565

# Saldrá algo como esto, significa que ya tienes a Jupyter y a Spark instalados!
# Si deseas probar, usa el siguiente código en un nuevo notebook

Spark_test.ipynb
-------------------------------------------------------------------------------
#Configuración de Spark con Python
!pip install -q findspark
!pip install pyspark==2.3.0

import os
import findspark
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-arm64"
os.environ["SPARK_HOME"] = "/home/mc/spark/spark-3.2.2-bin-hadoop3.2"

findspark.init()
findspark.find()

from pyspark.sql import SparkSession
spark_session = SparkSession.builder.appName('PySpark_session').getOrCreate()
spark_session
-------------------------------------------------------------------------------
```

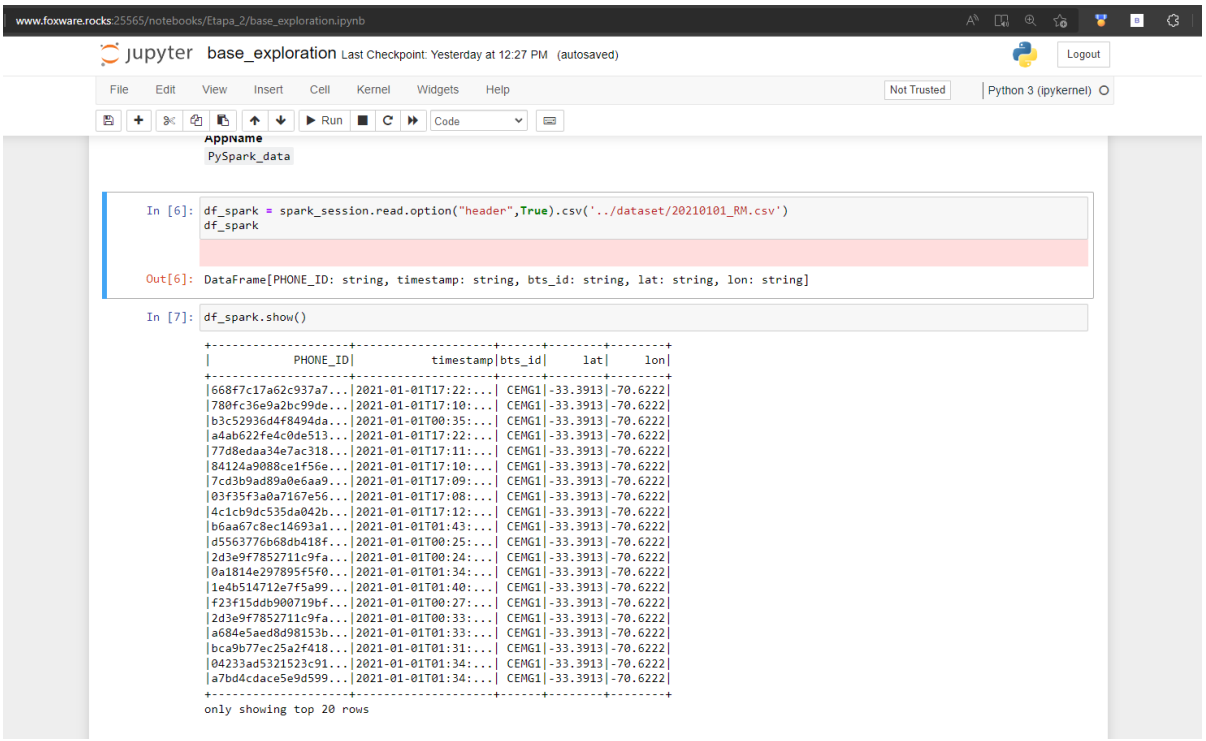
La salida debería ser la siguiente:

SparkSession - in-memory
SparkContext

Spark UI
Version
v3.2.2
Master
local[*]
AppName
PySpark_session

Carga de datos en Spark a través de PySpark

La carga de datos se puede ver aquí, la pequeña carga de datos estará en la misma carpeta del entregable 2 en GitHub.



El dataset se recortó y guardó con Pyspark, son 5000 datos.

```
In [82]: df_spark = spark_session.read.option("header",True).csv('../dataset/20210101_RM.csv')
         df_spark
Out[82]: DataFrame[PHONE_ID: string, timestamp: string, bts_id: string, lat: string, lon: string]

In [86]: df_spark = df_spark.limit(5000)

In [87]: df_spark
Out[87]: DataFrame[PHONE_ID: string, timestamp: string, bts_id: string, lat: string, lon: string]

In [89]: df_spark.write.csv("./recortado.csv")
```

Datos de prueba y entrenamiento

Por otro lado se realizó una separación de datos para prueba y entrenamiento con ayuda de Pandas y Sklearn, el script para dicha separación se encuentra en el GitHub del mismo entregable, dicho script es llamado “Data_divide”.

Dentro de este documento, también se realizó una prueba de entrenamiento de un modelo de clasificación de tipo Árbol de Decisión para que a partir de las coordenadas podamos predecir a qué antena se refiere.

Anteriormente se declaran un modelo de K-folds para entrenar el modelo, en ese caso se declararon 10 K-folds

```
In [5]:
kf = KFold(n_splits=10,random_state = 0, shuffle = True)
kf.get_n_splits() #to get the amount of folds

10
```

Después se declara el modelo y se entrega:

```
In [6]:
model = tree.DecisionTreeClassifier(max_depth = 3)

In [7]:
for train_index, test_index in kf.split(data):
    #print("TRAIN:", train_index, "TEST:", test_index)
    data_train, data_test = data.iloc[train_index], data.iloc[test_index]


    model.fit(data_train[['lat','lon']],data_train['bts_id'].values.ravel())

    y_model = model.predict(data_test[['lat','lon']])

    print("Acuracy_score: ",accuracy_score(data_test['bts_id'],y_model))

Acuracy_score:  0.051504
Acuracy_score:  0.05161
Acuracy_score:  0.05166
Acuracy_score:  0.05163
Acuracy_score:  0.051304
Acuracy_score:  0.052282
Acuracy_score:  0.051798
Acuracy_score:  0.051698
Acuracy_score:  0.051572
Acuracy_score:  0.051322
```

Esta prueba, fue realizada con solo 5 millones de registros (el dataset cuenta con alrededor de 50 millones de registros) y tomó alrededor de 10 minutos en realizar el entrenamiento con un accuracy bastante bajo.

Hecho esto nos pudimos dar cuenta, y como se mencionó previamente, el procesamiento de los datos se realizará con ayuda de Spark para facilitar y hacer más rápido el manejo y uso de los datos.

¿Es necesario usar un enfoque orientado a Big Data?

Dado lo mencionado anteriormente es posible decir que sí es necesario usar un enfoque orientado a Big Data, principalmente por la cantidad de datos. Por otro lado cabe mencionar que, si bien es posible cargarlos de manera correcta el uso de operaciones y funciones alenta el mucho el procesamiento de estos dada la gran cantidad, por lo que se confirma el uso del enfoque hacia Big Data.