**Jueves: Actividades Realizadas**

**Objetivo: Avances iniciales y configuración previa.**

| **Horario** | **Actividad** | **Detalles** |
| --- | --- | --- |
| **Todo el día** | **Crear repositorio en GitHub** | **Subir el repositorio base para controlar versiones del proyecto.** |
| **Todo el día** | **Crear entorno local y configurar librerías** | **Instalación de librerías iniciales para limpieza y procesamiento.** |
| **Todo el día** | **Investigación previa de herramientas a usar** | **Revisar y planificar el uso de herramientas en Azure y Databricks.** |

**Viernes: Preparación del Proyecto**

**Objetivo: Asegurar que todo el entorno y herramientas estén configuradas correctamente.**

| **Horario** | **Actividad** | **Detalles** |
| --- | --- | --- |
| **8:00 AM** | **Revisión de entorno local** | **Validar que el entorno virtual funcione correctamente.** |
| **9:00 AM** | **Subir entorno inicial al repositorio de GitHub** | **Asegurar control de versiones del proyecto.** |
| **10:00 AM** | **Crear cuenta gratuita en Azure** | **Configurar servicios principales (Databricks, Storage Account).** |
| **1:00 PM** | **Crear y montar Storage Account en Databricks** | **Montar el contenedor como /mnt.** |
| **3:00 PM** | **Buscar y descargar el dataset desde Kaggle** | **Usar API Kaggle y guardar datos crudos en local.** |
| **5:00 PM** | **Subir dataset crudo al Storage en Azure** | **Guardar como raw\_data.csv.** |
| **6:00 PM** | **Terminar configuraciones iniciales en Azure** | **Validar que Databricks acceda al Storage.** |

**Sábado: ETL y Preprocesamiento**

**Objetivo: Procesar los datos y normalizarlos en Databricks.**

| **Horario** | **Actividad** | **Detalles** |
| --- | --- | --- |
| **8:00 AM** | **Exploración inicial de datos en local** | **Inspeccionar valores nulos, tipos de datos, distribución.** |
| **10:00 AM** | **Limpieza y normalización local** | **Crear un archivo limpio (cleaned\_data.csv).** |
| **1:00 PM** | **Subir dataset limpio al Storage en Azure** | **Guardar archivo como cleaned\_data.csv.** |
| **2:00 PM** | **Crear una tabla Delta en Databricks** | **Configurar la estructura para almacenamiento escalable.** |
| **3:00 PM** | **Implementar ETL en Databricks** | **Cargar, limpiar y transformar datos usando PySpark.** |
| **5:00 PM** | **Validar resultados de ETL en Databricks** | **Asegurarse de que los datos transformados estén correctamente.** |
| **6:00 PM** | **Documentar las transformaciones en GitHub** | **Subir notebooks y descripción al repositorio.** |

**Domingo: Modelo Predictivo y Video**

**Objetivo: Crear el modelo ML y preparar la presentación final.**

| **Horario** | **Actividad** | **Detalles** |
| --- | --- | --- |
| **8:00 AM** | **Implementar modelo predictivo** | **Entrenar el modelo usando PySpark MLlib o local con Scikit-learn.** |
| **11:00 AM** | **Validar predicciones** | **Generar archivo de predicciones (predictions.csv).** |
| **1:00 PM** | **Crear visualización con Streamlit** | **Usar gráficos para presentar resultados (opcional).** |
| **3:00 PM** | **Grabar el video explicativo** | **Mostrar: entorno, código, resultados, y flujo general.** |
| **5:00 PM** | **Subir el proyecto final a GitHub** | **Archivos, notebooks y video explicativo.** |

**Lunes (entrega)**

* **Asegúrate de que todo esté en GitHub y verifica que el video sea claro y conciso.**

**Diagrama de Flujo del Proyecto**

1. **Inicio**
   * Crear cuenta gratuita en Azure.
2. **Configuración del Entorno**
   * Configurar un **Storage Account** (Data Lake).
   * Crear un **Workspace de Databricks**.
   * Montar el Storage Account en Databricks.
3. **Descarga del Dataset**
   * Buscar un dataset relevante en Kaggle (usando API).
   * Descargar y almacenar el dataset en el Data Lake.
4. **Preparación de Datos**
   * Crear una tabla Delta en Databricks.
   * Dividir los datos en archivos CSV por fecha.
   * Almacenar los CSV en el Storage Account.
5. **Transformaciones en Databricks**
   * Cargar los CSV diarios en la tabla Delta.
   * Reemplazar las fechas existentes con datos actualizados.
6. **Modelo Predictivo**
   * Entrenar un modelo de Machine Learning (ventas u otra predicción).
   * Guardar las predicciones.
7. **Visualización de Resultados**
   * Usar Streamlit para mostrar las predicciones.
8. **Fin**

**Parte 1: Preparación del Entorno**

**1. Crear una Cuenta Gratuita de Azure**

1. Ve a [Azure Free Account](https://azure.microsoft.com/free/) y regístrate.
2. Obtendrás $200 USD en crédito gratuito durante 30 días y acceso a servicios gratuitos como almacenamiento y Databricks.

**2. Buscar un Dataset Relevante en Kaggle**

* Ve a Kaggle Datasets y busca un dataset con datos de ventas o series temporales.
* Ejemplo recomendado: **"Retail Sales Data Set"** (Retail Sales). Este tiene múltiples fechas y es útil para modelos predictivos.

**Parte 2: Flujo de Trabajo en Azure y Databricks**

**1. Crear un Azure Databricks Workspace**

1. En el portal de Azure, busca "Databricks" en el buscador y selecciona **Azure Databricks**.
2. Crea un nuevo workspace con el plan gratuito.
3. Una vez creado, inicia sesión en el workspace de Databricks.

**2. Configurar un Storage Account como Data Lake**

1. En Azure, busca "Storage Accounts" y crea uno:
   * Tipo de cuenta: **StorageV2 (general purpose v2)**.
   * Configura acceso público para que Databricks pueda conectarse.
2. Habilita la jerarquía de directorios (Data Lake Gen2).

**3. Montar el Storage Account en Databricks**

1. Copia las claves de acceso del Storage Account desde Azure.
2. En Databricks, crea un nuevo notebook y usa este código para montar el almacenamiento:

python

Copiar código

storage\_account\_name = "<NOMBRE\_STORAGE>"

storage\_account\_key = "<CLAVE\_STORAGE>"

dbutils.fs.mount(

source=f"wasbs://<nombre\_contenedor>@{storage\_account\_name}.blob.core.windows.net",

mount\_point="/mnt/<punto\_montaje>",

extra\_configs={f"fs.azure.account.key.{storage\_account\_name}.blob.core.windows.net": storage\_account\_key}

)

**4. Crear una Tabla Delta en Databricks**

1. Crea una tabla Delta con la estructura del dataset de Kaggle:

python

Copiar código

from pyspark.sql.types import StructType, StructField, StringType, IntegerType, DateType

schema = StructType([

StructField("date", DateType(), True),

StructField("sales", IntegerType(), True),

StructField("product", StringType(), True)

])

df = spark.createDataFrame([], schema)

df.write.format("delta").save("/mnt/<punto\_montaje>/delta\_table")

**5. Descargar Dataset desde Kaggle**

1. Instala Kaggle en el cluster de Databricks:
   * Ve a **Libraries** > **Install New** > Escribe kaggle.
2. Configura tu API Key de Kaggle en Databricks:
   * Descarga la API Key desde tu perfil de Kaggle y súbela a Databricks.
   * Usa este código para descargar el dataset:

python

Copiar código

import kaggle

kaggle.api.dataset\_download\_files('<dataset\_path>', path='/mnt/<punto\_montaje>/data', unzip=True)

**6. Generar Archivos CSV Diarios**

1. Divide los datos por fecha y guarda cada uno como archivo CSV:

python

Copiar código

from pyspark.sql.functions import col

df = spark.read.csv('/mnt/<punto\_montaje>/data/dataset.csv', header=True)

dates = df.select("date").distinct().collect()

for date in dates:

daily\_data = df.filter(col("date") == date["date"])

daily\_data.write.csv(f"/mnt/<punto\_montaje>/daily/proyecto\_{date['date']}.csv", header=True)

**7. Modificar Fechas en la Tabla Delta**

1. Elimina fechas existentes y actualiza con los datos diarios:

python

Copiar código

delta\_table = DeltaTable.forPath(spark, "/mnt/<punto\_montaje>/delta\_table")

# Eliminar fechas existentes

delta\_table.delete("date is not null")

# Agregar nuevos datos

new\_data = spark.read.csv("/mnt/<punto\_montaje>/daily", header=True)

delta\_table.alias("table").merge(

new\_data.alias("data"),

"table.date = data.date"

).whenNotMatchedInsertAll().execute()

**Parte 3: Modelo Predictivo y Visualización**

**1. Crear el Modelo**

1. Usa PySpark o Scikit-learn para entrenar un modelo predictivo (puedes descargar los datos y entrenar localmente).
2. Entrena el modelo para predecir ventas futuras según los datos históricos.

**2. Visualizar con Streamlit**

1. Guarda las predicciones en un archivo CSV.
2. Crea una aplicación de Streamlit para mostrar los datos:

python

Copiar código

import streamlit as st

import pandas as pd

data = pd.read\_csv("predicciones.csv")

st.line\_chart(data)

**3. Generar Diagrama del Proceso**

1. Usa [LucidChart](https://www.lucidchart.com/) o PowerPoint para diagramar el flujo.

¿Te gustaría empezar con alguna parte en específico? 😊

El tratamiento de datos (ETL o normalización) puede realizarse en diferentes etapas, dependiendo de los recursos y herramientas disponibles. Analicemos las opciones en tu caso:

**1. Tratamiento ETL en Local (Previo al Montaje en Azure)**

Esta opción es útil si prefieres preparar los datos antes de cargarlos al entorno de Azure.

* **Ventajas**:
  + No consumes recursos de Azure durante el preprocesamiento.
  + Mayor control y flexibilidad usando Python con librerías como pandas o pyarrow.
  + Permite detectar problemas de calidad de datos temprano.
* **Proceso**:
  + Descarga el dataset con la API de Kaggle.
  + Realiza limpieza, normalización y transformaciones:
    - Manejo de valores nulos (fillna, eliminación).
    - Normalización de valores (escalado de datos numéricos).
    - Formato de fechas.
    - Transformaciones específicas según las columnas (por ejemplo, categorizar variables).
  + Guarda el resultado en archivos optimizados para su posterior carga (CSV limpio o Parquet).

**2. Tratamiento ETL Directamente en Azure**

Esta opción aprovecha las herramientas de Azure y Databricks para realizar ETL de manera escalable.

* **Ventajas**:
  + Aprovechas la infraestructura de Azure para manejar datasets grandes.
  + Permite trabajar en paralelo y escalar según sea necesario.
  + Integración directa con Data Lake y tablas Delta.
* **Proceso**:
  + Carga el dataset crudo al **Storage Account** en Azure.
  + Usa un **notebook de Databricks** para realizar las transformaciones:
    - Cargar el dataset crudo.
    - Aplicar limpieza y normalización utilizando PySpark o Spark SQL.
    - Guardar los datos tratados como una **tabla Delta** o archivos optimizados (Parquet/CSV).

**¿Cuál opción elegir?**

**Si el dataset es pequeño a mediano (menos de 1 GB):**

* **Haz el preprocesamiento localmente.**
  + No consumirás recursos de Azure innecesariamente.
  + Usa herramientas como pandas para limpiar y normalizar antes de cargarlo.

**Si el dataset es grande o quieres una solución escalable:**

* **Realiza el tratamiento directamente en Azure con Databricks.**
  + Es ideal para grandes volúmenes de datos o procesos repetitivos.
  + Puedes integrar el flujo completo (ETL + modelo ML) en un solo entorno.

**Recomendación para tu proyecto**

* **Empieza con ETL local.** Esto te permitirá familiarizarte con los datos sin gastar recursos.
* Una vez entendido el proceso, **mueve el flujo ETL completo a Databricks en Azure**, donde puedes escalar y optimizar las operaciones.

¿Te gustaría ayuda para un script inicial de ETL en Python o configuración en Azure? 😊

Claro, Juan Pablo! Aquí tienes el paso a paso para comenzar con un flujo ETL **local** y luego migrarlo a **Azure Databricks**.

**ETL Local: Paso a Paso**

1. **Descargar el Dataset desde Kaggle**
   * Asegúrate de tener la API Key de Kaggle configurada:
     1. Ve a tu perfil de Kaggle > **Account** > Descarga el archivo kaggle.json.
     2. Coloca kaggle.json en la carpeta ~/.kaggle/ (en Windows, ponlo en %USERPROFILE%\.kaggle\).
   * Descarga el dataset con Python:

python

Copiar código

import kaggle

kaggle.api.dataset\_download\_files('<dataset\_path>', path='./data', unzip=True)

Reemplaza <dataset\_path> con el nombre del dataset, por ejemplo, "my-dataset-path".

1. **Explorar y Entender los Datos**
   * Usa pandas para cargar el dataset y entender su estructura:

python

Copiar código

import pandas as pd

# Cargar el dataset

df = pd.read\_csv('./data/dataset.csv')

# Exploración básica

print(df.head()) # Primeras filas

print(df.info()) # Información de columnas y tipos de datos

print(df.describe()) # Estadísticas descriptivas

1. **Limpieza y Normalización**
   * **Manejo de valores nulos**:

python

Copiar código

# Reemplazar valores nulos

df.fillna(0, inplace=True)

* + **Formatear columnas**:

python

Copiar código

# Convertir la columna de fecha al formato datetime

df['date'] = pd.to\_datetime(df['date'])

* + **Normalización de datos numéricos**:

python

Copiar código

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

df[['sales']] = scaler.fit\_transform(df[['sales']])

1. **Guardar los Datos Normalizados**
   * Guarda el dataset limpio para usarlo en Azure:

python

Copiar código

df.to\_csv('./data/cleaned\_dataset.csv', index=False)

**Migración a Azure Databricks: Paso a Paso**

**1. Configurar Azure Databricks**

1. Ve al portal de Azure y busca **Databricks**.
2. Crea un **Databricks Workspace** usando la suscripción gratuita.
3. Inicia sesión en Databricks y crea un cluster gratuito.

**2. Cargar el Dataset Limpio a Azure**

1. Ve a **Storage Accounts** en Azure y crea un nuevo almacenamiento:
   * Tipo: **StorageV2 (general purpose v2)**.
   * Activa **Hierarchical Namespace** (Data Lake Gen2).
2. Sube el archivo cleaned\_dataset.csv al contenedor:
   * Usa el portal de Azure o Azure Storage Explorer.

**3. Montar el Storage Account en Databricks**

1. Copia la clave de acceso del Storage Account.
2. Monta el contenedor en Databricks:

python

Copiar código

storage\_account\_name = "<NOMBRE\_STORAGE>"

storage\_account\_key = "<CLAVE\_STORAGE>"

dbutils.fs.mount(

source=f"wasbs://<nombre\_contenedor>@{storage\_account\_name}.blob.core.windows.net",

mount\_point="/mnt/dataset",

extra\_configs={f"fs.azure.account.key.{storage\_account\_name}.blob.core.windows.net": storage\_account\_key}

)

**4. Cargar el Dataset en Databricks**

* Usa PySpark para cargar y procesar los datos:

python

Copiar código

df = spark.read.csv("/mnt/dataset/cleaned\_dataset.csv", header=True, inferSchema=True)

df.show()

**5. Procesamiento ETL en Databricks**

* **Normalización y transformación adicional**:

python

Copiar código

from pyspark.sql.functions import col

# Escalar datos de ventas

df = df.withColumn("sales\_normalized", col("sales") / col("sales").max())

* **Guardar los datos en una tabla Delta**:

python

Copiar código

df.write.format("delta").save("/mnt/dataset/delta\_table")

**Flujo Final**

1. **Descargar y limpiar datos localmente con Python.**
2. **Subir el dataset limpio a Azure Storage.**
3. **Montar el Storage en Databricks y procesar datos con PySpark.**
4. **Guardar los datos como una tabla Delta para uso futuro.**

¿Quieres ayuda con los scripts o una guía para instalar las herramientas necesarias? 😊