AVANCE | FASE II DE PROYECTO

Integrantes:
Andrea Monserrat Ruiz Gomez **- A01794631**
Daniel Acevedo Sainos - **A01795496**
Luis Alejandro Aguilar - **A01795362**
Juan Manuel Rodríguez Mateos – **A01794890**
Marcos Eduardo García Ortiz - **A01276213**
Héctor Raúl Peraza Alvarez - **A01795125**

CONTENIDO

1. Introducción

2. DVC: Control de Versiones de Datos y Seguimiento de Experimentos

3. Registro de Experimentos y Gestión de Modelos Mlflow

4. Conclusiones

1. INTRODUCCIÓN

En este proyecto, nuestro equipo implementó **DVC** y **MLflow** para gestionar el control de versiones de datos y el seguimiento de experimentos en el ciclo de vida de machine learning.

Estas herramientas nos permitieron asegurar la reproducibilidad y eficiencia en todas las etapas del flujo de trabajo, desde el preprocesamiento de datos hasta la gestión de modelos.

2. DVC: CONTROL DE VERSIONES DE DATOS Y SEGUIMIENTO DE EXPERIMENTOS

Para asegurar la reproducibilidad y control sobre los datos utilizados en nuestros experimentos, implementamos **DVC** como herramienta de control de versiones de datos. A través de un flujo de trabajo claramente definido, dividimos el proceso en tres etapas principales: preprocesamiento, entrenamiento y predicción.

- Etapa de preprocesamiento
- Etapa de entrenamiento
- Etapa de predicción

```
import sys
import os
sys.path.append(os.path.abspath(os.path.dirname(__file__)))
import pandas as pd
from config import Config

def load_data():
    """Carga el dataset desde la ruta especificada en config."""
    data = pd.read_csv(Config.DATA_PATH)
    return data

def preprocess_data(data):
    """Aplica las transformaciones necesarias al dataset."""
    # data['Gender'] = data['Gender'] == 'Male'
    data['class'] = data['class'] == 'Positive'

    # for column in data.columns[2:-1]:
    #     data[column] = data[column] == 'Yes'

    return data

def save_processed_data(data):
    """Guarda el dataset procesado en formato CSV en la ruta especificada."""
    processed_path = 'data/processed/diabetes_data_upload.csv'  # Asegurarse de que sea un .csv
    data.to_csv(processed_path, index=False)
    print(f"Datos procesados guardados correctamente en {processed_path}")

if __name__ == '__main__':
    # Cargar los datos
    raw_data = load_data()
    print("Datos crudos cargados correctamente")

    # Preprocesar los datos
    # processed_data = preprocess_data(raw_data)
    print("Datos preprocesados correctamente")

    # Guardar los datos preprocesados
    save_processed_data(raw_data)
```

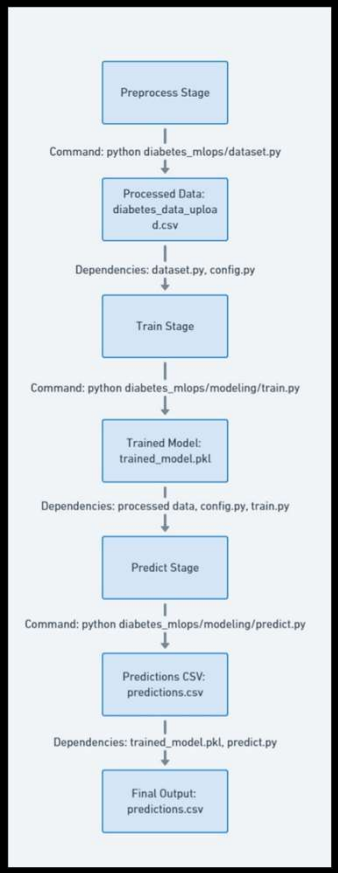
3. REGISTRO DE EXPERIMENTOS Y GESTIÓN DE MODELOS MLflow

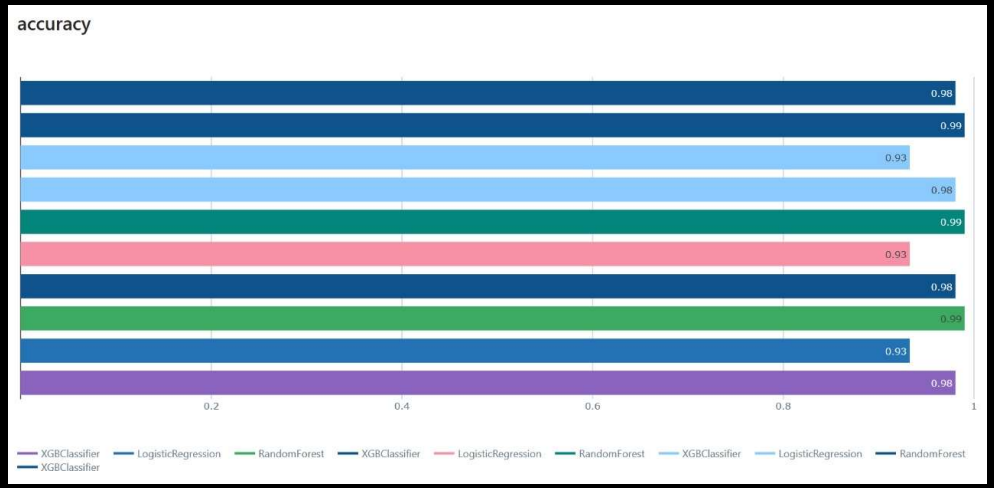
Para el seguimiento de los experimentos y la gestión de los modelos, hemos utilizado **MLflow**, lo que nos permitió llevar un registro detallado de cada ejecución, los parámetros utilizados y las métricas obtenidas.

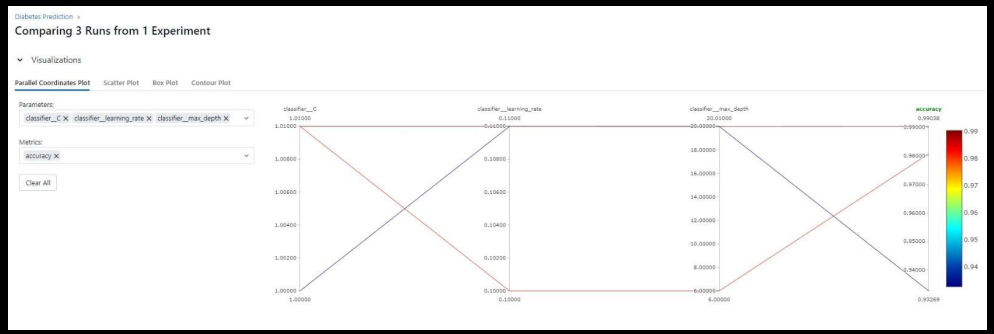
- Seguimiento de experimentos
- Registro de modelos
- Visualización de resultados.

```
import mlflow
import mlflow.sklearn
from sklearn.metrics import accuracy_score, f1_score

# Start an MLflow run for model tracking
with mlflow.start_run(run_name=model_name):
    # Train model
    grid_search.fit(X_train, y_train)

    # Evaluate model
    y_pred = grid_search.best_estimator_.predict(X_test)
    score = f1_score(y_test, y_pred)

    # Log parameters and metrics in MLflow
    mlflow.log_params(grid_search.best_params_)
    mlflow.log_metric("accuracy", accuracy_score(y_test, y_pred))
    mlflow.log_metric("f1_score", score)
    mlflow.sklearn.log_model(grid_search.best_estimator_, "model")
```

4. DIAGRAMAS


Preprocess Stage
Command: python diabetes_mlops/dataset.py
Processed Data: diabetes_data_upload.csv
Dependencies: dataset.py, config.py
Train Stage
Command: python diabetes_mlops/modeling/train.py
Trained Model: trained_model.pkl
Dependencies: processed data, config.py, train.py
Predict Stage
Command: python diabetes_mlops/modeling/predict.py
Predictions CSV: predictions.csv
Dependencies: trained_model.pkl, predict.py
Final Output: predictions.csv


accuracy
0.98
0.99
0.93
0.98
0.99
0.93
0.98
0.99
0.93
0.98
0.2
0.4
0.6
0.8
1
XGBClassifier
LogisticRegression
RandomForest
XGBClassifier
LogisticRegression
RandomForest
XGBClassifier
LogisticRegression
RandomForest
XGBClassifier


Diabetes Prediction
Comparing 3 Runs from 1 Experiment
Visualizations
Parallel Coordinates Plot
Scatter Plot
Box Plot
Contour Plot
Parameters:
classifier_C
classifier__learning_rate
classifier__max_depth
Metrics:
accuracy
Clear All

5. CONCLUSIONES

1. Reproducibilidad: El uso de DVC y MLflow garantizó que cada experimento fuera reproducible, permitiendo rastrear versiones de datos y modelos de manera clara.

2. Seguimiento de Experimentos: MLflow nos facilitó la comparación de diferentes configuraciones y modelos, mejorando la toma de decisiones al registrar parámetros y métricas clave.

3. Visualización Clara: Con MLflow, pudimos visualizar los resultados de cada experimento, lo que nos ayudó a realizar ajustes informados en el proceso de entrenamiento.

4. Gestión de Modelos: Gracias a la gestión de versiones, siempre contamos con un inventario organizado de modelos, lo que nos permitió comparar fácilmente los resultados en cada etapa del proyecto.

6. ANEXOS

1. **Liga GitHub**
   - https://github.com/LAguilar35/diabetes_mlops
2. **Liga Video**
   - https://youtu.be/uvKxORIOT-c