

Prevención de la diabetes

Semana 5

Grupo 22

Integrantes:

Angie Camila Valdes Fuentes

Diana Camila Cordoba Arenas

Robert Kenzo Medina Monsalve

Julian David Hernandez Correcha

Despliegue de soluciones analíticas

UNIVERSIDAD DE LOS ANDES

2024

Prevención de diabetes

Contexto del problema:

La diabetes es una enfermedad crónica de alta prevalencia que representa una carga significativa para los sistemas de salud y las economías a nivel global. La falta de detección y manejo temprano conduce a complicaciones graves y reduce la calidad de vida de los afectados. Actualmente, el desarrollo de modelos predictivos permite identificar de forma anticipada a individuos con alto riesgo de desarrollar diabetes, lo cual es esencial para implementar intervenciones preventivas efectivas. En un contexto organizacional, estas intervenciones pueden enfocarse en programas de educación en nutrición, promoción de actividad física y seguimiento médico personalizado, reduciendo la incidencia de la diabetes y sus complicaciones a largo plazo.

Pregunta de negocio y alcance del proyecto:

Este proyecto tiene como objetivo desarrollar un modelo predictivo que utilice datos de salud, hábitos de vida y factores socioeconómicos para anticipar el riesgo de diabetes en una población determinada. La pregunta de negocio central es: *¿Cómo pueden los modelos predictivos basados en indicadores de salud, hábitos de vida y factores socioeconómicos contribuir a la prevención de la diabetes mediante la predicción precisa del riesgo de la enfermedad?* A través de este modelo, las organizaciones de salud podrán tomar decisiones informadas, asignar recursos de manera eficiente y dirigir programas de intervención personalizados.

Descripción del conjunto de datos a emplear:

La base de datos incluye una variable objetivo y varios factores de riesgo asociados. A continuación, se presentan los hallazgos clave:

- Variable objetivo:
 - Diabetes_binary: Diagnóstico de diabetes (1 = sí, 0 = no). De la muestra total, 35,346 personas padecen diabetes, lo cual representa un 13% de los participantes, mientras que el 87% no presenta diabetes. Este desbalance en los datos sugiere que sería adecuado aplicar técnicas de balanceo en un modelo predictivo para mejorar la precisión en la predicción de casos de diabetes.
- Variables binarias: Las variables como presión arterial alta (HighBP), colesterol alto (HighChol), y consumo de frutas y vegetales (Fruits, Veggies) indican la presencia (1) o ausencia (0) de estas condiciones. Una mayoría de los participantes reporta acceso a atención médica (AnyHealthcare) y actividad física (PhysActivity), mientras que una minoría reporta consumo elevado de alcohol (HvyAlcoholConsump) o haber tenido un derrame cerebral (Stroke).
- Índice de Masa Corporal (BMI): Con una media de 28.38, sugiere una población con tendencia al sobrepeso. La distribución está sesgada hacia la derecha, indicando que una parte significativa tiene sobrepeso u obesidad.
- Salud General (GenHlth): Valorada de 1 a 5, con una media de 2.51, sugiere una percepción de salud general relativamente buena.
- Días de Mala Salud (MentHlth, PhysHlth): La mayoría reporta cero días de mala salud mental y física en el último mes, aunque una minoría presenta una carga significativa de enfermedad.

- Variables Demográficas: Las distribuciones sugieren una variada composición de edad, educación e ingresos. La edad media y avanzada predomina, y niveles educativos más altos se asocian a mayores ingresos.

Correlaciones destacadas con Diabetes_binary:

- GenHlth (0.29) y HighBP (0.26) tienen la mayor correlación positiva con el diagnóstico de diabetes, sugiriendo que una peor salud general y la presencia de hipertensión aumentan la probabilidad de diabetes.
- DiffWalk (0.22) y BMI (0.22) también están asociados positivamente, indicando que las dificultades para caminar y un mayor índice de masa corporal están vinculados a un mayor riesgo de diabetes.
- HighChol (0.20) muestra una correlación menor pero significativa, reflejando el riesgo cardiovascular.


Correlaciones entre variables de salud y demográficas:




- GenHlth y DiffWalk (0.46) y PhysHlth y GenHlth (0.52), resaltando que una peor salud física y problemas de movilidad están asociados a una mala salud general.
- Income y Education (0.45) sugieren que mayores niveles educativos están asociados con mayores ingresos.

Agrupación de Variables: Para facilitar el análisis, se agruparon las categorías de edad e ingreso en rangos más amplios, mejorando la interpretación en futuros modelos.

Fuentes de los modelos:

[prevencion_diabetes_dsa_g22](#) / [project-root](#) / [model](#) / [models.py](#) 

 Stornwar replacing the SVC model with xgboost

Code Blame 23 lines (22 loc) · 775 Bytes  Code 55% faster with Gi Raw  

```

1  from sklearn.ensemble import RandomForestClassifier
2  from sklearn.linear_model import LogisticRegression
3  from xgboost import XGBClassifier
4  from config.core import model_config
5
6  models = {
7      "RandomForest": RandomForestClassifier(
8          n_estimators=model_config.n_estimators,
9          max_depth=model_config.max_depth,
10         random_state=model_config.random_state
11     ),
12     "LogisticRegression": LogisticRegression(
13         random_state=model_config.random_state
14     ),
15     "XGBoost": XGBClassifier(
16         n_estimators=model_config.xgb_n_estimators,
17         max_depth=model_config.xgb_max_depth,
18         learning_rate=model_config.xgb_learning_rate,
19         use_label_encoder=False,
20         eval_metric="logloss",
21         random_state=model_config.random_state
22     )

```

Cada modelo utiliza parámetros específicos definidos en model_config, como n_estimators y max_depth, ajustados para optimizar su rendimiento. La implementación de XGBClassifier incluye parámetros adicionales como learning_rate y eval_metric configurado en "logloss". Este enfoque permite comparar el rendimiento de los tres modelos en términos de precisión y tiempo de ejecución, seleccionando finalmente el que mejor se adapta a las necesidades del proyecto.

Modelos desarrollados y su evaluación:

Se evaluaron tres modelos de clasificación supervisada (XGBoost, Regresión Logística y RandomForest) aplicados a un conjunto de datos cuya división para entrenamiento y prueba fue del 70% y 30%, respectivamente. El objetivo es identificar el modelo con mejor rendimiento basado en métricas de precisión, recall, área bajo la curva ROC (AUC) y tiempo de ejecución.

A continuación, se detallan los principales hiperparámetros ajustados para cada modelo y como se realizó el porcentaje de base que se usó para test:

```
21         # Parámetros para RandomForest
22         n_estimators: int = 100
23         max_depth: int = 10
24
25         test_size: float = 0.3
26
27         # Parámetros para XGBoost
28         xgb_n_estimators: int = 100
29         xgb_max_depth: int = 5
30         xgb_learning_rate: float = 0.1
```

También se midieron los tiempos de inicio y finalización para cada modelo, así como su duración total de ejecución:

Run details

Run ID:	2813a220910749d0ac5bec51b0d9f0f3	db53a0411ecb4799a95c44932da7f3f1	2968da295af04aaa87a2e915e57c96d3
Run Name:	XGBoost	LogisticRegression	RandomForest
Start Time:	2024-11-10 21:06:42	2024-11-10 21:06:40	2024-11-10 21:06:26
End Time:	2024-11-10 21:06:45	2024-11-10 21:06:42	2024-11-10 21:06:40
Duration:	3.3s	2.2s	13.6s

Las métricas de evaluación utilizadas para comparar los modelos fueron precisión, AUC, precisión y recall. Los resultados se presentan en la siguiente imagen:

Logistic Regression fue el modelo más rápido, completando su ejecución en 2.2 segundos, seguido por **XGBoost** con 3.3 segundos.

RandomForest tuvo el tiempo de ejecución más largo, tardando 13.6 segundos, lo cual podría influir en su viabilidad en casos donde el tiempo de respuesta es crítico.

Para comparar y evaluar la calidad de los modelos se usaron las métricas precisión, AUC, precisión y recall. Los resultados se presentan en la siguiente tabla:

☐ Show diff only

model_name	XGBoost	LogisticRegression	RandomForest
------------	---------	--------------------	--------------

▼ Metrics

☐ Show diff only

accuracy	0.868	0.866	0.866
auc	0.832	0.825	0.828
precision	0.573	0.545	0.605
recall	0.157	0.161	0.085

Exactitud (Accuracy): XGBoost presenta el mayor valor de exactitud con 0.868, seguido de Logistic Regression y RandomForest con 0.866, lo que indica un rendimiento muy similar entre los tres modelos en términos de clasificación global.

Área Bajo la Curva (AUC): XGBoost también se destaca en esta métrica con un valor de 0.832, lo cual sugiere una buena capacidad para distinguir entre las clases. RandomForest y Logistic Regression tienen valores de 0.828 y 0.825, respectivamente.

Precisión (Precision): RandomForest tiene el mejor valor de precisión (0.605), seguido por XGBoost (0.573) y Logistic Regression (0.545). Esto indica que RandomForest tiene menos falsos positivos, pero no necesariamente es el mejor modelo en general.

Recall: Logistic Regression tiene un recall levemente mayor (0.161) en comparación con XGBoost (0.157) y RandomForest (0.085), lo que indica una mayor capacidad para capturar correctamente las instancias positivas.

Observaciones y conclusiones sobre los modelos

Logistic Regression podría ser una opción viable cuando se requiere un tiempo de ejecución muy bajo y el recall es un factor importante, aunque a expensas de una precisión ligeramente menor.

RandomForest mostró una alta precisión, lo cual es útil en casos donde es importante reducir los falsos positivos, pero su bajo recall y mayor tiempo de ejecución lo hacen menos atractivo en comparación con los otros modelos.

XGBoost es el **modelo escogido** debido a su buen balance entre exactitud, AUC y tiempo de ejecución. Su alto valor de AUC sugiere una buena capacidad para distinguir entre las clases, y su tiempo de ejecución es razonable en comparación con RandomForest. Sin embargo también se identificó que se puede realizar un modelo que balancee un poco la población y corre nuevamente en este caso un Xgboost y comparar resultados.

Fuentes del tablero:

prevencion_diabetes_dsa_g22 / project-root / app / api / main.py

Code Blame 47 lines (39 loc) · 1.52 KB Code 55% faster with GitHub Copilot

```
3 from app.api.v1.endpoints import health, predict
4 from app.api.v1.config import settings
5 from fastapi.responses import HTMLResponse
6 from pathlib import Path
7 from fastapi.staticfiles import StaticFiles
8 from pydantic_settings import BaseSettings
9
10 app = FastAPI(
11     title=settings.PROJECT_NAME,
12     openapi_url=f"{settings.API_V1_STR}/openapi.json"
13 )
14
15 # Monta la carpeta 'dashboard' para servir archivos estáticos
16 app.mount("/dashboard", StaticFiles(directory="dashboard"), name="dashboard")
17
18 # Ruta para servir el archivo dash.html en la raíz
19 @app.get("/", response_class=HTMLResponse)
20 async def read_dashboard():
21     html_path = Path("dashboard/dash.html")
22     return HTMLResponse(content=html_path.read_text(), status_code=200)
23
```

```
24 # Configuración de CORS
25 if settings.BACKEND_CORS_ORIGINS:
26     app.add_middleware(
27         CORSMiddleware,
28         allow_origins=[str(origin) for origin in settings.BACKEND_CORS_ORIGINS],
29         allow_credentials=True,
30         allow_methods=["*"],
31         allow_headers=["*"],
32     )
33
34 # Incluye las rutas
35 app.include_router(health.router, prefix=settings.API_V1_STR)
36 app.include_router(predict.router, prefix=settings.API_V1_STR)
37
38 class Settings(BaseSettings):
39     PROJECT_NAME: str = "Your Project Name"
40     API_V1_STR: str = "/api/v1"
41     BACKEND_CORS_ORIGINS: list[str] = ["*"] # or specific origins as a list
42
43 settings = Settings()
44
45 if __name__ == "__main__":
46     import uvicorn
47     uvicorn.run(app, host="0.0.0.0", port=8001, log_level="info")
```

Se utiliza FastAPI para crear una aplicación web que despliega un tablero para la prevención de diabetes. Se configuran rutas para servir archivos estáticos, como la carpeta "dashboard", que contiene el archivo HTML del tablero (dash.html). Además, se emplea configuración de CORS para permitir el acceso desde distintos orígenes, asegurando que la API pueda interactuar con el frontend sin restricciones.

El archivo también define rutas (health y predict) para acceder a los endpoints de salud y predicción, fundamentales para evaluar el riesgo de diabetes en base a los datos ingresados por el usuario. La clase Settings establece configuraciones generales, como el nombre del proyecto y la configuración de CORS, y utiliza Uvicorn para ejecutar la aplicación en un servidor local. Este diseño modular y seguro permite servir el tablero y gestionar las predicciones de manera efectiva.

Descripción del tablero desarrollado y la funcionalidad que éste ofrece

El Tablero de Prevención de la Diabetes es una herramienta interactiva diseñada para evaluar el riesgo de desarrollar diabetes en función de varios factores de salud y estilo de vida. El tablero permite a los usuarios ingresar datos específicos sobre su estado de salud actual y sus hábitos, con el objetivo de proporcionarles una estimación del riesgo que tienen de desarrollar diabetes.

Como se observa en las imágenes el tablero incluye una sección en la que el usuario puede ingresar sus datos personales y de salud, cada entrada tiene un menú desplegable o un campo numérico adecuado para capturar la información de manera fácil y estructurada. Una vez ingresados todos los datos, el usuario puede hacer clic en el botón "Evaluar riesgo". Esto activa una función que evalúa el riesgo de padecer diabetes según los datos ingresados a través de una API que emplea el modelo seleccionado.

La sección de resultados muestra al usuario su nivel de riesgo, clasificado como "**Alto**" o "**Bajo**":

- Si el riesgo es **alto**, se muestra un mensaje de advertencia en color rojo que recomienda al usuario consultar a un profesional médico.
- Si el riesgo es **bajo**, el tablero muestra un mensaje positivo en color verde, alentando al usuario a mantener sus buenos hábitos de salud.

Prevención de la Diabetes

Ingresa tus datos de salud

Presión arterial alta
Sí

Colesterol alto
Sí

Chequeo de colesterol (últimos 5 años)
Sí

Índice de masa corporal (BMI)

Fumador
Sí

Derrame cerebral
Sí

Enfermedad del corazón o ataque al corazón
Sí

Actividad física
Sí

Consumo de frutas
Sí

Consumo de vegetales
Sí

Consumo pesado de alcohol

Resultados

Bajo

¡Excelente! Tu evaluación es favorable. Sigue cuidando tu salud para mantener estos resultados.

Acceso a atención médica
Sí

No visitó al doctor por costo
Sí

Salud general (1-5)

Días de mala salud mental (último mes)

Días de mala salud física (último mes)

Dificultad para caminar sin ayuda
Sí

Sexo
Femenino

Categoría de edad (1, 2, 3, o 4)

Nivel de educación (1, 2, 3 o 4)

Categoría de ingreso (1, 2 o 3)

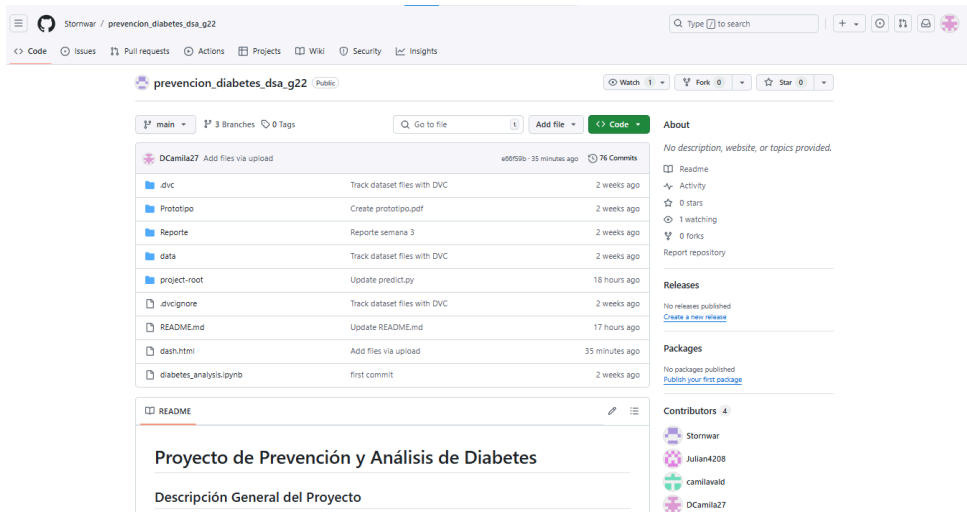
Resultados

Alto

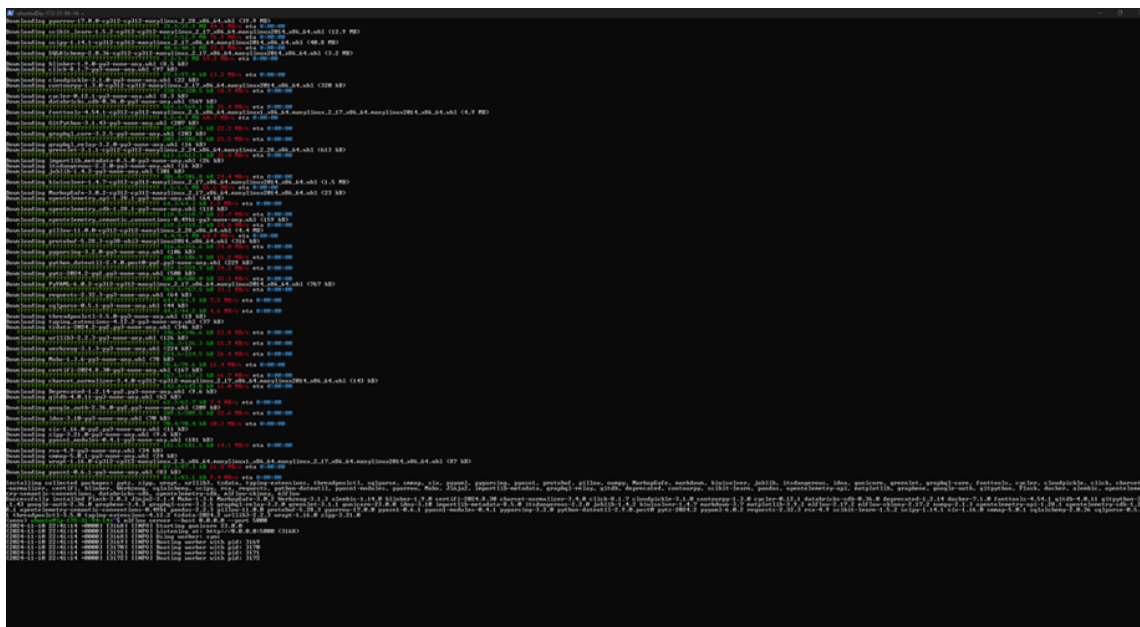
Tu salud es lo primero. Te recomendamos consultar a un profesional médico para un seguimiento adecuado y personalizado.

Repositorio Git

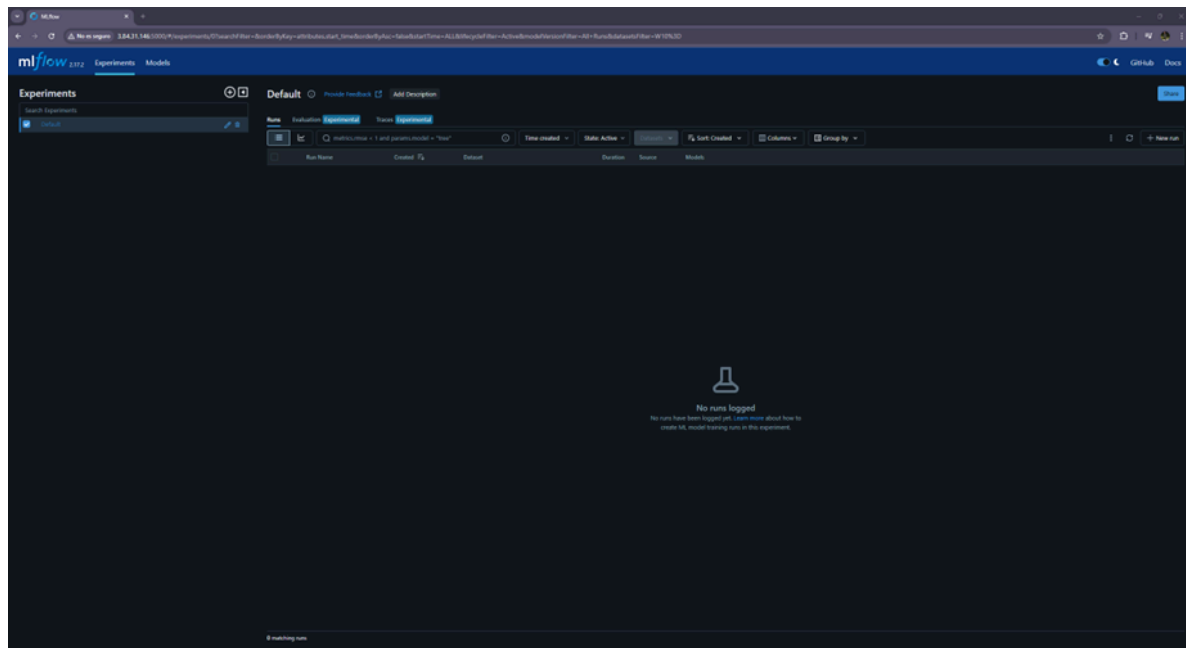
Link: https://github.com/Stornwar/prevencion_diabetes_dsa_g22.git



Experimentos registrados en ML Flow



Se registraron y evaluaron múltiples experimentos de modelos predictivos en **MLflow** para comparar su desempeño. Los modelos incluidos son **XGBoost**, **Regresión Logística**, y **Random Forest**. Cada ejecución registra métricas clave como **precisión**, **AUC** (Área Bajo la Curva), **precisión** y **recall**, lo que permite analizar y seleccionar el modelo óptimo basado en su capacidad para predecir el riesgo de diabetes.



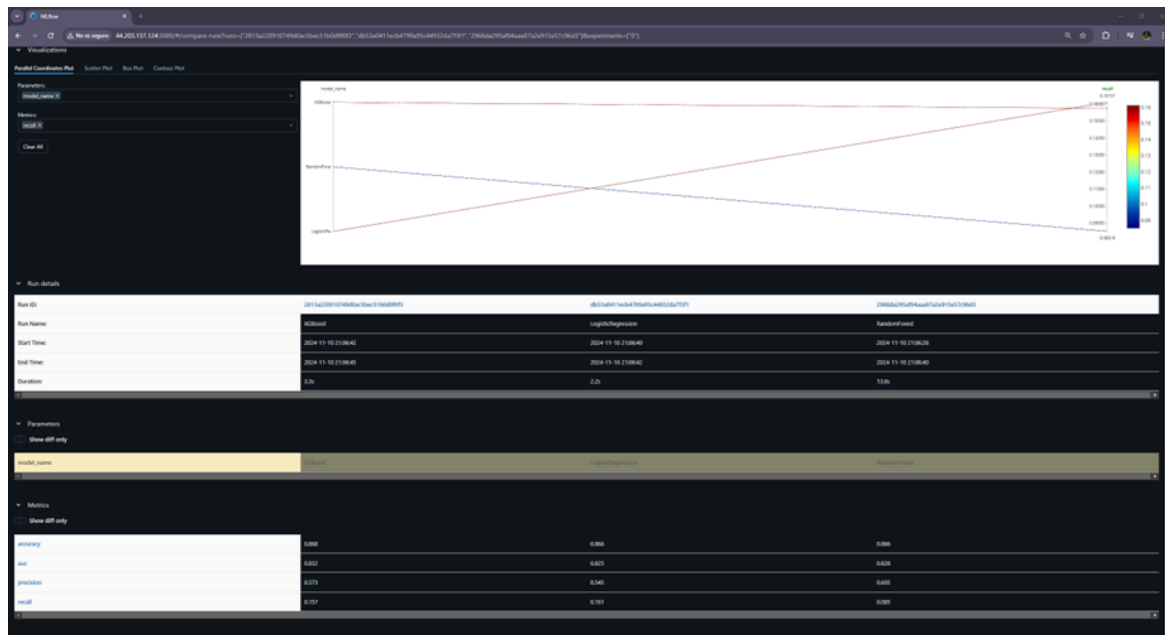
Experimentos con los diferentes modelos

Se decidió implementar un modelo de regresión logística, random forest y el de XGboost de los tantos que se intentaron.

The screenshot shows the mlflow Experiments page with 12 runs logged. The table displays the following data:

Run Name	Created	Duration	Source	Models
Default	20 seconds ago	1.7s	train.py	sklearn
logistic-regression	20 seconds ago	2.3s	train.py	sklearn
random-forest	30 seconds ago	11.6s	train.py	sklearn
XGBoost	13 minutes ago	-	train.py	-
logistic-regression	13 minutes ago	2.7s	train.py	sklearn
random-forest	13 minutes ago	12.4s	train.py	sklearn
XGBoost	30 minutes ago	-	train.py	-
logistic-regression	31 minutes ago	3.5s	train.py	sklearn
random-forest	31 minutes ago	14.3s	train.py	sklearn
XGBoost	1 hour ago	-	train.py	-
logistic-regression	1 hour ago	2.5s	train.py	sklearn
random-forest	1 hour ago	13.7s	train.py	sklearn

A continuación, la comparación de los 3 modelos:



Como se observa en la comparación, el XGboost es el indicado por ser el mejor en las 3 métricas.

El tablero de **MLflow** muestra el historial de los experimentos, incluyendo el tiempo de ejecución de cada modelo. Los resultados destacan que **XGBoost** logra el mejor balance entre precisión y AUC, siendo elegido como el modelo más adecuado. Este registro de experimentos facilita la trazabilidad y permite un análisis visual detallado de las métricas, ayudando en la optimización y validación del modelo final.

Reporte de trabajo en equipo

Merge pull request #2 from Stornwar/Reporte Pull request merge	...
Julian4208 pushed 2 commits to main • 1e76045...62b7cb9 • 21 seconds ago	
Reporte	...
Julian4208 pushed 1 commit to Reporte • 1e76045...476856b • 50 seconds ago	
Create prototipo.pdf	...
Julian4208 created Reporte • 1e76045 • 2 minutes ago	
Create prototipo.pdf	...
DCamila27 pushed 1 commit to main • 71342f4...1e76045 • 6 minutes ago	
Track dataset files with DVC	...
Stornwar pushed 2 commits to main • 478de28...71342f4 • 42 minutes ago	
Merge pull request #1 from Stornwar/README Pull request merge	...
camilavald pushed 2 commits to main • 0e6f044...478de28 • 1 hour ago	
Create README.md	...
camilavald pushed 1 commit to README • 0e6f044...1680759 • 1 hour ago	
first commit	...
camilavald created README • 0e6f044 • 1 hour ago	
first commit	...
Stornwar created main • 0e6f044 • 1 hour ago	

El equipo trabajó en los siguientes puntos

1. **Configuración del Repositorio:** Organicé el repositorio del proyecto, definiendo una estructura ordenada y configurando el entorno de desarrollo para facilitar el trabajo en equipo.
2. **Ajustes en la Instancia:** Implementé y optimicé la instancia en la nube, asegurando que todas las dependencias necesarias estuvieran correctamente instaladas.
3. **Ejecución del Modelo:** Ejecuté el modelo de predicción de diabetes, realizando ajustes de hiperparámetros y validaciones para mejorar su rendimiento.
4. **Documentación:** Documenté el proceso y el uso de scripts, asegurando que el equipo tenga una guía clara para futuras referencias y reproducibilidad del proyecto.
5. Creación del tablero en HTML
6. Desarrollo del informe