.UBAfiuba
FACULTAD DE INGENIERÍA

INFORME

Aprendizaje Automático

Integrantes:

Nombre	Padrón	Mail
Clara Ruano Frugoli	106835	cruano@fi.uba.ar
Francisco Orquera Lorda	105554	forqueral@fi.uba.ar
Francisco Ezequiel Martínez	108460	femartinez@fi.uba.ar
Carolina Di Matteo	103963	cdimatteo@fi.uba.ar
Camila General	105552	cgeneral@fi.uba.ar

Introducción

Objetivo del Trabajo

El objetivo de este trabajo práctico fue desarrollar e implementar un pipeline de Machine Learning de punta a punta para solucionar un problema real de predicción de diabetes. A diferencia de un enfoque puramente automatizado, este trabajo se centra en un proceso deliberado que abarca desde la exploración de datos y la selección comparativa de modelos hasta el entrenamiento, la evaluación y el despliegue de un algoritmo específico, transparente e interpretable.

Problemática Escogida

La diabetes es una enfermedad crónica con una prevalencia global en aumento. Su detección temprana es fundamental para prevenir complicaciones graves. El aprendizaje automático ofrece una oportunidad para crear herramientas de soporte que ayuden a los profesionales de la salud a identificar pacientes en riesgo a partir de datos clínicos estándar.
Esta problemática es particularmente idónea para ser resuelta con técnicas de Machine Learning por varias razones fundamentales. En primer lugar, el desarrollo de la diabetes es un proceso **multifactorial**, donde el riesgo no depende de una única variable, sino de la compleja interacción entre múltiples indicadores clínicos y demográficos (como el nivel de glucosa, el índice de masa corporal, la edad, etc.). Los algoritmos de aprendizaje automático son especialmente eficaces para **identificar patrones sutiles y relaciones no lineales** en conjuntos de datos complejos, patrones que a menudo no son evidentes para el análisis humano tradicional.

La implementación de un modelo predictivo en este contexto trae consigo beneficios significativos:

- **Apoyo a la Detección Temprana y Prevención:** El principal valor del modelo es su capacidad para funcionar como un sistema de alerta temprana. Permite estratificar a los pacientes según su nivel de riesgo antes de que la enfermedad se manifieste clínicamente, facilitando intervenciones preventivas (como cambios en el estilo de vida o un monitoreo más frecuente) que pueden retrasar o incluso evitar su aparición.
- **Soporte Objetivo para la Decisión Clínica:** La herramienta no busca reemplazar el juicio del médico, sino complementarlo. Ofrece una evaluación de riesgo objetiva, consistente y estandarizada, que sirve como una segunda opinión basada en datos. Esto es especialmente útil para confirmar diagnósticos o para identificar casos límite que requieran una investigación más profunda.
- **Eficiencia y Escalabilidad:** Una vez entrenado, el modelo puede evaluar el perfil de riesgo de un gran número de pacientes de manera casi instantánea. Esto lo convierte en una herramienta muy valiosa para la salud pública, permitiendo realizar tamizajes (screenings) a gran escala de forma eficiente y a bajo costo.

Marco Teórico: Machine Learning

¿Qué es machine learning?

Machine learning es un subcampo de la IA junto con la ciencia de los datos. Su objetivo es comprender la estructura de los datos y ajustarlos a modelos que puedan ser entendidos y utilizados por ingenieros y agentes del aprendizaje automático en diferentes áreas de trabajo.

¿Cómo funciona el machine learning?

El machine learning se diferencia de la programación tradicional porque, en lugar de seguir instrucciones fijas, los algoritmos aprenden automáticamente a partir de datos. Así, mediante el uso de métodos estadísticos, pueden identificar patrones y generar modelos que predicen resultados o toman decisiones sin que cada paso esté explícitamente programado. Una versión más avanzada de este enfoque son las redes neuronales.

¿Cuáles son los distintos métodos del machine learning?

El machine learning tiene tareas en categorías generales, las cuales provienen del aprendizaje recibido o de la retroalimentación dada al sistema.

Los dos métodos de machine learning más adoptados son:

- Aprendizaje supervisado: entrena algoritmos basados en datos de entrada y salida de ejemplo que etiquetan los humanos.
- Aprendizaje no supervisado: proporciona al algoritmo de machine learning datos no etiquetados que le permiten encontrar la estructura dentro de sus datos de entrada.

En nuestro caso, tenemos una problemática de aprendizaje supervisado, donde contamos con un conjunto de datos que incluye las características o variables predictoras (como edad, índice de masa corporal, nivel de glucosa, entre otras) junto con la etiqueta asociada (en este caso, si la persona es diabética o no), y el objetivo es que el modelo aprenda a predecir correctamente la probabilidad de que una nueva persona tenga diabetes a partir de sus características.

Algoritmos de machine learning

Los algoritmos son la parte computacional de un proyecto de aprendizaje automático. Una vez entrenados, los algoritmos producen modelos con una probabilidad estadística de responder a una pregunta o lograr un objetivo. En nuestro caso, predecir la probabilidad de que una persona tenga diabetes.

Algunos de los algoritmos más comunes en el aprendizaje automático son los siguientes:

- **Redes neuronales**: imitan el cerebro humano mediante capas de nodos que aprenden relaciones entre los datos.
  Regresión lineal: ajusta una línea a los datos para predecir valores numéricos (como estimar peso a partir de estatura).
  Regresión logística: para predecir resultados binarios, como si alguien es diabético o no, o si un correo es spam.
  Agrupamiento (clustering): agrupa datos sin etiquetar en clústeres con características similares, como clasificar frutas por tipo.

- **Árboles de decisión**: siguen reglas tipo "si-entonces" para llegar a una predicción.
- **Bosques aleatorios**: combinan muchos árboles de decisión para decisiones más precisas y menos sesgadas.

Herramientas de AutoML

Las herramientas de AutoML automatizan gran parte del proceso de desarrollo de modelos de machine learning, facilitando tareas como la selección de algoritmos, ajuste de hiperparámetros y preprocesamiento de datos. Esto reduce la necesidad de conocimientos técnicos profundos, acelera el desarrollo y mejora la accesibilidad para usuarios no expertos. Sin embargo, presentan algunas desventajas, como la menor capacidad de personalización, la posible falta de transparencia en los modelos generados (cajas negras) y dependencia de la plataforma, lo que puede limitar el control sobre el proceso y la interpretabilidad de los resultados.

Solución implementada

Resumen

La solución se construyó sobre una arquitectura cliente-servidor, donde un modelo de Machine Learning personalizado se despliega a través de una API REST y es consumido por una interfaz de usuario interactiva.

Dataset Utilizado: Se empleó un dataset público de Kaggle sobre diabetes, el cual incluye variables predictoras como la edad, el índice de masa corporal y el nivel de glucosa.
Metodología y Backend (Modelo de Machine Learning): Se desarrolló un pipeline personalizado en Python utilizando librerías como PyCaret y Scikit-learn. El proceso fue el siguiente:

1. **Comparación y Selección de Modelos**: Se utilizó la biblioteca PyCaret para entrenar y evaluar rápidamente 14 algoritmos de clasificación distintos. Se analizaron métricas como Accuracy, AUC, Recall y Precisión para obtener una visión comparativa del rendimiento potencial de cada modelo.
2. **Criterio de Selección**: Para una aplicación clínica de este tipo, se priorizó el **Recall** (para minimizar los falsos negativos y no omitir pacientes en riesgo) y la **Precisión** (para reducir las falsas alarmas). La **Regresión Logística** destacó por ofrecer un excelente equilibrio entre estas métricas, además de su alta interpretabilidad y eficiencia.
3. **Pipeline de Entrenamiento Final**: Con el modelo ya seleccionado, se construyó un pipeline de entrenamiento detallado usando Scikit-learn. Este incluyó la selección de características (descartando variables de baja correlación), el escalado de datos con StandardScaler y el entrenamiento del modelo final de Regresión Logística.
4. **Despliegue**: El modelo entrenado y el objeto scaler fueron serializados con joblib y desplegados a través de una API RESTful construida con **FastAPI**.

Frontend (Aplicación Web):
- **Páginas del sistema**: La aplicación web cuenta con una interfaz donde el personal médico puede ingresar los datos clínicos de un paciente en un formulario. Una vez enviados, el sistema se comunica con la API del modelo y muestra el resultado de la predicción en tiempo real.
- **Tecnologías**: Se utilizaron **React** y **Next.js** para el desarrollo del frontend. Estas tecnologías fueron elegidas por su eficiencia, su capacidad para crear interfaces web interactivas y modernas, y su soporte para renderizado híbrido.

Descripción Detallada: Proceso de ML

Planificación

En la planificación del proyecto, se definió el alcance y las métricas de éxito para la predicción de diabetes. Para un problema clínico de esta naturaleza, se identificó que era fundamental gestionar el compromiso entre dos tipos de errores críticos:

- Minimizar los Falsos Negativos (pacientes no detectados), lo que exige un Recall alto.
- Limitar los Falsos Positivos (pacientes sanos mal clasificados), lo que requiere una buena Precisión.

Por lo tanto, el criterio de éxito fue seleccionar el modelo que ofreciera el mejor balance posible entre un Recall elevado y una Precisión robusta, garantizando así una solución clínicamente relevante y segura.

Disponibilidad de los datos

La disponibilidad de los datos es clave para entrenar un modelo, ya que es necesario contar con suficientes registros de calidad. En nuestro caso, contamos con un dataset clínico ya preparado que incluye una cantidad suficiente de datos históricos y variables relevantes —como edad, índice de masa corporal, glucosa, entre otras— que son completas y con pocos valores faltantes, por lo que no es necesario adquirir más información.

Aplicabilidad

La aplicabilidad hace referencia a si la solución es útil para resolver el problema planteado o mejorar el proceso actual, y si el machine learning es la herramienta adecuada.

En nuestro proyecto, la solución es directamente aplicable en un entorno clínico. El objetivo es transformar datos que se recolectan de manera rutinaria en chequeos médicos en una evaluación de riesgo instantánea. Esto se logra a través de una arquitectura moderna donde un backend desarrollado en **FastAPI** sirve el modelo de Regresión Logística, y es consumido por un frontend interactivo construido con **React y Next.js**. El resultado es una herramienta web accesible que permite al personal médico ingresar los parámetros de un paciente y obtener una clasificación de riesgo inmediata, apoyando así la toma de decisiones.

Robustez y escalabilidad

La robustez y escalabilidad miden qué tan fiable es el sistema y si puede crecer sin perder desempeño. En nuestro proyecto:

- **Robustez**: La fiabilidad del modelo se fundamenta en el riguroso proceso de validación implementado. Al evaluar su rendimiento sobre un conjunto de datos de prueba que no vio durante el entrenamiento y analizar métricas clínicas clave (Recall y Precisión), se asegura un entendimiento claro de su comportamiento en escenarios realistas.
- **Escalabilidad**: La arquitectura está diseñada para crecer de manera eficiente. El backend, desarrollado en FastAPI, puede ser desplegado en contenedores y escalar horizontalmente para gestionar un alto volumen de solicitudes. Por su parte, el frontend en React y Next.js es inherentemente escalable en cualquier entorno de nube moderno.

El tiempo de respuesta final del sistema integrado depende de la configuración de la infraestructura de despliegue y podría requerir optimizaciones para garantizar alta disponibilidad.

Explicabilidad

Uno de los objetivos centrales de este proyecto fue garantizar un alto grado de explicabilidad. Para lograrlo, se seleccionó deliberadamente el modelo de Regresión Logística, no solo por su rendimiento sino también por su naturaleza interpretable. Las predicciones de este modelo se basan en una combinación ponderada de las variables de entrada, lo que permite entender cómo cada factor contribuye al resultado.

Además, se realizó un análisis de importancia de variables (feature importance) para cuantificar el impacto de cada predictor. Esto permite que el personal médico no solo reciba una clasificación de riesgo, sino que también comprenda qué factores clínicos (como el nivel de glucosa o el índice de masa corporal) influyeron más en la decisión del modelo, aumentando la confianza y el valor de la herramienta como un verdadero sistema de apoyo.

Disponibilidad de recursos

La disponibilidad de recursos incluye tanto los recursos informáticos (cómputo, almacenamiento, red) como el personal calificado. Para este proyecto, el entrenamiento del modelo se realizó utilizando bibliotecas de Python de código abierto como PyCaret y Scikit-learn, lo que permitió un desarrollo eficiente en computadoras locales sin depender de costosos servicios de nube especializados.

Preparación de datos

Recogida de datos y etiquetado

Este paso consiste en obtener los datos necesarios y asegurarse de que estén correctamente etiquetados para el entrenamiento del modelo. Para este proyecto se utilizó un dataset público y ya etiquetado de la plataforma Kaggle.

Limpieza de datos

En esta etapa se corrigen datos faltantes, se eliminan valores atípicos y se reduce el ruido para mejorar la calidad del dataset. Se suele crear una canalización para automatizar estas tareas y validar la calidad de los datos.

En nuestro proyecto, el dataset utilizado era de alta calidad y no presentaba valores faltantes significativos. El principal paso de esta fase fue la **selección de características** (feature selection). Basado en el análisis exploratorio, donde se observó una baja correlación con la variable objetivo, se tomó la decisión de descartar las columnas BloodPressure y SkinThickness. Este paso fue crucial para simplificar el modelo y centrar el entrenamiento en las variables más influyentes.

Ingeniería de modelos

En esta fase, se utiliza toda la información de la fase de planificación para construir y entrenar un modelo de machine learning. Esto incluye el seguimiento de las métricas del modelo, la garantía de escalabilidad y robustez, y la optimización de los recursos. El foco está en la arquitectura de los modelos, la calidad del código, los experimentos de machine learning y el entrenamiento.

En nuestro caso, la ingeniería del modelo se abordó desde dos frentes: la arquitectura del sistema y el pipeline de entrenamiento.

1. **Arquitectura del Sistema**: se definió una arquitectura cliente-servidor. Un backend desarrollado en FastAPI se encarga de servir el modelo de machine learning, como fue explicado anteriormente.
2. **Pipeline de Entrenamiento**: se implementó un pipeline explícito con Scikit-learn. Primero, los datos se dividieron en conjuntos de entrenamiento y prueba de forma estratificada para mantener la proporción de clases. Luego, se entrenó el modelo de Regresión Logística seleccionado, utilizando el parámetro class_weight='balanced' para manejar el desbalance de clases. Finalmente, el modelo entrenado y el escalador de datos fueron serializados y guardados como artefactos (.pkl) utilizando joblib, dejándolos listos para su implementación en la API.

Evaluación del modelo

Una vez finalizado el modelo, se evalúa su desempeño usando diversas métricas sobre un conjunto de datos de prueba, y se involucra a expertos para revisar posibles errores en las predicciones. Se prueba su robustez con datos reales y aleatorios, asegurando además que la velocidad de inferencia sea suficiente para su aplicación práctica. Finalmente, se comparan los resultados con las métricas de éxito definidas para decidir su implementación.

En nuestro caso, la evaluación se realizó de manera explícita sobre el conjunto de prueba (20% de los datos), que el modelo no había visto durante el entrenamiento. Se ejecutaron los siguientes pasos:

1. Se generó un **reporte de clasificación** completo para analizar las métricas de Precisión, Recall y F1-Score, validando el desempeño del modelo contra los criterios definidos en la fase de planificación.
2. Se visualizó una **matriz de confusión** para examinar en detalle la distribución de aciertos y errores, específicamente los Falsos Positivos y Falsos Negativos.
3. Se calculó el **área bajo la curva ROC (AUC)** como una medida global de la capacidad del modelo para discriminar entre las clases positiva y negativa.

Desafíos Presentados

A diferencia de un enfoque automatizado, los retos principales de este proyecto fueron de naturaleza técnica y conceptual:

- **Selección de Métricas de Evaluación**: El desafío más significativo fue definir las métricas de éxito adecuadas para el contexto clínico. Se determinó que optimizar para una sola métrica como el Accuracy o el AUC era insuficiente. El reto consistió en conceptualizar y justificar la necesidad de balancear el **Recall** (para minimizar falsos negativos) y la **Precisión** (para evitar un exceso de falsas alarmas), creando así un modelo clínicamente relevante.
- **Construcción de un Pipeline Reproducible**: Implementar el pipeline de forma manual exigió un control riguroso para asegurar que cada paso del preprocesamiento (como el escalado de datos y la selección de características) se ejecutará de manera idéntica en el entorno de entrenamiento y en el de producción (la API de FastAPI), para así garantizar la consistencia y fiabilidad de las predicciones.
- **Integración de la Arquitectura**: La integración del modelo con la aplicación web implicó desafíos técnicos en la comunicación entre el frontend (React/Next.js) y el backend (FastAPI), incluyendo la gestión de solicitudes y la optimización de los tiempos de respuesta para ofrecer una experiencia de usuario en tiempo real.

Estos desafíos evidenciaron que el desarrollo de una solución de machine learning efectiva y confiable requiere un profundo conocimiento del dominio del problema y una implementación técnica rigurosa.

Conclusiones

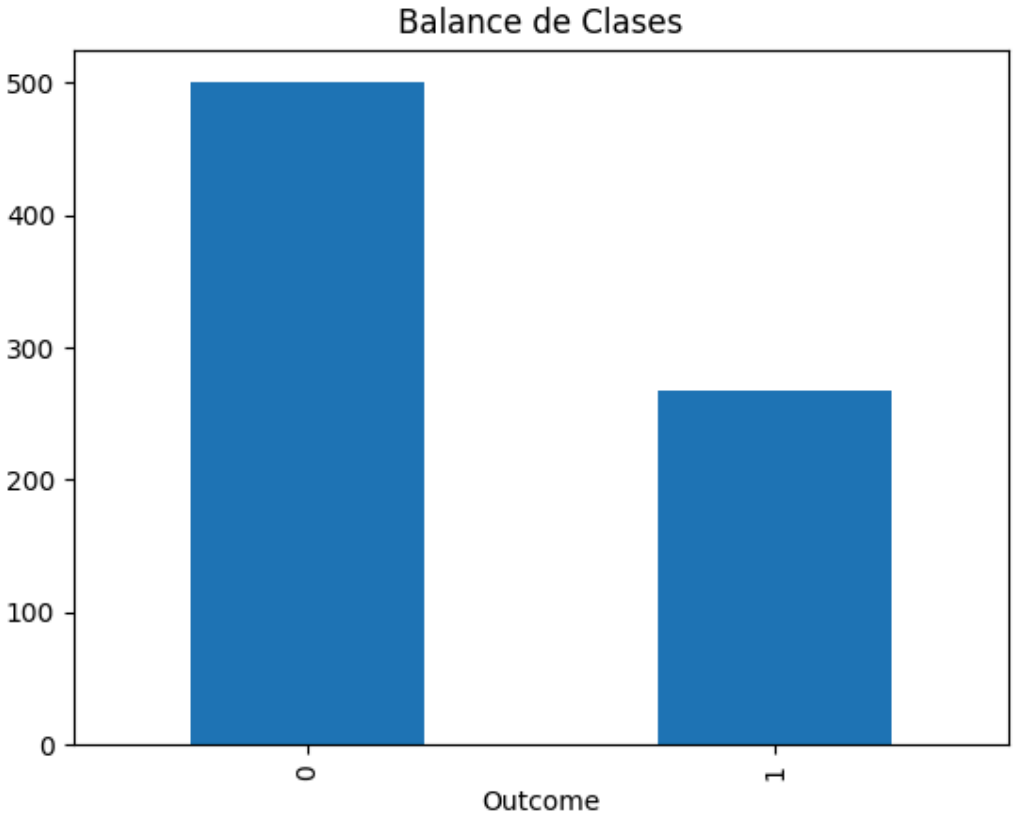
El proyecto desarrolló con éxito una solución completa de aprendizaje automático, desde el análisis de datos hasta una aplicación web funcional.

La conclusión principal es que un enfoque personalizado y transparente es superior a las herramientas automatizadas de tipo "caja negra" para aplicaciones en el sector de la salud. Esto se debe a que permite un control total sobre el proceso, la elección de métricas de evaluación que son clínicamente relevantes (como Recall y Precisión) y, fundamentalmente, la capacidad de interpretar *por qué* el modelo toma sus decisiones.

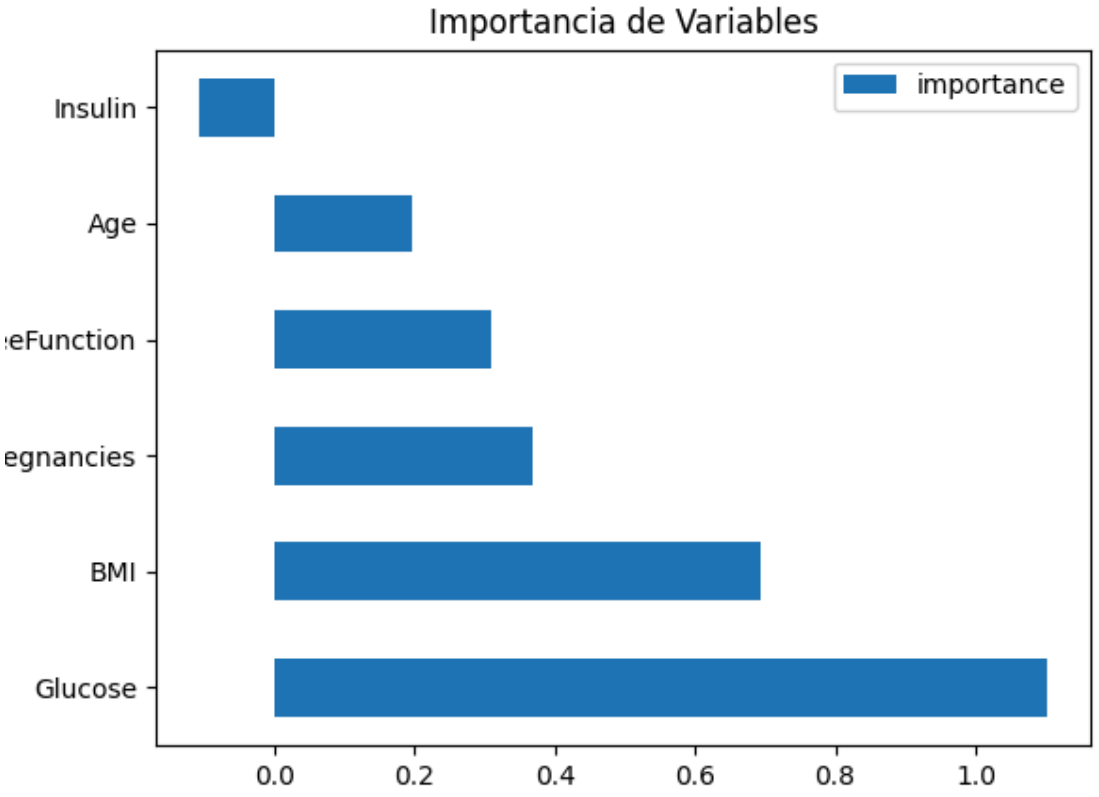
El trabajo demuestra el valor de combinar herramientas de código abierto con tecnologías web para crear soluciones de apoyo diagnóstico que son prácticas, escalables y confiables.

Anexo

Análisis Exploratorio de Datos



Dataset desbalanceado. Para evitar que el modelo se incline a favor de la clase mayoritaria, es crucial tomar medidas. La decisión de usar el parámetro class_weight='balanced' en el modelo de Regresión Logística es una consecuencia directa de este análisis. Esta técnica penaliza más los errores en la clase minoritaria, forzando al modelo a prestarle más atención.



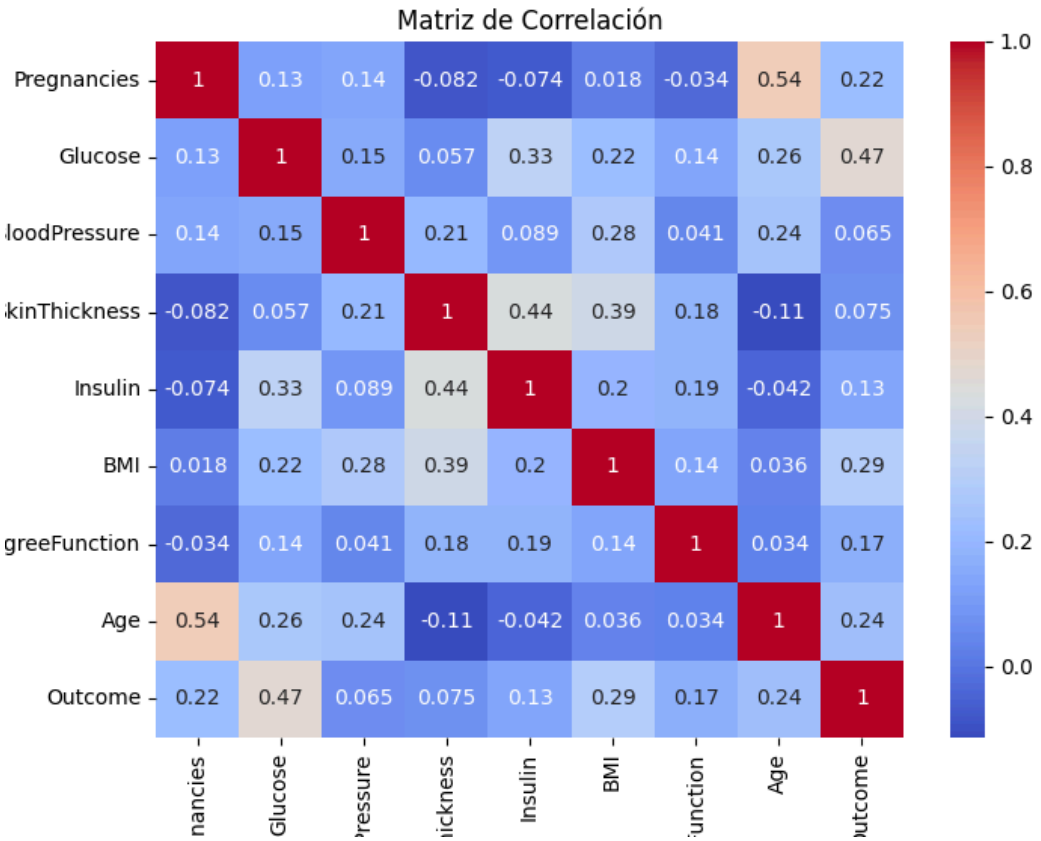
Interpretación del Gráfico

Este gráfico muestra el "peso" o la influencia que tiene cada variable en la predicción final del modelo de Regresión Logística entrenado.

- **Glucose** es, por un amplio margen, la variable más influyente.
- **BMI** es la segunda más importante.
- Age, Pregnancies y DiabetesPedigreeFunction tienen una importancia moderada.
- **Insulin** es la variable con menor impacto entre las que se utilizaron.

Decisiones que se Toman a Partir de Esto

- **Validar y Explicar el Modelo:** Este gráfico es la herramienta clave para la **explicabilidad**. Permite afirmar con confianza que el modelo no es una "caja negra". Se puede explicar a un médico que las predicciones se basan principalmente en factores lógicos y clínicamente relevantes como la glucosa y el BMI. Esto aumenta la confianza en la herramienta.

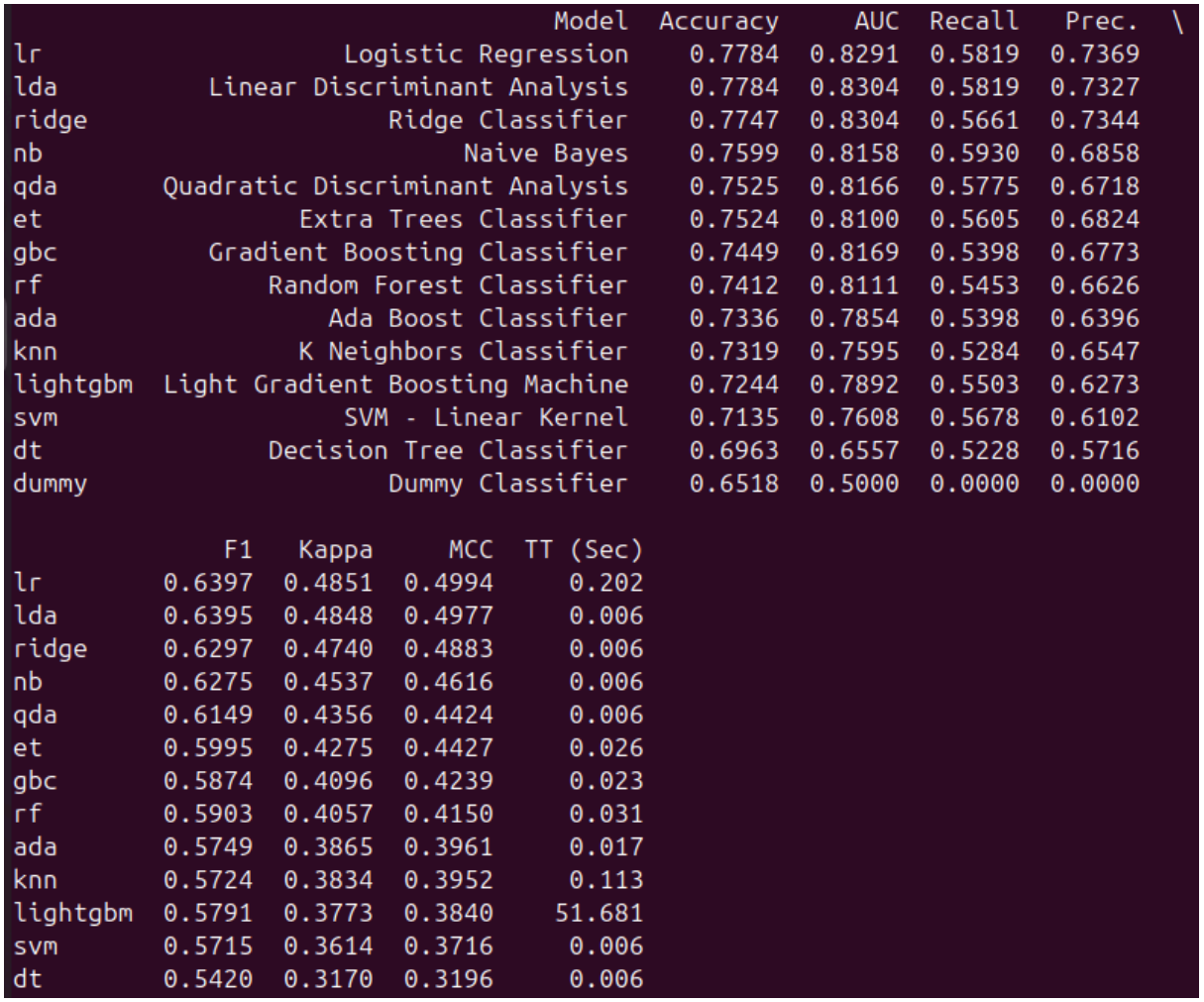


Glucose (0.47) y **BMI (0.29)** son las variables con la correlación positiva más fuerte con el resultado de diabetes. **BloodPressure (0.065)** y **SkinThickness (0.075)** tienen una correlación muy débil, cercana a cero.

Decisiones que se Toman a Partir de Esto

- **Selección de Características (Feature Selection):** Esta es la decisión más importante que se deriva de este gráfico. Al ver que BloodPressure y SkinThickness apenas se correlacionan con la probabilidad de tener diabetes, se justifica plenamente la decisión de **descartarlas del entrenamiento**. Esto simplifica el modelo, reduce el ruido y puede mejorar su rendimiento al centrarse en las variables que realmente importan.
- **Validación de Hipótesis:** Confirma el conocimiento médico general de que el nivel de glucosa y el índice de masa corporal son predictores clave de la diabetes.

Elección del modelo



	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	0.7784	0.8291	0.5819	0.7369	0.6397	0.4851	0.4994	0.202
lda	Linear Discriminant Analysis	0.7784	0.8304	0.5819	0.7327	0.6395	0.4848	0.4977	0.006
ridge	Ridge Classifier	0.7747	0.8304	0.5661	0.7344	0.6297	0.4740	0.4883	0.006
nb	Naive Bayes	0.7599	0.8158	0.5930	0.6858	0.6275	0.4537	0.4616	0.006
qda	Quadratic Discriminant Analysis	0.7525	0.8166	0.5775	0.6718	0.6149	0.4356	0.4424	0.006
et	Extra Trees Classifier	0.7524	0.8100	0.5605	0.6824	0.5995	0.4275	0.4427	0.026
gbc	Gradient Boosting Classifier	0.7449	0.8169	0.5398	0.6773	0.5874	0.4096	0.4239	0.023
rf	Random Forest Classifier	0.7412	0.8111	0.5453	0.6626	0.5903	0.4057	0.4150	0.031
ada	Ada Boost Classifier	0.7336	0.7854	0.5398	0.6396	0.5749	0.3865	0.3961	0.017
knn	K Neighbors Classifier	0.7319	0.7595	0.5284	0.6547	0.5724	0.3834	0.3952	0.113
lightgbm	Light Gradient Boosting Machine	0.7244	0.7892	0.5503	0.6273	0.5791	0.3773	0.3840	51.681
svm	SVM - Linear Kernel	0.7135	0.7608	0.5678	0.6102	0.5715	0.3614	0.3716	0.006
dt	Decision Tree Classifier	0.6963	0.6557	0.5228	0.5716	0.5420	0.3170	0.3196	0.006
dummy	Dummy Classifier	0.6518	0.5000	0.0000	0.0000				

Captura de la salida del script compare_models.py, donde PyCaret analiza distintos modelos por nosotros.

Código fuente

https://github.com/claram97/predictor-diabetes

Referencias

Explicación del ciclo de vida del machine learning | DataCamp
Machine learning: definición, métodos y ejemplos
¿Qué es el machine learning? | Oracle Argentina
Predict Diabetes From Medical Records