Universidad Técnica Particular de Loja  
Maestría en Inteligencia Artificial Aplicada  
Módulo. Herramientas para Inteligencia Artificial  
Estudiante: Ing.: José Gabriel Aguirre Andrade Msc.  
Docente: PhD. Gonzalez Eras Alexandra Cristina

Fecha: 17 de Mayo de 2025

## **Práctica 2: Uso de una herramienta para desplegar un modelo de inteligencia artificial**

**🔍 Introducción**

La inteligencia artificial (IA) ha comenzado a desempeñar un papel clave en la reducción de riesgos laborales. En este proyecto se desarrolla una aplicación predictiva basada en datos abiertos de la OSHA, con el objetivo de prevenir accidentes industriales mediante el análisis de texto y la proyección de incidentes futuros. Utilizando herramientas de software libre y colaborativas como pandas, scikit-learn, Flask y MLflow, se construyó una solución completa, trazable y reproducible.

## ✅ Informe paso a paso según el requerimiento

### 1. Obtener un dataset de libre acceso

Para este proyecto se utilizó un conjunto de datos público y gratuito disponible en Kaggle, específicamente la base de datos de resumen anual de accidentes laborales de la OSHA (Occupational Safety and Health Administration) de EE.UU., versión FY16. Esta fuente es legítima, oficial y cumple con el criterio de libre acceso.

📂 Archivo utilizado: fy16\_federal-state\_summaries.csv  
🛠️ Transformado y renombrado como: osha\_riesgo.csv  
📌 Ubicación: carpeta /data

En este archivo se conservó la columna descripcion\_riesgo (texto del accidente) y se creó una nueva variable binaria riesgo\_alto basada en la gravedad del incidente (hospitalización o muerte).

### 2. En el notebook, usar MLflow como herramienta para el manejo del ciclo de vida

MLflow fue utilizado para rastrear el ciclo de vida del modelo de machine learning de forma completa. Se implementó en el script train\_model.py mediante las siguientes instrucciones:

mlflow.set\_tracking\_uri("http://localhost:9090")

mlflow.set\_experiment("Prevencion\_Accidentes\_Maquinaria")

El modelo se registró, versionó y se guardaron sus métricas con:

mlflow.log\_metric("accuracy", acc)

mlflow.sklearn.log\_model(pipeline, "modelo\_riesgo\_logistic")

📂 Archivo donde se implementó: train\_model.py

### 3. Aplicar algún algoritmo de aprendizaje automático o aprendizaje profundo en función del dataset seleccionado

Se implementaron **dos modelos de aprendizaje automático**:

#### a) Clasificación con Logistic Regression

Usado para predecir si un incidente representa “riesgo alto” o “riesgo bajo”, a partir del campo de texto descripcion\_riesgo.  
📂 Implementado en: train\_model.py  
📚 Librerías: scikit-learn, TfidfVectorizer, Pipeline, LogisticRegression

#### b) Proyección con Linear Regression

Se entrenó un modelo de regresión lineal que simula la disminución de accidentes en un período de 5 años (2025–2030) con la implementación de mejoras de seguridad industrial en maquinaria.

📂 Implementado en: train\_model.py  
📈 Resultado gráfico: static/curva\_proyeccion.png

### 4. Gestionar el código de tal manera que se graben los experimentos en MLflow

Todos los experimentos fueron gestionados usando MLflow:

* Se inició y cerró correctamente la sesión de ejecución:

with mlflow.start\_run():

...

mlflow.log\_metric(...)

mlflow.sklearn.log\_model(...)

* Se registró el experimento con nombre y URI definidos.

📂 Archivo principal: train\_model.py  
📌 Carpeta de tracking: MLflow en http://localhost:9090

### 5. El servidor de MLflow deberá ejecutarse en el puerto 9090

MLflow fue configurado para ejecutarse localmente en el **puerto 9090**, cumpliendo con la exigencia docente.

📜 Comando usado en terminal:

mlflow ui --port 9090

📍 Dirección para verificar:

http://localhost:9090

🟢 Confirmación: al ejecutar el script train\_model.py, se registró la ejecución y se generó un enlace directo al experimento y al run (View run salty-penguin-957).

### 6. Generar un modelo de machine learning

El modelo de clasificación fue guardado localmente para consumo por Flask:

mlflow.sklearn.save\_model(pipeline, "modelo/logistic\_model")

📂 Carpeta de salida: /modelo/logistic\_model/  
📁 Contenido generado: MLmodel, conda.yaml, model.pkl

Adicionalmente, la proyección generada con regresión lineal fue guardada como imagen para mostrar en la interfaz web.

### 7. Consumir el modelo a través de un servicio web, a través de Flask

Se creó una interfaz web con Flask para mostrar:

* La curva proyectada de accidentes (curva\_proyeccion.png)
* Un **top 10** de tipos de accidentes con mayor frecuencia (calculado desde descripcion\_riesgo)
* Logo institucional
* Pie de página con nombre del estudiante, docente y universidad

📂 Estructura:

/templates/

├── index.html ← Página de entrada

└── result.html ← Muestra resultados y visualización

/static/

├── style.css ← Estilo visual

└── curva\_proyeccion.png ← Imagen generada automáticamente

📂 Código principal de servidor: app.py  
🛠️ Librerías usadas: flask, pickle, numpy, pandas

## 🧾 Paso a paso de lo ha realizado.

### 🔹 1. Selección del dataset

* Se eligió el dataset de libre acceso del portal **Kaggle**, relacionado con **accidentes laborales en EE. UU. (OSHA)**.
* El archivo base fue fy16\_federal-state\_summaries.csv.
* Luego, se generó un archivo limpio llamado osha\_riesgo.csv, que incluyó columnas como:
  + descripcion\_riesgo (texto del evento)
  + riesgo\_alto (1 o 0 como etiqueta de clasificación)
  + tipo\_evento (como atrapamiento, corte, golpe, etc.)

### 🔹 2. Instalación de librerías requeridas

Las siguientes librerías se instalaron mediante pip o desde Google Colab:

pip install pandas numpy matplotlib scikit-learn flask mlflow

Librerías utilizadas:

* pandas: manejo de datos tabulares (CSV).
* numpy: manejo de arrays numéricos.
* matplotlib: generación de gráficos de líneas y barras.
* scikit-learn: creación del modelo de machine learning (LogisticRegression, LinearRegression).
* mlflow: seguimiento del ciclo de vida del modelo.
* flask: consumo del modelo como servicio web.

### 🔹 3. Configuración de MLflow

* Se ejecutó el servidor con:

mlflow ui --port 9090

* En el script train\_model.py se agregó:

mlflow.set\_tracking\_uri("http://localhost:9090")

mlflow.set\_experiment("Prevencion\_Accidentes\_Maquinaria")

### 🔹 4. Entrenamiento del modelo de predicción

* Se creó un modelo de regresión lineal para estimar la **disminución de accidentes** en un período de 5 años, basado en mejoras en seguridad de máquinas.
* El script train\_model.py:
  + Entrenó el modelo.
  + Generó una gráfica llamada curva\_proyeccion.png.
  + Guardó el modelo como model.pkl usando mlflow.sklearn.save\_model(...).

### 🔹 5. Consumo del modelo con Flask

* Se desarrolló un servidor web en app.py que:
  + Carga el modelo model.pkl.
  + Lee el archivo osha\_riesgo.csv.
  + Muestra la gráfica de proyección y el **Top 10 de tipos de accidentes** más frecuentes.
* Se usaron los archivos HTML:
  + index.html → redirecciona a /result.
  + result.html → muestra la curva + tabla de eventos.

### 🔹 6. Archivos HTML y estilo

* Se agregó el logo institucional (logo\_utpl.png) en la carpeta static/.
* Se creó style.css con diseño amigable.
* result.html contiene:
  + La gráfica proyectiva.
  + Una tabla de eventos.
  + Pie de página con tu nombre, el de la docente y la UTPL.

### 🔹 7. Organización del proyecto

Estructura final del proyecto:

/proyecto\_accidentes

│

├── data/

│ └── osha\_riesgo.csv

│

├── modelo/

│ └── logistic\_model/

│ └── model.pkl

│

├── static/

│ ├── curva\_proyeccion.png

│ ├── logo\_utpl.png

│ └── style.css

│

├── templates/

│ ├── index.html

│ └── result.html

│

├── train\_model.py

├── app.py

### ✅ Resultado final

* Se entrenó y almacenó un modelo predictivo usando datos reales.
* Se generó una **curva de proyección de accidentes hasta 2030**.
* Se presentó el resultado mediante una **página web profesional** con visualizaciones claras.
* Todo el ciclo fue **gestionado con MLflow en el puerto 9090**.

**📌 Conclusiones**

* El uso de herramientas libres permite diseñar soluciones efectivas para la seguridad industrial.
* MLflow garantiza la trazabilidad completa de los modelos.
* El enfoque proyectivo ayuda a visualizar el impacto futuro de las medidas de seguridad de máquina.

**📎 Recomendaciones**

* Ampliar el dataset incluyendo más años y detalles técnicos del accidente.
* Incorporar algoritmos más complejos (e.g., redes neuronales para clasificación).
* Desplegar la app en la nube (Render, Heroku, AWS) para pruebas externas.

**📚 Referencias (formato APA 7.ª edición)**

* McKinney, W. (2010). *Data structures for statistical computing in Python*. Proceedings of the 9th Python in Science Conference, 51–56.
* Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). *Scikit-learn: Machine learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.
* OSHA. (2016). *Federal/State Summary Data*. Occupational Safety and Health Administration. <https://www.osha.gov>
* MLflow. (2024). *MLflow Documentation*. https://mlflow.org/docs/latest/
* Flask. (2024). *Flask Documentation*. https://flask.palletsprojects.com/
* Python Software Foundation. (2024). *pandas: Powerful Python data analysis toolkit*. <https://pandas.pydata.org>
* Matplotlib. (2024). *Matplotlib documentation*. <https://matplotlib.org>







