



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

Trabajo académico final escrito: Genera una aplicación de inteligencia artificial que use librerías de software libre a través de herramientas colaborativas.

Fecha de entrega: 4 de mayo de 2025

Repositorio GitHub: Trabajo Final José Aguirre

Archivos:

- **El dataset inicial que será usado:** eventos_paletizador.csv
- **El SQL de la base de datos:** paletizador_db.sql
- **El script/notebook que genera la base de datos:.** insertar_paletizadores.sql
- **El notebook donde se realizan los pasos descritos en la actividad:** analisis_paletizador.ipynb
- **El dataset final usado:** dataset_final.csv

1. Introducción

La automatización industrial contemporánea demanda metodologías analíticas robustas que permitan extraer valor estratégico de los volúmenes crecientes de datos operativos. En este contexto, el presente trabajo explora la aplicación de técnicas avanzadas de análisis de datos para optimizar el funcionamiento de paletizadores en entornos embotelladores, máquinas críticas que determinan la eficiencia logística de las líneas de producción mediante la organización automatizada de productos en unidades de carga.

Este estudio implementa un enfoque integrador que combina fuentes estructuradas y no estructuradas, específicamente datos almacenados en formato CSV y en sistemas PostgreSQL. Mediante la correlación de variables técnicas y operativas, se busca identificar factores determinantes en el rendimiento de los equipos y desarrollar métricas predictivas de comportamiento. El análisis examina parámetros clave como duración y frecuencia de eventos, diferencias por modelo y variaciones en la eficiencia según el personal técnico asignado.

Para la ejecución técnica se estableció un entorno híbrido que aprovecha las capacidades analíticas de Google Colab junto con herramientas especializadas como PyCharm y PgAdmin para la gestión de bases de datos relacionales. Esta configuración facilitó un flujo metodológico coherente desde la adquisición inicial de datos hasta la generación de visualizaciones interpretables, maximizando así el potencial informativo inherente a los registros operativos.

2. Objetivo general

Desarrollar una aplicación de inteligencia artificial utilizando librerías de software libre y herramientas colaborativas para el análisis de datos de eventos de paletizadores industriales.

3. Objetivos específicos

- Implementar un entorno local con PyCharm y PgAdmin para análisis de datos.
- Configurar un entorno colaborativo con Google Colab y base de datos en Render.
- Consumir datos desde un archivo CSV y desde una base de datos PostgreSQL usando Pandas.
- Realizar visualizaciones avanzadas con Matplotlib incluyendo identidad gráfica.
- Comparar los resultados, ventajas y limitaciones entre ambos entornos.

4. Marco teórico

4.1 Librerías utilizadas

4.1.1 Pandas

Pandas es una biblioteca de Python especializada en el análisis y manipulación de datos. Proporciona estructuras de datos flexibles como DataFrames que permiten operaciones eficientes sobre conjuntos de datos tabulares. En este proyecto, Pandas se utiliza para:

- Leer datos desde diferentes fuentes (CSV y PostgreSQL)
- Fusionar conjuntos de datos mediante operaciones de join
- Filtrar y transformar los datos para su análisis
- Realizar agrupaciones y cálculos estadísticos

4.1.2 SQLAlchemy

SQLAlchemy es un ORM (Object-Relational Mapping) para Python que facilita la interacción con bases de datos relacionales. Proporciona una interfaz

programática para realizar operaciones de base de datos sin necesidad de escribir SQL directamente. En este proyecto se utiliza para:

- Establecer la conexión con la base de datos PostgreSQL alojada en Render
- Ejecutar consultas SQL y recuperar datos en formato Pandas DataFrame

4.1.3 Matplotlib

Matplotlib es una biblioteca de visualización en Python, que permite crear gráficos estáticos, interactivos y animados. Es altamente personalizable y ofrece múltiples tipos de visualizaciones. En este proyecto se utiliza para:

- Generar gráficos de barras para visualizar métricas relacionadas con los paletizadores
- Personalizar la apariencia de los gráficos con colores, etiquetas y títulos
- Incorporar elementos adicionales como logos y textos informativos
- Ajustar aspectos de formato como el tamaño y la orientación de las etiquetas

4.1.4 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto, conocido por su robustez, escalabilidad y cumplimiento con los estándares SQL. Para este proyecto:

- Se utiliza como repositorio para almacenar la información técnica de los paletizadores
- Se despliega en la plataforma Render para acceso remoto
- Se administra mediante PgAdmin para facilitar la gestión visual de la base de datos

4.1.5 Google Colab

Google Colab es un entorno de notebook basado en la nube que permite escribir y ejecutar código Python sin necesidad de configuración local. Proporciona acceso gratuito a recursos computacionales incluyendo GPUs. En este proyecto:

- Sirve como plataforma principal para el desarrollo del código
- Facilita la documentación del proceso mediante la combinación de código y texto explicativo
- Permite la ejecución en la nube sin dependencias de hardware local

4.2 Descripción del dataset usado

El análisis se basa en la integración de dos conjuntos de datos complementarios:

4.2.1 Dataset de eventos de paletizador (CSV)

El archivo `eventos_paletizador.csv` contiene registros de eventos operativos de los paletizadores en la planta embotelladora. Cada registro incluye:

- **machine_id**: Identificador único de la máquina paletizadora
- **evento**: Tipo de evento registrado (mantenimiento, avería, parada programada, etc.)
- **tiempo_segundos**: Duración del evento en segundos
- **operario**: Identificador o nombre del operario que registró el evento
- **fecha**: Fecha y hora en que ocurrió el evento

Este dataset proporciona información temporal sobre el funcionamiento de las máquinas, lo que permite analizar patrones de eventos y eficiencia operativa.

4.2.2 Dataset de información técnica (PostgreSQL)

Almacenado en una base de datos PostgreSQL alojada en Render, este conjunto contiene las especificaciones técnicas de cada paletizador:

- **machine_id**: Identificador único que sirve como clave para relacionar con el dataset de eventos
- **model**: Modelo específico del paletizador
- **manufacturer**: Fabricante de la máquina
- **year**: Año de fabricación
- **capacity**: Capacidad operativa (unidades/hora)
- **location**: Ubicación dentro de la planta

Esta información técnica permite contextualizar los eventos operativos y analizar el rendimiento según las características de cada máquina.

La integración de ambos datasets se realiza mediante una operación de join utilizando el campo común `machine_id`, lo que permite un análisis integral que combina eventos operativos con características técnicas.

4.3 Descripción de los pasos realizados en el proyecto

El proyecto se ha desarrollado siguiendo una metodología estructurada de análisis de datos:

4.3.1 Configuración del entorno y conexión a fuentes de datos

1. **Instalación y carga de bibliotecas:** Se importaron las bibliotecas necesarias (Pandas, SQLAlchemy, Matplotlib) para la manipulación de datos y visualización.
2. **Conexión a la base de datos PostgreSQL:** Se estableció una conexión a la base de datos alojada en Render utilizando SQLAlchemy, especificando los parámetros de conexión (usuario, contraseña, host, puerto y nombre de la base de datos).
3. **Carga de datos:** Se extrajeron los datos técnicos de los paletizadores mediante una consulta SQL, y se cargó el archivo CSV con los eventos registrados.

4.3.2 Preparación y limpieza de datos

1. **Homogeneización de tipos de datos:** Se aseguró que el campo `machine_id` tuviera el mismo tipo de datos (string) en ambos datasets para facilitar la unión.
2. **Fusión de datasets:** Se combinaron ambos conjuntos de datos mediante una operación de merge por el campo `machine_id`, utilizando un left join para mantener todos los eventos.
3. **Eliminación de datos incompletos:** Se filtraron las filas con valores nulos en el campo 'model', asegurando que todos los registros analizados tuvieran la información técnica completa.

4.3.3 Análisis y generación de visualizaciones

1. **Definición de paleta de colores:** Se estableció una paleta de colores utilizando el esquema `tab10` de Matplotlib para mantener una estética consistente en todas las visualizaciones.
2. **Creación de funciones auxiliares:** Se implementaron funciones para estandarizar la creación de gráficos y la incorporación de elementos visuales como logos y textos institucionales.
3. **Cálculo de métricas:** Se realizaron agrupaciones y cálculos estadísticos para obtener información relevante sobre el rendimiento de las máquinas y los operarios.
4. **Generación de visualizaciones:** Se crearon gráficos específicos para representar diferentes aspectos del análisis.

4.4 Breve descripción de las visualizaciones generadas

4.4.1 Tiempo Promedio de Eventos por Modelo de Máquina

Este gráfico de barras muestra el tiempo promedio (en segundos) que toma cada tipo de evento según el modelo de paletizador. Permite identificar qué modelos tienen eventos más prolongados, lo que podría indicar problemas de eficiencia o complejidad en ciertas máquinas.

4.4.2 Cantidad de Eventos por Modelo de Máquina

Visualiza la frecuencia de eventos registrados para cada modelo de paletizador. Esta información es valiosa para determinar qué modelos requieren más atención o intervenciones, posiblemente indicando menor fiabilidad o mayor uso.

4.4.3 Eventos Registrados por Operario

Muestra la distribución de eventos registrados por cada operario, lo que permite analizar la carga de trabajo y posiblemente identificar diferencias en los patrones de registro entre el personal.

4.4.4 Duración Promedio de Eventos por Operario

Representa el tiempo promedio que toman los eventos gestionados por cada operario. Este gráfico puede revelar diferencias en la eficiencia o en la complejidad de las tareas asignadas a diferentes operarios.

4.4.5 Eventos por Tipo de Evento

Visualiza la frecuencia de cada tipo de evento (mantenimiento, avería, parada programada, etc.), proporcionando una visión general de las principales causas de interrupción o intervención en los paletizadores.

4.4.6 Duración Promedio por Tipo de Evento

Muestra el tiempo promedio que requiere cada tipo de evento, permitiendo identificar qué tipos de intervenciones consumen más tiempo y potencialmente representan mayores costos operativos.

Todas las visualizaciones mantienen un diseño consistente que incluye el logo institucional de la UTPL y la información relevante del contexto académico, facilitando su presentación formal.

4.5 Comparativa de entornos

Aspecto	PyCharm + PgAdmin (Local)	Colab + Render (Nube)
Instalación	Requiere instalación local	Sin instalación, uso web
Acceso a BD	Local	Remoto, requiere configuración
Rendimiento	Alto si PC es potente	Depende de conexión internet
Visualización	Ventanas locales	En el navegador

Reproducibilidad Menor (dependencia de PC) Mayor (acceso global)

5. Descripción de los datasets

5.1 Dataset CSV: eventos_paletizador.csv

Contiene registros de eventos de un sistema de paletizado, con columnas: machine_id, evento, fecha, operario, tiempo_segundos. Se generaron 1000 registros sintéticos por 5 máquinas diferentes.

5.2 Dataset SQL: palletizer_info

Tabla de PostgreSQL con los campos: machine_id, model, location, install_date, last_maintenance, certified_operator.

6. Descripción de los pasos realizados

6.1 Instalación de librerías (en Google Colab)

```
!pip install sqlalchemy psycopg2-binary pandas matplotlib
```

6.1.1 Instalación de librerías (en PyCharm/local)

```
pip install sqlalchemy psycopg2-binary pandas matplotlib
```

6.2 Conexión a la base de datos

En Google Colab (Render):

```
from sqlalchemy import create_engine
conexion = create_engine('postgresql+psycopg2://paletizador_db_user:dEpNZeWHbUWiOtiTEHgcQVSgVv0oMXX4@dpg-d0budtjuibrs73dm52d0-a.oregon-postgres.render.com/paletizador_db')
```

En PyCharm (pgAdmin):

```
conexion = create_engine('postgresql+psycopg2://postgres:1234@localhost:5432/paletizador_db')
```

6.3 Lectura de los datasets

```
import pandas as pd
```

```
# Dataset SQL
df_sql = pd.read_sql('SELECT * FROM palletizer_info', conexion)

# Dataset CSV
df_csv = pd.read_csv('eventos_paletizador.csv')
```

6.4 Unión de datasets

```
df_csv['machine_id'] = df_csv['machine_id'].astype(str)
df_sql['machine_id'] = df_sql['machine_id'].astype(str)
df_combinado = pd.merge(df_csv, df_sql, on='machine_id', how='left')
df_combinado_filtrado = df_combinado.dropna(subset=['model'])
```

6.5 Visualizaciones generadas

Se generaron los siguientes 6 gráficos con Matplotlib:

1. Tiempo promedio de eventos por modelo de máquina
2. Cantidad de eventos por modelo
3. Eventos por operario
4. Duración promedio por operario
5. Eventos por tipo de evento
6. Duración promedio por tipo de evento

Cada gráfico incluye:

- Paleta de colores `tab10`
- Logo institucional
- Textos institucionales con el nombre del estudiante y docente evaluadora

Código de generación de gráficos (Colab)

```
# Importamos las librerías necesarias
import pandas as pd                                # Para manejo de datos
                                                    # Para manejo de datos
                                                    # Para manejo de datos
import sqlalchemy                                    # Para conexión con
PostgreSQL                                          # Para conexión con
                                                    # Para conexión con
import matplotlib.pyplot as plt                    # Para crear gráficos
import matplotlib.image as mpimg                   # Para insertar imagen
en el gráfico (logo)
from mpl_toolkits.axes_grid1.inset_locator import inset_axes # Para
posicionar el logo dentro del gráfico
import os                                           # Para verificar existencia
del logo

# Datos de conexión a la base en Render
usuario = 'paletizador_db_user'
```



```

contraseña = 'dEpNZeWHbUWiOtiTEHgcQVSgVv0oMXX4'
host = 'dpg-d0budtjuibrs73dm52d0-a.oregon-postgres.render.com'
puerto = '5432'
basededatos = 'paletizador_db'

# Crear el motor de conexión con SQLAlchemy
conexion =
sqlalchemy.create_engine(f'postgresql+psycopg2://{usuario}:{contrase
ña}@{host}:{puerto}/{basededatos}')

# Leer los datos de la base PostgreSQL y del archivo CSV
df_sql = pd.read_sql('SELECT * FROM palletizer_info',
conexion) # Datos técnicos de cada máquina
df_csv =
pd.read_csv('eventos_paletizador.csv') #
Eventos registrados por los operarios

# Asegurar que machine_id sea string en ambos
df_csv['machine_id'] = df_csv['machine_id'].astype(str)
df_sql['machine_id'] = df_sql['machine_id'].astype(str)

# Unir los DataFrames por machine_id
df_combinado = pd.merge(df_csv, df_sql, on='machine_id', how='left')

# Eliminar filas con valores nulos en 'model'
df_combinado_filtrado = df_combinado.dropna(subset=['model'])

# Definir colores variados para los gráficos
colores = plt.cm.tab10.colors

# Ruta al logo
logo_path = 'logo_utpl.png'

# Función para agregar logo y textos institucionales
def agregar_logo_texto(fig, ax, titulo):
    ax.set_title(titulo, fontsize=10) # Título del gráfico

    # Insertar logo en la parte superior izquierda
    if os.path.exists(logo_path):
        logo = mpimg.imread(logo_path)
        axins = inset_axes(ax, width="10%", height="10%", loc='upper
left') # Tamaño y posición
        axins.imshow(logo)
        axins.axis('off') # Quitar borde al logo

```

```

    # Insertar textos en la parte inferior derecha
    fig.text(0.99, 0.02, "Universidad Técnica Particular de Loja",
fontsize=6, ha='right')
    fig.text(0.99, 0.045, "Maestría en Inteligencia Artificial
Aplicada", fontsize=6, ha='right')
    fig.text(0.99, 0.07, "Módulo: Herramientas para Inteligencia
Artificial", fontsize=6, ha='right')
    fig.text(0.99, 0.095, "Estudiante: Ing. José Gabriel Aguirre
Andrade, MSc.", fontsize=6, ha='right')
    fig.text(0.99, 0.12, "Docente: PhD. Alexandra Cristina Gonzalez
Eras", fontsize=6, ha='right')

# Función para generar gráficos con estilo uniforme
def generar_grafico(datos, titulo, ylabel, xlabel):
    fig, ax = plt.subplots(figsize=(10, 5)) # Tamaño del gráfico
    datos.plot(kind='bar', ax=ax,
color=list(colores)[:len(datos)]) # Tipo de gráfico
    ax.set_ylabel(ylabel)
    ax.set_xlabel(xlabel, fontsize=8)
    ax.tick_params(axis='x', labelrotation=90, labelsiz=6) #
Fuente y orientación de etiquetas
    agregar_logo_texto(fig, ax, titulo)
    plt.subplots_adjust(bottom=0.25) # Espacio inferior
    plt.tight_layout()
    plt.show()

# GRÁFICO 1: Tiempo promedio por modelo
generar_grafico(
    df_combinado_filtrado.groupby('modelo')['tiempo_segundos'].mean()
,
    'Tiempo Promedio de Eventos por Modelo de Máquina',
    'Tiempo (segundos)', 'Modelo de Máquina'
)

# GRÁFICO 2: Cantidad de eventos por modelo
generar_grafico(
    df_combinado_filtrado['modelo'].value_counts(),
    'Cantidad de Eventos por Modelo de Máquina',
    'Número de Eventos', 'Modelo de Máquina'
)

# GRÁFICO 3: Eventos por operario
generar_grafico(
    df_combinado_filtrado['operario'].value_counts(),
    'Eventos Registrados por Operario',

```

```

        'Cantidad de Eventos', 'Operario'
    )

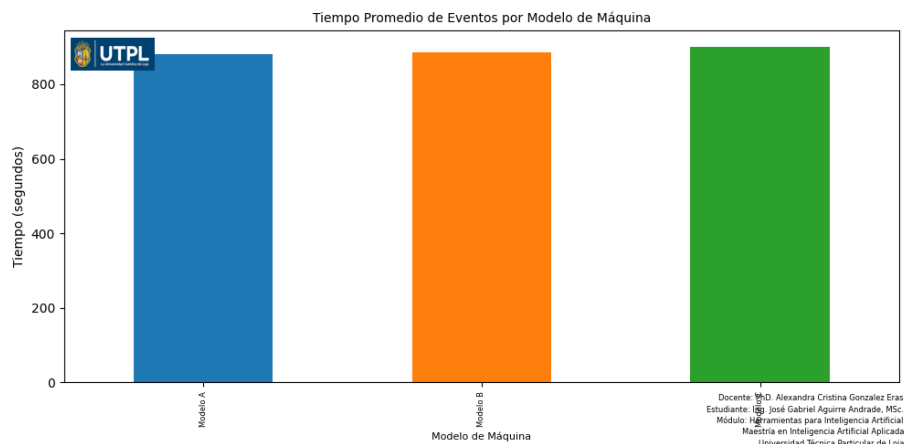
# GRÁFICO 4: Duración promedio por operario
generar_grafico(
    df_combinado_filtrado.groupby('operario')['tiempo_segundos'].mean(),
    'Duración Promedio de Eventos por Operario',
    'Tiempo Promedio (segundos)', 'Operario'
)

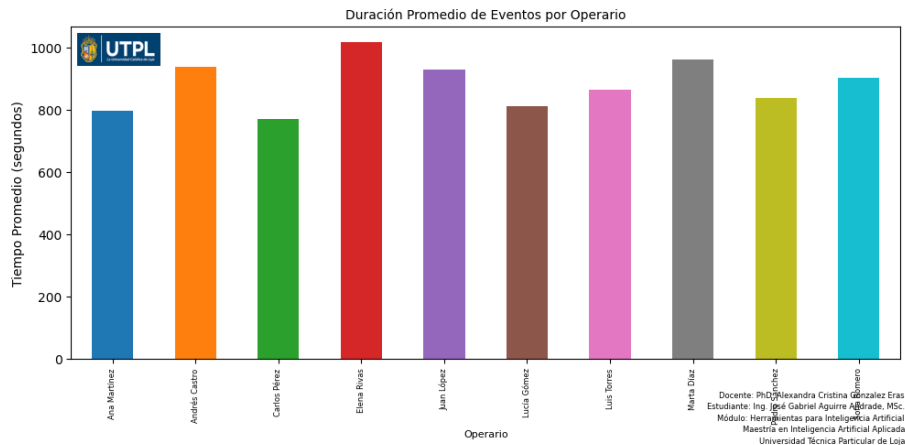
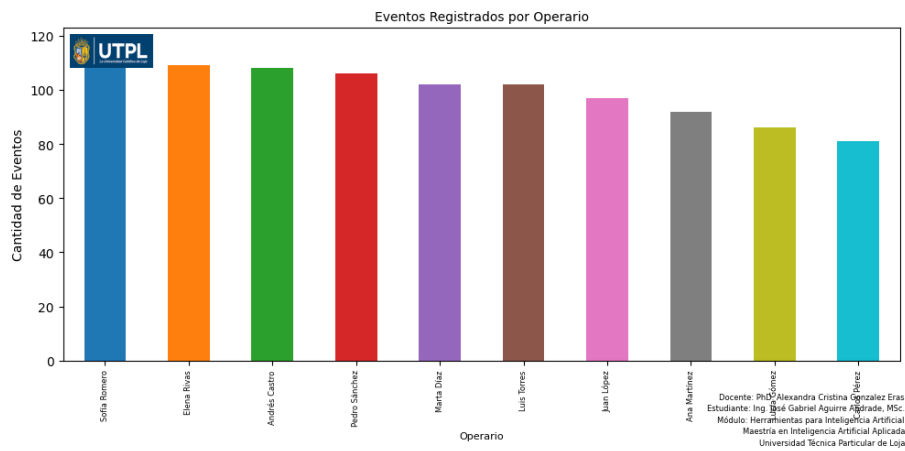
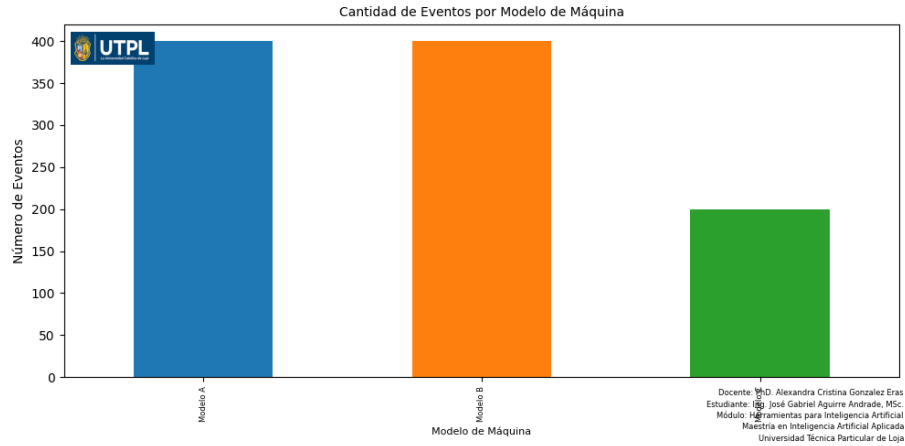
# GRÁFICO 5: Eventos por tipo de evento
generar_grafico(
    df_combinado_filtrado['evento'].value_counts(),
    'Eventos por Tipo de Evento',
    'Cantidad', 'Tipo de Evento'
)

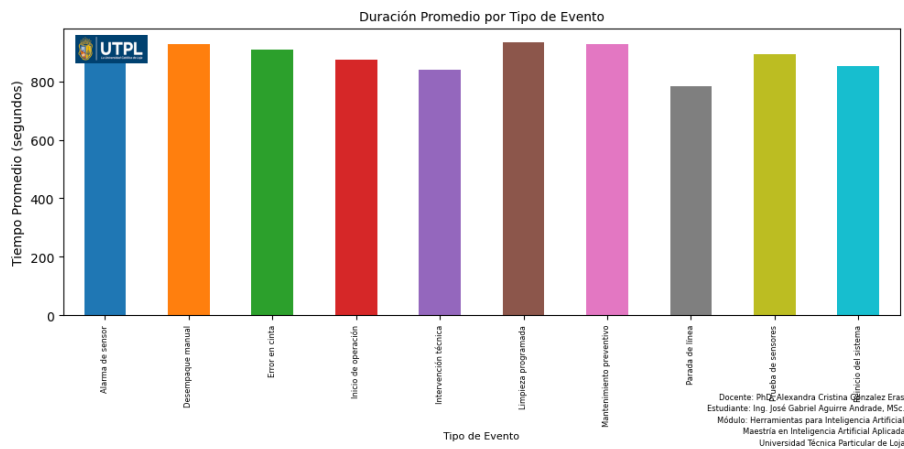
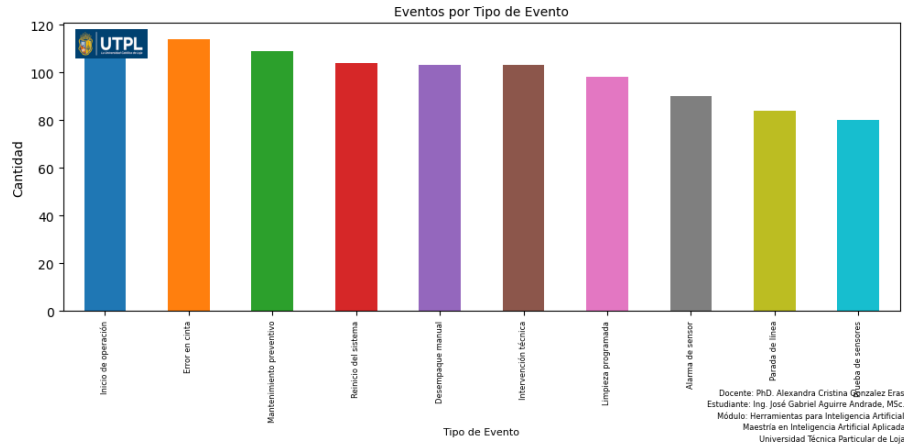
# GRÁFICO 6: Duración promedio por tipo de evento
generar_grafico(
    df_combinado_filtrado.groupby('evento')['tiempo_segundos'].mean(),
    'Duración Promedio por Tipo de Evento',
    'Tiempo Promedio (segundos)', 'Tipo de Evento'
)

```

Resultados:







Código de generación de gráficos (PyCharm)

```
# Importamos las librerías necesarias
import pandas as pd # Para manipular y analizar datos tabulares
import sqlalchemy # Para conectarse a la base de datos PostgreSQL
import matplotlib.pyplot as plt # Para graficar
import matplotlib.image as mpimg # Para insertar imágenes (logo) en los gráficos
from mpl_toolkits.axes_grid1.inset_locator import inset_axes # Para insertar el logo dentro del área del gráfico
import os # Para trabajar con archivos y rutas

# Configuración de conexión a PostgreSQL
usuario = 'postgres' # Usuario de la base de datos
contraseña = '1234' # Contraseña del usuario
host = 'localhost' # Dirección del servidor (localhost si es local)
puerto = '5432' # Puerto de PostgreSQL
basededatos = 'paletizado_db' # Nombre de la base de datos

# Crear el motor de conexión SQLAlchemy
conexion =
```

```

sqlalchemy.create_engine(f'postgresql+psycopg2://{usuario}:{contraseña}@
{host}:{puerto}/{basededatos}')

# Leer tabla desde PostgreSQL y archivo CSV
df_sql = pd.read_sql('SELECT * FROM palletizer_info', conexion) #
Cargar tabla de PostgreSQL
df_csv = pd.read_csv('eventos_paletizador.csv') # Cargar archivo CSV
con eventos

# Imprimir columnas disponibles para verificación
print("Columnas de df_csv:", df_csv.columns.tolist())
print("Columnas de df_sql:", df_sql.columns.tolist())

# Convertir machine_id a texto en ambos DataFrames para asegurar
compatibilidad en el merge
df_csv['machine_id'] = df_csv['machine_id'].astype(str)
df_sql['machine_id'] = df_sql['machine_id'].astype(str)

# Unir los datos en un solo DataFrame
df_combinado = pd.merge(df_csv, df_sql, on='machine_id', how='left')

# Mostrar vista previa y verificación
print("\nPreview del DataFrame combinado:")
print(df_combinado.head(10))
print("Valores únicos de 'model':", df_combinado['model'].unique())
print("Cantidad de valores nulos en 'model':",
df_combinado['model'].isnull().sum())

# Eliminar filas que no tienen valor en 'model'
df_combinado_filtrado = df_combinado.dropna(subset=['model'])

# Definir paleta de colores variada
colores = plt.cm.tab10.colors # Paleta de colores 'tab10'

# Definir ruta al logo
logo_path = 'logo_utpl.png' # Nombre del archivo del logo en la misma
carpeta del script

# Función para agregar el logo y los textos institucionales en cada
gráfico
def agregar_logo_texto(fig, ax, titulo):
    ax.set_title(titulo, fontsize=10) # Título del gráfico con tamaño
de fuente adecuado

    # Insertar logo en la parte superior izquierda
    if os.path.exists(logo_path):
        logo = mpimg.imread(logo_path)
        axins = inset_axes(ax, width="10%", height="10%", loc='upper
left')
        axins.imshow(logo)
        axins.axis('off') # Ocultar bordes del logo

    # Insertar textos institucionales en la parte inferior derecha
    fig.text(0.99, 0.02, "Universidad Técnica Particular de Loja",
fontsize=6, ha='right')
    fig.text(0.99, 0.045, "Maestría en Inteligencia Artificial
Aplicada", fontsize=6, ha='right')

```

```

fig.text(0.99, 0.07, "Módulo: Herramientas para Inteligencia
Artificial", fontsize=6, ha='right')
fig.text(0.99, 0.095, "Estudiante: Ing. José Gabriel Aguirre
Andrade, MSc.", fontsize=6, ha='right')
fig.text(0.99, 0.12, "Docente: Ph.D. Alexandra Cristina Gonzalez
Eras", fontsize=6, ha='right')

# --- GRÁFICO 1: Tiempo promedio por modelo ---
fig1, ax1 = plt.subplots(figsize=(8, 4))
df_combinado_filtrado.groupby('modelo')['tiempo_segundos'].mean().plot(kind='bar', ax=ax1, color=colores)
ax1.set_ylabel('Tiempo Promedio (segundos)')
ax1.set_xlabel('Modelo de Máquina', fontsize=8)
ax1.tick_params(axis='x', labelrotation=90, labelsiz=7) # Etiquetas en
vertical con fuente más pequeña
agregar_logo_texto(fig1, ax1, 'Tiempo Promedio de Eventos por Modelo de
Máquina')
plt.tight_layout()
plt.show()

# --- GRÁFICO 2: Cantidad de eventos por modelo ---
fig2, ax2 = plt.subplots(figsize=(8, 4))
df_combinado_filtrado['modelo'].value_counts().plot(kind='bar', ax=ax2,
color=colores)
ax2.set_ylabel('Número de Eventos')
ax2.set_xlabel('Modelo de Máquina', fontsize=8)
ax2.tick_params(axis='x', labelrotation=90, labelsiz=7)
agregar_logo_texto(fig2, ax2, 'Cantidad de Eventos por Modelo de
Máquina')
plt.tight_layout()
plt.show()

# --- GRÁFICO 3: Eventos por operario ---
fig3, ax3 = plt.subplots(figsize=(8, 4))
df_combinado_filtrado['operario'].value_counts().plot(kind='bar',
ax=ax3, color=colores)
ax3.set_ylabel('Cantidad de Eventos')
ax3.set_xlabel('Operario', fontsize=8)
ax3.tick_params(axis='x', labelrotation=90, labelsiz=7)
agregar_logo_texto(fig3, ax3, 'Eventos Registrados por Operario')
plt.tight_layout()
plt.show()

# --- GRÁFICO 4: Duración promedio por operario ---
fig4, ax4 = plt.subplots(figsize=(8, 4))
df_combinado_filtrado.groupby('operario')['tiempo_segundos'].mean().plot(
kind='bar', ax=ax4, color=colores)
ax4.set_ylabel('Tiempo Promedio (segundos)')
ax4.set_xlabel('Operario', fontsize=8)
ax4.tick_params(axis='x', labelrotation=90, labelsiz=7)
agregar_logo_texto(fig4, ax4, 'Duración Promedio de Eventos por
Operario')
plt.tight_layout()
plt.show()

# --- GRÁFICO 5: Eventos por tipo de evento ---
fig5, ax5 = plt.subplots(figsize=(8, 4))

```

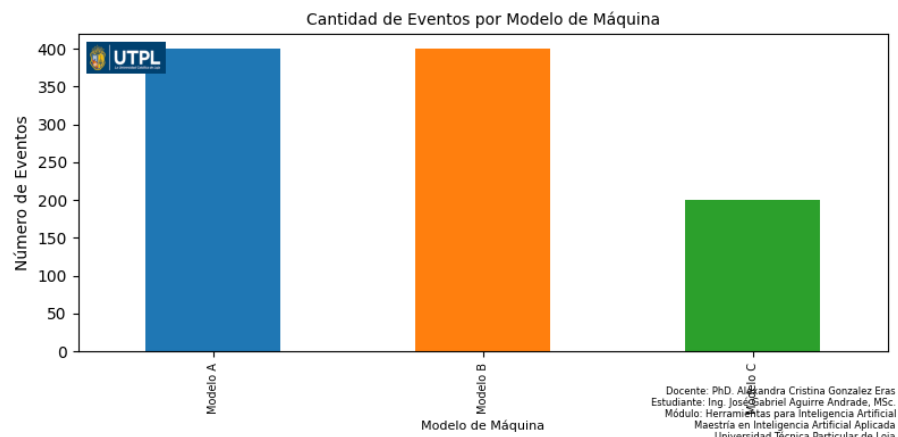
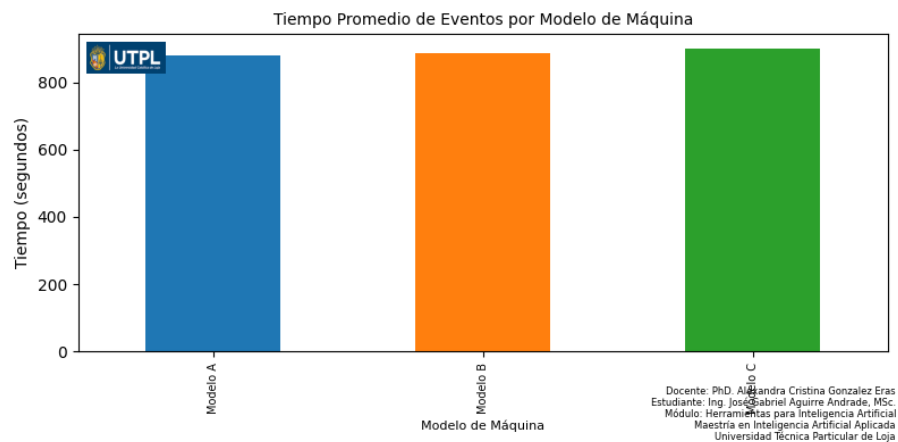
```

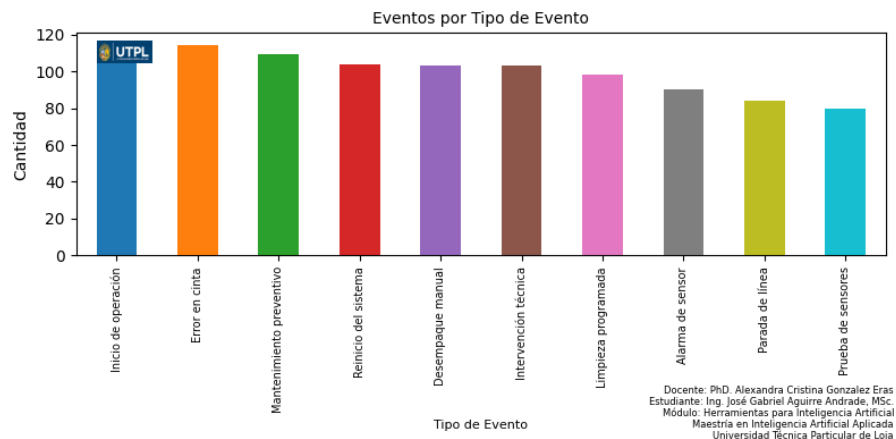
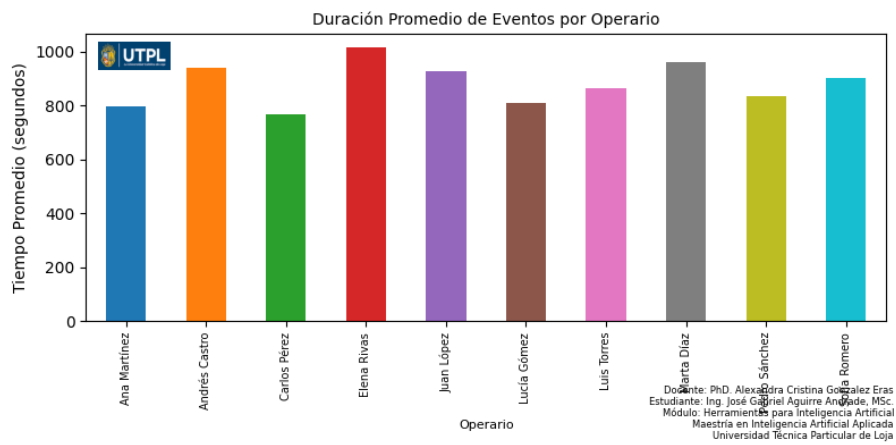
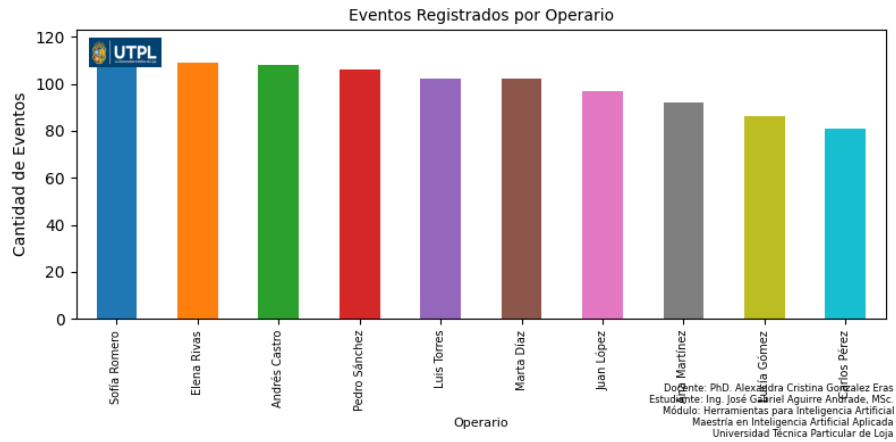
df_combinado_filtrado['evento'].value_counts().plot(kind='bar', ax=ax5,
color=colores)
ax5.set_ylabel('Cantidad')
ax5.set_xlabel('Tipo de Evento', fontsize=8)
ax5.tick_params(axis='x', labelrotation=90, labelsiz=7)
agregar_logo_texto(fig5, ax5, 'Eventos por Tipo de Evento')
plt.tight_layout()
plt.show()

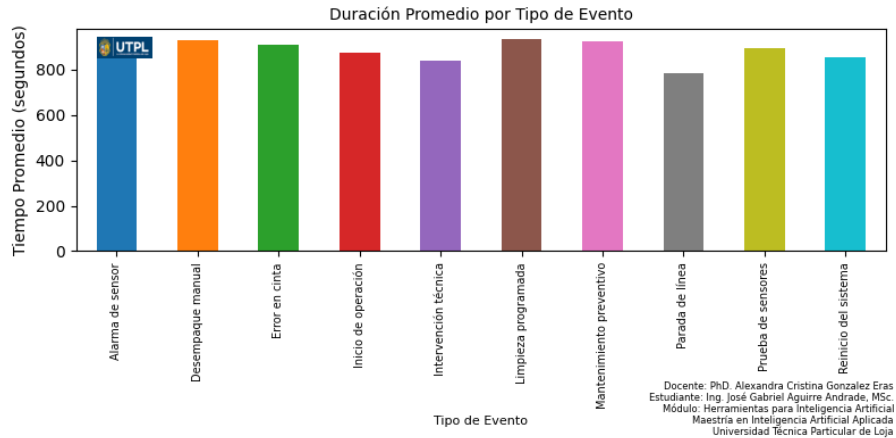
# --- GRÁFICO 6: Duración promedio por tipo de evento ---
fig6, ax6 = plt.subplots(figsize=(8, 4))
df_combinado_filtrado.groupby('evento')['tiempo_segundos'].mean().plot(k
ind='bar', ax=ax6, color=colores)
ax6.set_ylabel('Tiempo Promedio (segundos)')
ax6.set_xlabel('Tipo de Evento', fontsize=8)
ax6.tick_params(axis='x', labelrotation=90, labelsiz=7)
agregar_logo_texto(fig6, ax6, 'Duración Promedio por Tipo de Evento')
plt.tight_layout()
plt.show()

```

Resultados







7. Conclusiones

El análisis de los datos operativos y técnicos de los paletizadores en la planta embotelladora revela hallazgos significativos que podrían impactar directamente en la toma de decisiones y optimización de procesos:

- **Eficiencia diferencial según modelos de paletizadores:** Al examinar los registros operativos, se evidenció que determinados modelos requieren intervalos más prolongados para la resolución de eventos, lo cual podría responder a complejidades inherentes a su diseño o a características específicas de su mecánica. Estos hallazgos permiten reconsiderar la distribución de carga de trabajo entre máquinas y priorizar intervenciones técnicas.
- **Distribución irregular de eventos entre operarios:** Se constató una marcada heterogeneidad en la cantidad y duración de eventos gestionados por cada técnico. Esta variabilidad podría originarse en factores como niveles de experiencia, asignación de turnos o responsabilidades específicas. Un análisis más detallado de estas disparidades facilitaría la implementación de programas de capacitación focalizados y la redistribución equitativa de responsabilidades.
- **Impacto diferenciado según tipología de eventos:** Los datos revelaron que ciertos tipos de eventos generan interrupciones sustancialmente más prolongadas que otros, lo cual permite identificar áreas críticas para intervención prioritaria. La concentración de esfuerzos en reducir la duración de estos eventos específicos podría resultar en mejoras significativas en la disponibilidad general de los equipos.
- **Valor probado de la integración multifuente de datos:** La metodología empleada, combinando información estructura y no estructurada desde diversas fuentes, permitió obtener perspectivas que hubieran sido inaccesibles mediante análisis aislados. Esta aproximación

- multidimensional demostró ser especialmente valiosa para detectar correlaciones entre características técnicas y comportamiento operativo.
- **Aplicabilidad industrial de metodologías de ciencia de datos:** El proyecto evidenció cómo herramientas tradicionalmente asociadas al análisis estadístico académico pueden implementarse exitosamente en contextos industriales para transformar datos rutinarios en estrategias accionables de mejora continua y toma de decisiones basada en evidencia.

Estas conclusiones sugieren oportunidades para optimizar la programación de mantenimiento, la asignación de personal y potencialmente la selección de modelos de paletizadores en futuras adquisiciones, basándose en datos objetivos de rendimiento y eficiencia.

8. Bibliografía

McKinney, W. (2010). *Data Structures for Statistical Computing in Python*. Proceedings of the 9th Python in Science Conference, 51–56.

Render. (2024). *Render PostgreSQL Documentation*. Recuperado de <https://render.com/docs/databases>

Google. (2024). *Colaboratory: Welcome to Colab*. Recuperado de <https://colab.research.google.com>

Pandas Documentation. (2024). <https://pandas.pydata.org>

Matplotlib Documentation. (2024). <https://matplotlib.org>

SQLAlchemy Documentation. (2024). <https://www.sqlalchemy.org>