



# Tecnológico de Monterrey

Inteligencia artificial avanzada para la ciencia de datos 2

Gpo 501

## Docentes

Dr. Benjamín Valdés Aguirre

Ma. Eduardo Daniel Juárez Pineda

Dr. Ismael Solis Moreno

Dr. José Antonio Cantoral Ceballos

Dr. Carlos Alberto Dorantes Dosamantes

## Integrantes

Carlos Rodrigo Salguero Alcántara	A00833341
-----------------------------------	-----------

Diego Perdomo Salcedo	A01709150
-----------------------	-----------

Dafne Fernández Hernández	A01369230
---------------------------	-----------

José Emiliano Riosmena Castañón	A01704245
---------------------------------	-----------

Luis Arturo Rendón Iñarritu	A01703572
-----------------------------	-----------

Querétaro, Querétaro

<b>1. Introducción</b>	<b>2</b>
1.1 Contexto del cliente	2
1.2 Contexto del proyecto	2
<b>2. Entendimiento de los datos</b>	<b>2</b>
2.1 Adquisición de datos	2
2.2 Caracterización del conjunto de datos	3
2.3 Análisis exploratorio de Datos	3
2.4 Examinar la calidad de los datos	7
<b>3. Alcance</b>	<b>8</b>
3.1 Contexto de los datos	8
3.2 Contexto del Cliente	8
3.3 Contexto utilizado	9

# **1. Introducción**

## **1.1 Contexto del cliente**

El Centro Agropecuario Experimental del Tec de Monterrey (CAETEC) es una unidad de producción e investigación que sirve como laboratorio viviente para estudiantes y profesores. Este campo experimental está dedicado a la formación académica, investigación y desarrollo de proyectos en el sector agropecuario, donde se llevan a cabo prácticas profesionales, servicio social, proyectos de investigación y actividades docentes. En sus instalaciones ganaderas, opera un sistema de producción lechera que cuenta con tres corrales equipados con un sistema de monitoreo, y tienen capacidad para tres máquinas de ordeño automático.

## **1.2 Contexto del proyecto**

Se desarrollará un modelo de machine learning para la identificación y conteo de vacas, utilizando específicamente las imágenes capturadas de un solo corral. El dataset para este proyecto consta de 8,110 imágenes, las cuales serán utilizadas para entrenar un modelo que permita identificar la congestión en la fila de espera para el ordeño.

# **2. Entendimiento de los datos**

En esta fase realizamos el entendimiento de los datos, para ello se hicieron actividades como la adquisición de datos hasta la evaluación de la calidad del dataset.

## **2.1 Adquisición de datos**

Se llevó a cabo utilizando una webcam Logitech instalada en el techo del corral, capturando imágenes desde una vista superior. Este enfoque permitió obtener datos visuales en ciclos continuos tanto de día como de noche. El conjunto de datos fue compartido y descargado mediante la plataforma OneDrive, garantizando un acceso centralizado y eficiente para el equipo de análisis.

Considerando que los datos fueron otorgados por el CAETEC no tuvimos problemas en adquirirlos. Lo único que pudiera llegar a ser un problema es que si queremos adquirir una mayor cantidad de imágenes, se deben de solicitar con anticipación.

## 2.2 Caracterización del conjunto de datos

El dataset contiene un aproximado de 8110 imágenes con resolución de 1920x1080, clasificados según el número de vacas presentes. La principal característica del conjunto es su variación extrema de iluminación, con imágenes muy oscuras o completamente negras durante la noche y alta luminosidad con posible sobreexposición en el día.

## 2.3 Análisis exploratorio de Datos

Para el análisis exploratorio de los datos, se empleó Tableau y un jupyter notebook. Los resultados obtenidos son los siguientes:

Se diseña un script para la generación de un csv con los links y meta-data de todas las imágenes en el s3.

Creamos la función main y escribimos en el meta-data

```
def main():
    # Inicializar el cliente de S3
    s3 = boto3.client('s3')

    # Listar todos los objetos en el bucket
    print("Listando objetos en el bucket ... ")
    all_objects = list_all_objects(s3, BUCKET_NAME)
    total_objects = len(all_objects)
    print(f"Total de objetos en el bucket: {total_objects}")

    if total_objects == 0:
        print("No se encontraron objetos en el bucket.")
        return

    # Elegir 300 objetos al azar si hay suficientes, si no, toma todos
    sample_size = min(total_objects, MAX_IMAGES)
    sampled_objects = random.sample(all_objects, sample_size)
    print(f"Seleccionando {sample_size} objetos al azar para procesar ... ")

    # Definir los encabezados del CSV
    headers = [
        'Key', 'Url', 'LastModified', 'Size', 'ETag',
        'StorageClass',
        'Width', 'Height', 'Luminosity', 'Contrast',
        'Histogram_Red', 'Histogram_Green', 'Histogram_Blue'
    ]

    # Procesar los objetos en paralelo para mejorar el rendimiento
    print("Procesando objetos y extrayendo metadata ... ")
    processed_data = []
    with ThreadPoolExecutor(max_workers=MAX_WORKERS) as executor:
        # Enviar tareas al pool de hilos
        futures = {executor.submit(process_object, s3, BUCKET_NAME, obj): obj for obj in sampled_objects}
        for future in as_completed(futures):
            data = future.result()
            processed_data.append(data)

    # Escribir los datos en el CSV
```

Luego otro script prepara el CSV para que pueda ser utilizado en Tableau, esto para la separación de los datos como de histogramas.

## Código para preparar el CSV para tableau

```
import pandas as pd

# Cargar el CSV
df = pd.read_csv('s3_image_metadata_advanced_with_histograms.csv')

# Función para dividir los histogramas en filas
def explode_histogram_columns(df, columns):
    """
    Divide las columnas de histogramas en múltiples filas.
    """
    # Dividir las columnas de histogramas y expandir las filas
    for column in columns:
        # Convertir las cadenas separadas por ';' en listas
        df[column] = df[column].apply(lambda x: list(map(int, x.split(';'))))

    # Usar 'pd.DataFrame.explode' para expandir las listas en filas
    df = df.explode(columns)

    return df

# Especificar las columnas que contienen los histogramas
histogram_columns = ['Histogram_Red', 'Histogram_Green', 'Histogram_Blue']

# Dividir los histogramas en múltiples filas
df_exploded = explode_histogram_columns(df, histogram_columns)

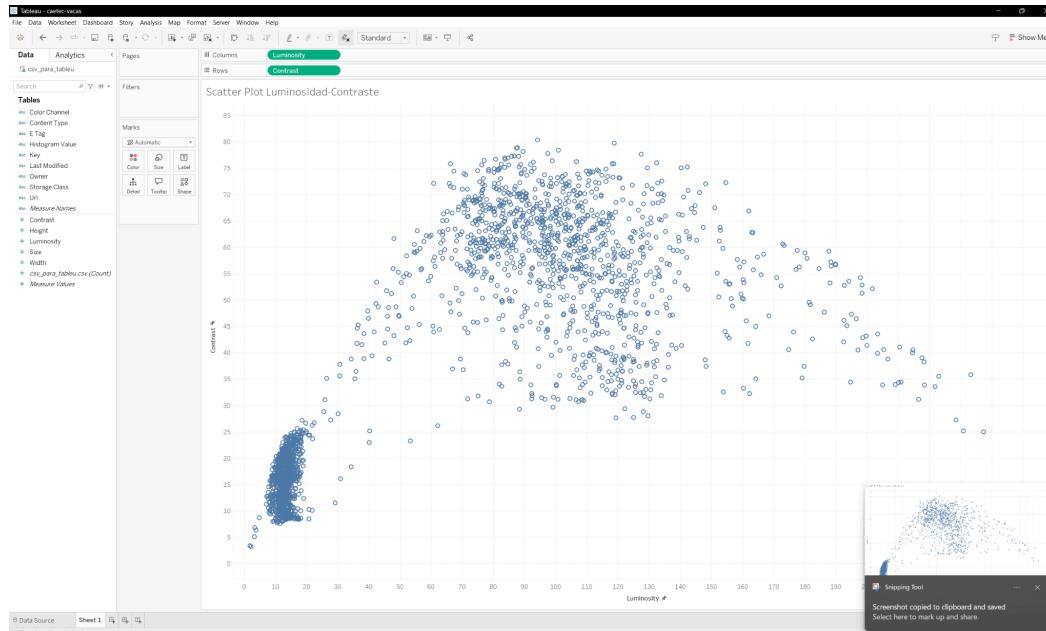
# Crear un nuevo DataFrame con las columnas para cada canal de color
df_red = df[['Key', 'Url', 'LastModified', 'Size', 'ETag', 'StorageClass', 'Owner', 'ContentType', 'Width', 'Height', 'Luminosity', 'Contrast', 'Histogram_Value']]
df_green = df[['Key', 'Url', 'LastModified', 'Size', 'ETag', 'StorageClass', 'Owner', 'ContentType', 'Width', 'Height', 'Luminosity', 'Contrast', 'Histogram_Value']]
df_blue = df[['Key', 'Url', 'LastModified', 'Size', 'ETag', 'StorageClass', 'Owner', 'ContentType', 'Width', 'Height', 'Luminosity', 'Contrast', 'Histogram_Value']]

# Renombrar la columna del histograma para que sea común en todas
df_red = df_red.rename(columns={'Histogram_Red': 'Histogram_Value'})
df_green = df_green.rename(columns={'Histogram_Green': 'Histogram_Value'})
df_blue = df_blue.rename(columns={'Histogram_Blue': 'Histogram_Value'})

# Añadir la columna 'Color_Channel' a cada DataFrame
df_red['Color_Channel'] = 'Red'
df_green['Color_Channel'] = 'Green'
```

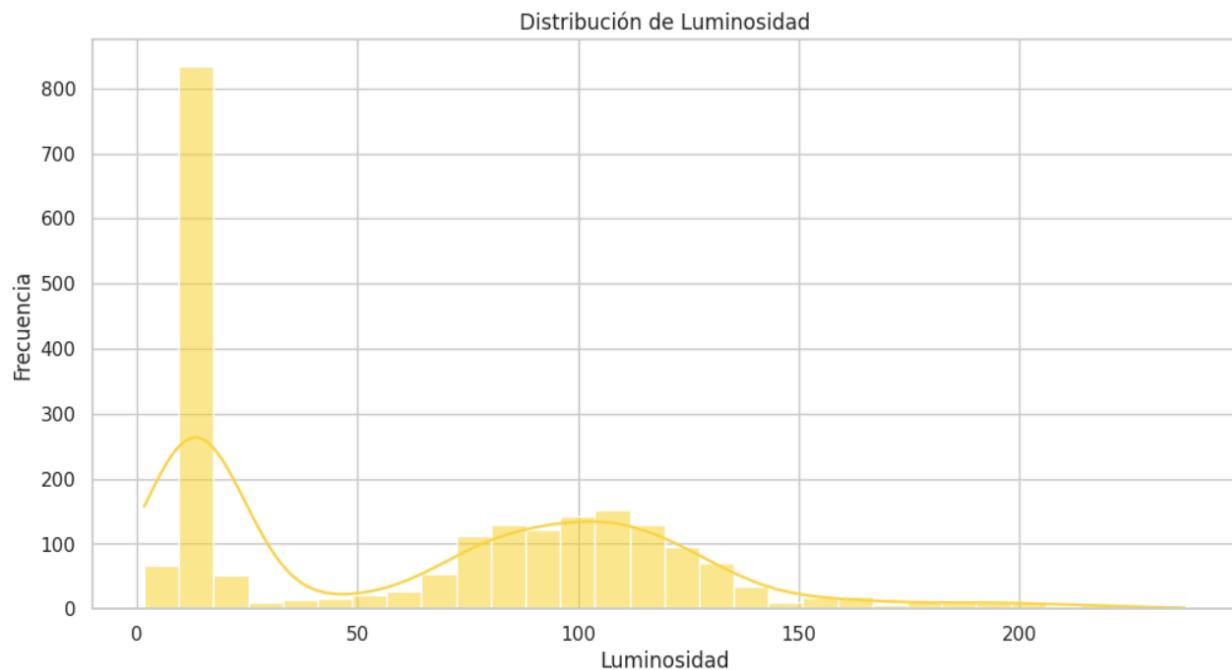
Dicho CSV puede ser utilizado en Tableau.

La primera gráfica es un ScatterPlot donde se muestra la luminosidad y el contraste de las imágenes.



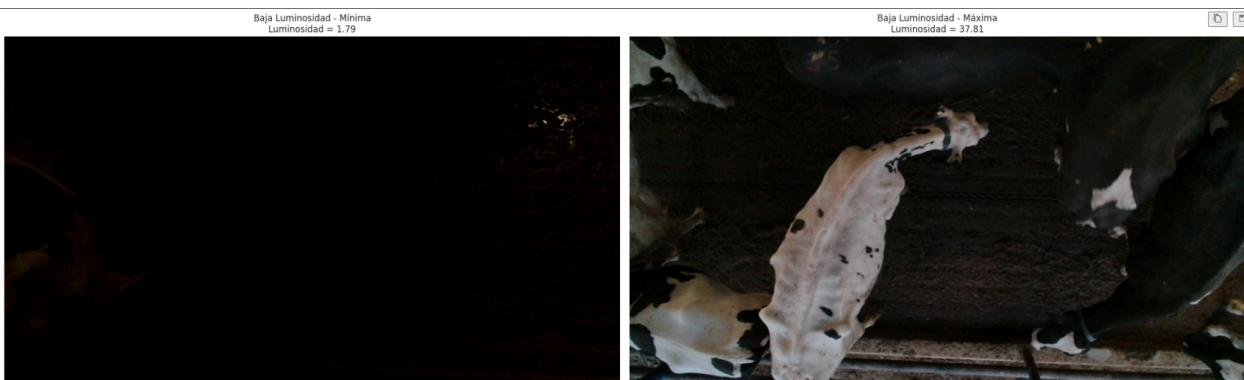
La siguiente representa la luminosidad presente en las imágenes y con qué frecuencia aparece.

Se pueden ver los dos conjuntos de imágenes que poseemos, las que son en la noche, estando todas en un rango muy similar entre 10 y 20 de luminosidad y las que son durante el día. Estas como el sol va cambiando de posición presenta una luminosidad cambiante entre 75 y 125 aproximadamente. Es interesante como se ve el movimiento del sol en esta parte del espectro.



También se dividieron las imágenes en 3 grupos, luminosidad baja, media y alta.

Desplegamos las imágenes con la luminosidad mínima y máxima en cada grupo.





Al final se tomó la decisión de solo tener dos grupos, uno de día y uno de noche, pero pudimos ver en las representaciones superiores que no era necesario hacer un grupo de luminosidad excesiva o de outliers.

Ademas con este análisis podemos observar que el nivel de iluminación en las fotos de noche las vacas apenas se ven. Esto puede llegar a impactar negativamente el modelo nocturno lo cual puede causar un fuerte impacto en los resultados finales del proyecto.

## 2.4 Verificar la calidad de los datos

Al analizar los datos, observamos que están completos y que disponemos de una cantidad adecuada de información para trabajar. Dado que todas las imágenes fueron capturadas con la misma cámara en una posición fija, el formato entre ellas es consistente. Sin embargo, debido a la variabilidad en los niveles de iluminación, podrían surgir problemas al desarrollar el modelo. Para abordar esta diferencia, se desarrollarán dos modelos: uno para imágenes diurnas y otro para

nocturnas. Esto permitirá que todas las fotos contribuyan de manera efectiva, mejorando la precisión de las predicciones. Lo malo es que al hacer esta división de datasets nos quedamos con menos imágenes para entrenar y probar el modelo.

## 3.0 Alcance

Este proyecto, aunque maneja imágenes de vigilancia continua, no requiere soluciones de Big Data. Explicamos esto analizando las características típicas de Big Data y por qué nuestro proyecto no las cumple:

### 3.1 Contexto Actual

Actualmente trabajamos con aproximadamente 8,000 imágenes de un solo corral. Este volumen de datos puede manejarse fácilmente en una computadora moderna sin necesidad de procesamiento distribuido o almacenamiento especial. Los datos son simples en el sentido de que:

- Son únicamente imágenes de un formato específico
- Provienen de una sola cámara
- El análisis se enfoca solo en contar vacas
- No requiere procesamiento en tiempo real

### 3.2 Contexto Futuro

El cliente planea expandir el sistema instalando cámaras en dos corrales adicionales, lo que triplicaría el volumen a aproximadamente 24,000 imágenes. Sin embargo, incluso con esta expansión:

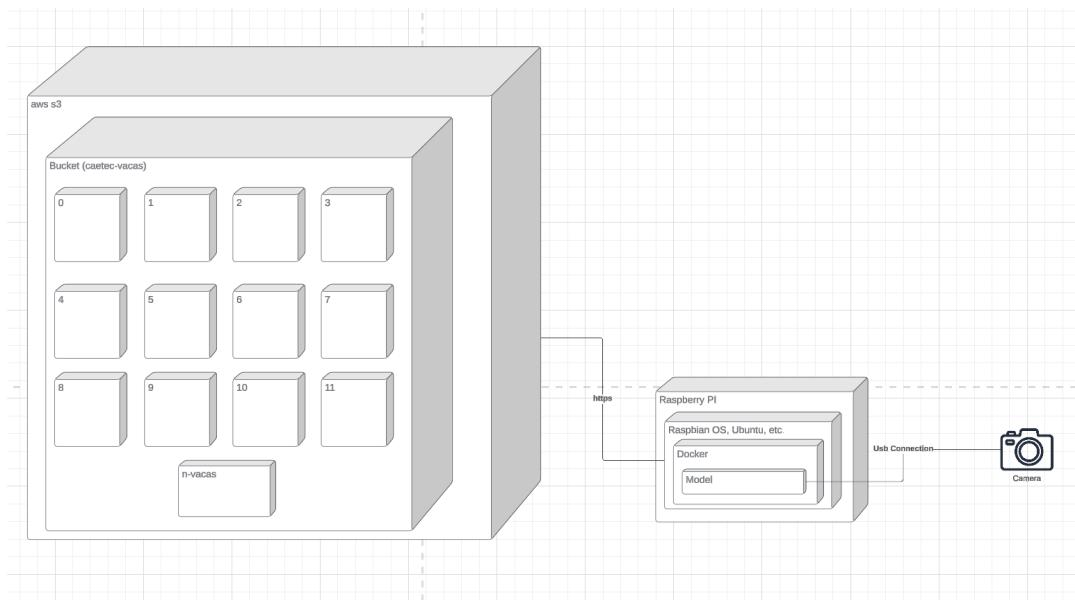
- El volumen sigue siendo manejable con herramientas convencionales
- La naturaleza de los datos permanece simple
- No se requiere infraestructura especializada

### 3.3 Decisión de Alcance

Aunque actualmente trabajamos con datos de un corral, decidimos diseñar la solución pensando en la futura expansión a tres corrales. Esto asegura que nuestro modelo será útil cuando el cliente

implemente las cámaras adicionales, sin necesidad de modificaciones significativas en la arquitectura de la solución.

## 4.0 Arquitectura y flujo de datos



- 1- La nueva información será obtenida por la cámara web colocada en la parte superior de la fila de ordeño.
- 2- El modelo estará ubicado en la Raspberry Pi, el cual recibe la imagen mandada por la cámara y la clasifica.
- 3- Dependiendo de la categoría que le asigne el modelo, la imagen viaja a través de http rumbo a AWS S3 para ser almacenada en la carpeta correspondiente.

## 5.0 Herramientas

Fases de CRISP-DM	Herramientas del proyecto	Herramientas si fuera un proyecto de Big Data
-------------------	---------------------------	---

<b>Comprensión de negocio</b>	<ul style="list-style-type: none"> <li>❖ Microsoft Excel para el plan de valor ganado del proyecto</li> <li>❖ Google Docs para la redacción de la documentación</li> </ul>	<ul style="list-style-type: none"> <li>❖ Tableau/PowerBi para la visualización de métricas</li> </ul>
<b>Comprensión de datos</b>	<ul style="list-style-type: none"> <li>❖ OpenCV para procesamiento inicial de imágenes de ganado vacuno</li> <li>❖ PIL para manipulación básica de imágenes</li> <li>❖ Matplotlib para visualizar patrones de identificación</li> <li>❖ Tableau para análisis de metadatos de las imágenes</li> </ul>	<ul style="list-style-type: none"> <li>❖ Spark Image para procesamiento distribuido de imágenes</li> <li>❖ HDFS para almacenamiento de imágenes en formato top-view</li> <li>❖ MongoDB para metadata de imágenes a gran escala</li> <li>❖ AWS S3 para almacenamiento cloud de imágenes</li> </ul>
<b>Preparación de datos</b>	<ul style="list-style-type: none"> <li>❖ Roboflow para etiquetado de características del ganado</li> <li>❖ OpenCV para preprocessamiento (contraste, brillo, normalización)</li> <li>❖ Albumentations para</li> </ul>	<ul style="list-style-type: none"> <li>❖ Spark ML para preprocessamiento distribuido</li> <li>❖ Tensorflow Data para pipelines optimizados</li> <li>❖ Petastorm para lectura eficiente</li> <li>❖ DALI para carga</li> </ul>

	<p>aumentación (rotación, zoom, cambios de iluminación)</p> <ul style="list-style-type: none"> <li>❖ NumPy para operaciones matriciales</li> <li>❖ ImageDataGenerator de Keras para batching</li> </ul>	rápida de datos
<b>Modelado</b>	<ul style="list-style-type: none"> <li>❖ Tensorflow / Keras para CNNs de clasificación</li> <li>❖ Modelo transfer learning (ResNet50V2)</li> <li>❖ Callbacks para checkpoints</li> </ul>	<ul style="list-style-type: none"> <li>❖ Distributed TensorFlow</li> <li>❖ Horovod para entrenamiento multi-GPU</li> <li>❖ Ray para paralelización</li> </ul>
<b>Evaluación</b>	<ul style="list-style-type: none"> <li>❖ Scikit-learn para métricas de clasificación (accuracy, precision, recall y f1-score)</li> <li>❖ TensorBoard para curvas de entrenamiento</li> <li>❖ Matplotlib / seaborn para matrices de confusión</li> </ul>	<ul style="list-style-type: none"> <li>❖ MLflow para tracking</li> <li>❖ Weight &amp; biases para visualización</li> <li>❖ Prometheus para métricas en tiempo real</li> </ul>
<b>Implementación</b>	<ul style="list-style-type: none"> <li>❖ Docker para containerización</li> </ul>	<ul style="list-style-type: none"> <li>❖ Kubernetes para escalado</li> <li>❖ Tensorflow serving</li> <li>❖ PostgreSQL para logging escalable</li> </ul>
<b>Generales</b>	<ul style="list-style-type: none"> <li>❖ Git para control de</li> </ul>	<ul style="list-style-type: none"> <li>❖ Git LFS para datasets</li> </ul>

	<ul style="list-style-type: none"> <li>versiones</li> <li>❖ Python como lenguaje principal</li> <li>❖ Jupyter Notebook para experimentación</li> <li>❖ VSCode, Google Colab y Kaggle como ambientes de desarrollo</li> </ul>	<ul style="list-style-type: none"> <li>grandes</li> <li>❖ Python optimizado</li> <li>❖ JupyterHub multi-usuario</li> <li>❖ Cloud GPUs (AWS)</li> <li>❖ Docker Swarm / Kubernetes</li> </ul>
--	--	---

## Implementación de herramientas seleccionadas

Considerando que el proyecto no entra en la categoría de Big Data, las siguientes herramientas han sido utilizadas para la implementación actual.

### Almacenamiento y gestión de datos

Se implementó Amazon S3 como solución principal de almacenamiento, siguiendo prácticas, organización y seguridad:

- ❖ Organización jerárquica en carpetas de train, test y validation
- ❖ Subcarpetas clasificadas por cantidad de vacas y agrupadas por día y noche
- ❖ Control de acceso mediante políticas IAM de AWS
- ❖ Sin acceso público para protección de datos del cliente

### Procesamiento y Etiquetado

Para el procesamiento inicial y etiquetado de imágenes, se implementó Roboflow debido a su especialización en visión computacional:

- ❖ Clasificación de las imágenes
- ❖ Gestión del dataset

### Infraestructura de Procesamiento

El procesamiento de datos se realiza mediante una arquitectura híbrida:

#### Procesamiento local

- ❖ Instancias de Google Colab y Kaggle

## Procesamiento en Campo

### ❖ Hardware implementado

- Raspberry Pi con Raspbian OS
- Configuración optimizada para captura de imágenes
- Sistema operativo ligero y eficiente

## Visualización de los datos

Se implementó Tableau como herramienta principal de visualización para analizar los metadatos de las imágenes. El tablero tiene una capacidad ajustable según demanda y se integra con AWS S3 para acceder a las imágenes.

