ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERÍA

Trabajo de Grupo Curso Servicios de Tribología en IIoT y ML

Agosto, 2024

Alumnos:

Felipe Suazo
Francisco Peña
Kevin Alba
Fabian Rojas
Camilo Henríquez

Tabla de Contenido

1 Introducción

El presente reporte corresponde al trabajo realizado para el curso "***Servicios de Tribología Basados en IIoT y ML***", del diplomado en ***"Internet of Things y machine learning aplicado a la minería"***, impartido por la Universidad Católica.

Este documento se desarrolla como sigue. La sección 2 entrega una revisión general del trabajo realizado, con un mayor detalle de la estrategia definida para la construcción de los modelos requeridos. La sección siguiente responde las preguntas solicitadas como parte del trabajo.

2 Consideraciones del Modelamiento

2.1 Introducción

Para el desarrollo del proceso de modelamiento se tomó como punto de partida la base analítica obtenida de aplicar el código de la sección de EDA (exploratory data analysis) del código propuesto. Al respecto cabe hacer presente las siguientes simplificaciones adoptadas en el modelamiento:

1. **Sub-utilización de la data disponible**

La tabla de resultados (*results* según esquema del modelo de datos), contenía un mayor número de campos, tales como lic, lim, lsm y lsc. Si bien las columnas lic y lim eran nulas, los campos lsm (límite superior marginal) y lsc (límite superior condenatorio), presentaban data para aproximadamente el 50% de los registros, se prefirió descartar su uso por cuanto no existía un mayor entendimiento del fenómeno de la tribología por parte del grupo de trabajo.

Situación similar aconteció con la data asociada a componentes y tipo de máquinas. Si bien la intuición indica que sería posible segmentar la data en familias de maquinarias o equipos, no se pudo llegar a un número razonable para hacer posible un análisis más detallado. Por lo tanto, se descartó el uso de dichas columnas.

2. **Registros con datos nulos**

Si bien en el mundo real la disponibilidad de registros con datos nulos efectivamente acontece, con lo cual un modelo de machine learning debiera considerar esa casuística en el proceso de predicción (de hecho un dato nulo desde el punto de vista de la teoría de la información tiene impacto en el cálculo de la entropía dependiendo del contexto), se decidió su descarte, principalmente por el desconocimiento del fenómeno.

2.2 Técnicas Utilizadas en el Modelamiento

En Machine Learning es sabido que el proceso de modelamiento es del tipo ***"Trial and Error"***, por lo cual es recomendable utilizar una amplia gama de técnicas y algoritmos de tal forma de obtener un modelo que pueda identificar los patrones implícitos en la data y que sea capaz de generalizar ante data nueva.

Por lo anterior, se optó por una combinación de algoritmos en el rango ***"high variance - high bias"***. Un ejemplo de algoritmo con alto error de varianza corresponde a los árboles de decisión, donde el algoritmo incluso puede aprender del ruido de los datos. Por su parte, un ejemplo de modelo con alto error de sesgo corresponde a la regresión lineal, donde, aún para datos no lineales el algoritmo siempre va a entregar como solución una recta (o plano). Así, las técnicas utilizadas fueron las siguientes:

1. **Decision Tree:**

   Corresponde a una técnica con un alto error por varianza y bajo error por sesgo. Su principal desventaja es que pueden caer en overfitting (sobre ajuste) incluso ajustando perfectamente el ruido de la data.

2. **Random Forest:**

   Es una técnica que presenta un error por sesgo moderado y un bajo error por varianza (si se compara con los árboles de decisión). Su principal ventaja es que reduce el error de varianza de los árboles de decisión, sin embargo, podría presentar algún sesgo dependiendo de su profundidad (número de niveles) y número de estimadores (árboles).

3. **AdaBoost:**

   Esta técnica presenta un sesgo bajo, pero una alta varianza, similar a los árboles de decisión. Su desventaja es que puede aprender el ruido de los datos.

4. **Bagging:**

   Presenta un bajo a moderado error sesgo y bajo error de varianza. Su principal ventaja es que reduce el error de varianza al promediar las predicciones de múltiples modelos (generalmente árboles de decisión).

5. **SVM:**

   Su error de sesgo depende del tipo de kernel y técnica de regularización usada, donde el kernel lineal tiende a presentar mayor error por sesgo en comparación con los kernels no lineales. Lo contrario ocurre con el error por varianza, donde el kernel lineal tiene menor error por varianza comparado con los kernels no lineales. En resumen, esta técnica puede presentar ya sea un alto error por sesgo o varianza dependiendo del kernel utilizado, por lo que siempre es necesario considerar el uso de los parámetros de regularización.

6. **Logistic Regression:**

   Esta técnica presenta un bajo error por varianza, pero un alto error por sesgo. Esta es una técnica que pertenece al mundo de los modelos lineales por lo cual presenta un error de sesgo cuando se aplica a datos no lineales o de alta complejidad. Sin embargo, usualmente presenta bajo error de varianza por lo cual, comparativamente a otras técnicas, es menos proclive al overfitting.

2.3 Estrategia de Modelamiento

El proceso de modelamiento consiste en la búsqueda del ***mejor*** modelo en el espacio de soluciones factibles, que siempre es infinito. De ahí que una estrategia habitualmente utilizada en Machine Learning es la de ***Grid Search*.** Esta es una técnica utilizada para la búsqueda de los ***hiperparámetros*** de un modelo. Los ***hiperparámetros*** son parámetros de configuraciones de un modelo que se establecen antes del entrenamiento y que no se aprenden a partir de los datos (por ejemplo, el tipo de kernel y el valor de sus parámetros en SVM, el valor de regularización en una regresión logística, etc.). Para lo anterior se utilizó el objeto ***RandomizedSearchCV*** de la librería ***open source*** de machine learning para python ***scikit-learn***.

Finalmente, para el caso de regresión logística y SVM, los atributos fueron normalizados utilizando el objeto ***RobustScaler*** de ***scikit-learn***. En efecto, la normalización de los atributos de entrada para este tipo de modelos es considerado una buena práctica, especialmente cuando se trabaja con atributos que tienen diferentes escalas o cuando

se utiliza regularización. Esto asegura que el modelo converja más rápido, se comporte de manera consistente y proporcione resultados más confiables.

3 Respuestas a Preguntas

3.1 Investigue el concepto de "clase desbalanceada en problemas de clasificación" y responda

a) ¿Qué Significa el concepto de clases desbalanceadas?¿Cuándo sucede?.
b) ¿Es este un caso de clases desbalanceadas?.¿Cómo se podría remediar?.

Respuestas:

Pregunta a:

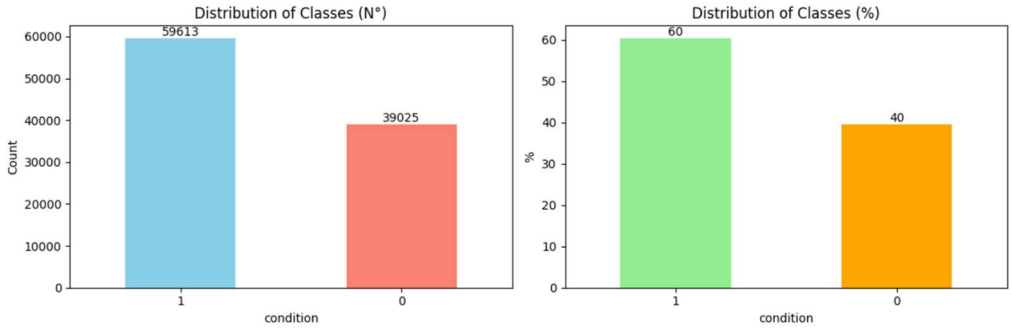
En problemas de clasificación, una **clase desbalanceada** se refiere a una situación en la que las diferentes clases (o etiquetas) no tienen una representación equitativa en el conjunto de datos. Es decir, una clase tiene significativamente más evidencia (ejemplos) que las otras. Generalmente este desbalance afecta negativamente la performance de los modelos. En efecto, en la presencia de desbalances, los algoritmos tienden a favorecer la identificación de los patrones presentes en la clase mayoritaria, ignorando o subestimando la clase minoritaria.

Un ejemplo habitualmente referido de este problema corresponde al caso de transacciones de tarjetas de créditos fraudulentas. A nivel global, se estima que el porcentaje de transacciones fraudulentas oscila entre el 0.1% y el 0.4% del total de transacciones con tarjetas de crédito. Aunque este porcentaje parece bajo, el impacto económico es significativo debido al gran volumen de transacciones que se realizan a nivel mundial. Un modelo construido sin considerar este desbalance probablemente llevará a métricas de ***Precision*** muy cercana a 100% y ***Recall*** (es decir, cuántas de las transacciones realmente fraudulentas fueron detectadas por el modelo) de cero.

Pregunta b:

El problema propuesto corresponde a un caso de clases desbalanceadas donde la clase ***normal*** corresponde al 60% de la evidencia. Sin embargo, para fines del modelamiento, esta ratio corresponde a un desbalance leve, pues la abundancia relativa de la clase ***anormal*** aún permite que los algoritmos puedan identificar patrones característicos de

ella, cuestión que queda en evidenciada por la calidad de las métricas de desempeño obtenidas para los diversos modelos construidos.



Si bien es un tema opinable, se considera como desbalance moderado a alto a partir de 1:5 a 1:10 (es decir 20% a 10% de evidencia de la clase minoritaria). Ahora, en casos de desbalance severo (es decir donde la clase minoritaria no llega al 10%), con toda seguridad los modelos construidos tendrán un sesgo hacia la clase mayoritaria.

El ejemplo típico de problema con desbalance severo, ocurre con las transacciones de tarjeta de crédito fraudulentas donde la clase minoritaria no llega al 1%, o como también ocurre en el caso de los score de créditos, donde la clase minoritaria (cartera vencida) no llega al 6% aún en los casos más extremos.

Para resolver problemas de desbalance de clases, la literatura propone las siguientes estrategias de solución:

a. **Recolección de más evidencia**
   Como mejor solución, se recomienda, cuando sea factible, recolectar más ejemplos de la clase minoritaria para ayudar a balancear las clases.

b. **Técnicas de Muestreo**
   La práctica habitual es aplicar ***submuestreo*** (undersampling), que corresponde a reducir el número de ejemplos de la clase mayoritaria para aproximarlo al de la clase minoritaria. Otra técnica de muestreo menos utilizada es el **sobremuestreo** (oversampling), donde se aumenta el número de la clase minoritaria mediante la duplicación de los datos existentes. También se ha propuesto la generación de datos sintéticos mediante la técnica del SMOTE o IA generativa.

Adicional a las estrategias indicadas, se debe tener presente un uso adecuado del conjunto de métricas de calidad de los modelos que provee el ***machine learning***, entre las que cabe mencionar ***F1-score*** y la ***AUC-RO***C, que son más adecuadas para evaluar modelos en situaciones de clases desbalanceadas.

3.2 ¿Qué métrica considera más adecuada para este problema?

Para dar una respuesta a esta pregunta partamos por tomar en consideración la llamada matriz de confusión:

	Guessed Positive	Guessed Negative
Positive	True Positives	False Negatives
Negative	False Positives	True Negatives

en cuyas líneas se despliega la realidad mientras que en sus columnas se despliega la inferencia del modelo en discusión. El modelo perfecto, ideal, es aquel en que tanto los llamados ***falsos positivos*** (o error tipo I) como los ***falsos negativos*** (error tipo II) son cero. Pero la realidad siempre se aleja del ideal. Y peor aún, nos enfrenta a la disyuntiva del trade-off inevitable entre minimizar el error tipo I vs el error tipo II. Es decir, si se busca minimizar uno, inevitablemente se incrementará el otro.

En términos de optimización la respuesta vendría dada por la minimización del costo del error de clasificación, que puede ser expresado como función del neto entre falsos negativos y falsos positivos por sus respectivos factores de cotos:

$$\boldsymbol{CE_{Clasificación} = C_{FN} \cdot FN - C_{FP} \cdot FP}$$

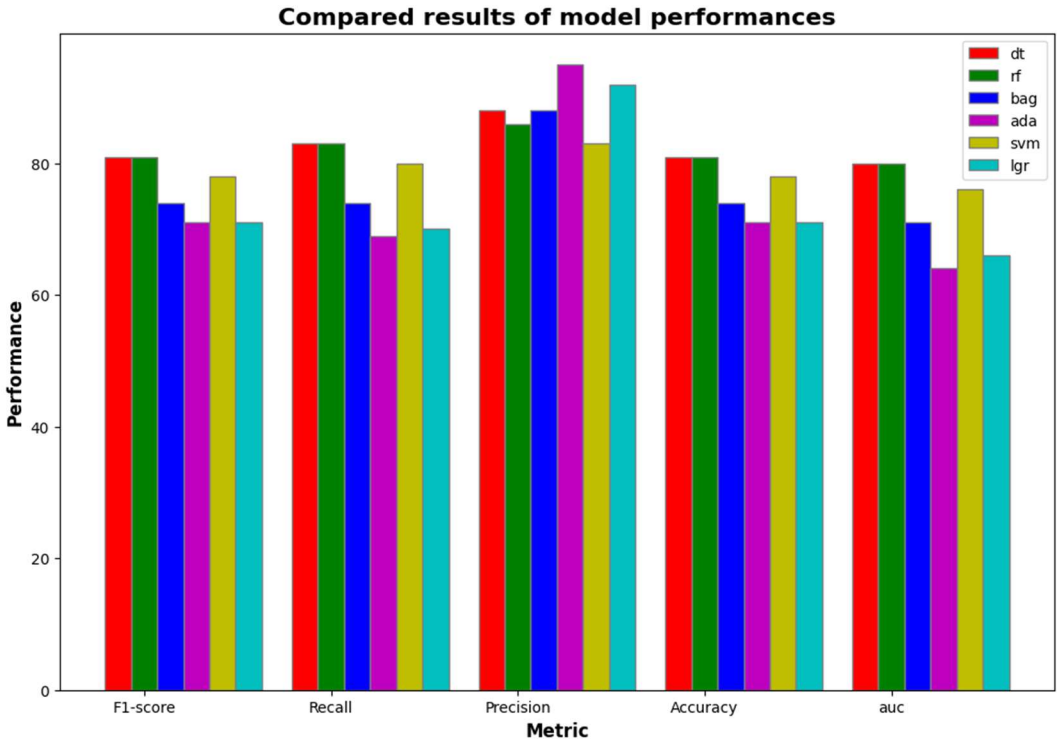
Dado el problema en análisis, se esperaría que $C_{FN} >> C_{FP}$, esto es, el costo de no alertar a tiempo sobre la anomalía de una máquina, que puede significar detenciones y costos de reparaciones no planificados, se espera sea mucho mayor al costo de una falsa alarma. Por lo cual la estrategia sería reducir el error tipo II (lo que implica un incremento del error tipo I) hasta el punto donde se cumpla que:

$$FN = FP \cdot (\frac{C_{FP}}{C_{FN}})$$

Dado lo anterior, y considerando también el desbalance (aunque leve) entre las clases, las métricas más recomendadas para la evaluación de los modelos serían ***Recall***, ***F1-score*** y ***AUC***. En efecto, el ***Recall*** permite llevar el control del error tipo II. Por su parte ***F1-score***, que es una métrica de media armónica, permite dar cuenta de error neto, esto es, del trade-off entre los errores tipo I y II. Finalmente, ***AUC***, el área bajo la curva ROC, mide en qué grado el modelo clasifica correctamente todas las instancias positivas y negativas.

3.3 ¿Qué modelo es mejor?,¿Por qué?. Apóyese en las visualizaciones ya creadas

La gráfica siguiente entrega el resumen de los resultados obtenidos por cada modelo para el conjunto de métricas analizadas.



Por su parte, la tabla siguiente muestra en detalle los valores obtenidos por modelo.

	models_name	F1-score	Recall	Precision	Accuracy	auc
0	dt	0.78	0.8	0.83	0.78	0.76
1	rf	0.81	0.83	0.88	0.81	0.8
2	bag	0.81	0.83	0.86	0.81	0.8
3	ada	0.74	0.74	0.88	0.74	0.71
4	svm	0.71	0.69	0.95	0.71	0.64
5	lgr	0.71	0.7	0.92	0.71	0.66

Así, considerando el ***Recall***, que permite llevar el control del ***error tipo II***, dos modelos son los que destacan: ***random forest*** (rf) y ***bagging boost*** (bag). A similar resultado se lleva observando el ***F1-score***, donde ambos modelos arrojan un indicador de 81%. Finalmente, el ***AUC*** da cuenta que el desempeño general de ambos modelos es bastante razonable.

Se debe recordar que, para clasificadores binarios ideales o perfectos, el ***AUC*** alcanza a 1 (o 100%). Por su parte, el azar, o el modelo que es similar a lanzar una moneda al aire para predecir la clase, alcanza al 0.5 (o 50%). En el caso de ***random forest*** y ***bagging boost***, ambos modelos están más cerca del clasificador perfecto.

3.4 ¿Qué Consideraciones extra se podrían tomar para la evaluación del proyecto?

En presencia de modelos con desempeños similares, como es el caso, se debiera preferir según establece la ***navaja de Ockham***, a aquel de mayor parsimonia, esto es, el modelo más simple. Esto también permite reducir las complejidades en la implementación del modelo final en producción, así como el seguimiento posterior del desempeño del modelo en el tiempo.

3.5 Identifique un caso de uso de su vida laboral que podría ser solucionado con herramientas de Machine Learning

a) Explique el caso de uso en detalle, identificando stakeholders, KP's de interés, impacto potencial, herramientas para el desarrollo, etc.
b) Identifique los principales desafíos y consideraciones que involucra este proyecto.

a) Solución propuesta y StakeHolders

Específicamente, se propone una solución IIoT para la continuidad operacional de correas transportadoras en minería subterránea, tomando como referencia la mina Chuquicamata. Esta solución integral permite la detección y prevención proactiva de problemas en correas transportadoras mediante el uso de cámaras de alta resolución (normal y termográficas), dispositivos Edge, sensores IIoT y machine learning. Se considera cámaras de alta resolución fijas y móviles, estas últimas montadas en robots tipo PLUTO, que asemejan un perro.

Así, los siguientes son los problemas a los cuales se busca dar solución y la herramienta de ML propuesta para ello:

Objetivo	Tipo	Técnica Propuesta	Data Utilizada
identificar desgastes y daños en Correa transportadora	Clasificación binaria	YOLO + CNN	frames de videos
desalineación de correa transportadora	Clasificación binaria	YOLO + CNN	frames de videos
suciedad y acumulación de material bajo la correa	Clasificación binaria	YOLO + CNN	frames de videos
desgaste y daños de rodillos	Clasificación binaria	YOLO + CNN	frames de videos
desgaste y daños de poleas	Clasificación binaria	YOLO + CNN	frames de videos
suciedad y acumulación de material en Rodillos	Clasificación binaria	YOLO + CNN	frames de videos
suciedad y acumulación de material en Poleas	Clasificación binaria	YOLO + CNN	frames de videos
tensión de correas	Clasificación Binaria	YOLO + CNN	frames de videos
desgaste y daños en empalmes	Clasificación Binaria	YOLO + CNN	frames de videos
acciones de riesgo por parte de operador	Clasificación binaria	YOLO + CNN	frames de videos

b) Herramientas a Utilizar

Para las tareas de modelamiento y desarrollo de los modelos de machine learning, se propone ***Python*** como lenguaje principal. Luego ***Pytorch***, que es un framework de código abierto para ***deep learning*** que se utiliza para crear redes neuronales, combinando la biblioteca de aprendizaje automático de ***Torch*** con una ***API*** de alto nivel basada en ***Python***. Su flexibilidad y facilidad de uso, entre otros beneficios, lo han convertido en el marco de aprendizaje automático líder para las comunidades académicas y de investigación.

En lo referente a visión por computadora, se propone utilizar ***OpenCV***. Esta es una librería principalmente para visión por computadora en tiempo real. Esta librería es multiplataforma y tiene licencia de software gratuito y de código abierto bajo la licencia Apache 2.

Como se explicará posteriormente, la solución de visión por computadora considera una cascada donde en primer término se aplica ***YOLO***. Esta aplicación es de código abierto y se usa para detección de objetos en tiempo real, para lo cual hace uso de una única red neuronal convolucional para detectar objetos en imágenes. Siempre requiere de un proceso de ***fine-tuning*** para satisfacer los requerimientos de cada proyecto.

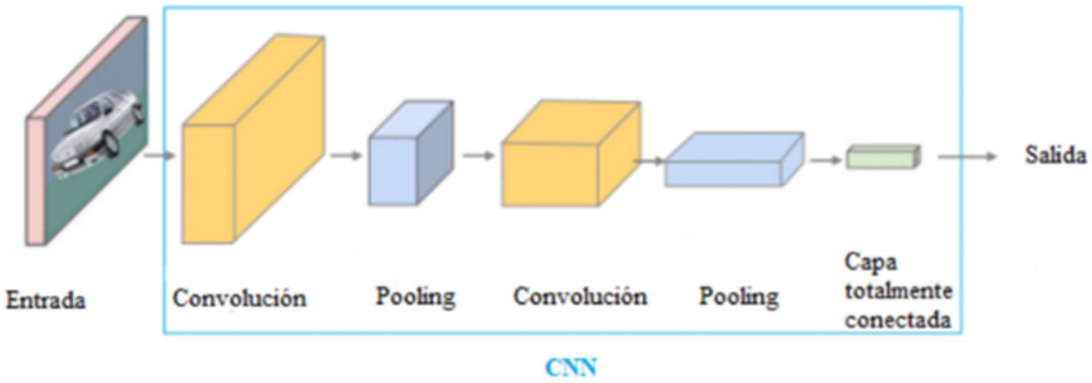
Finalmente, como librería general para ***machine learning*** se propone ***Scikit learn***. Esta es una de las librerías más utilizadas en la actualidad por las grandes empresas de tecnología. Destaca por contener clases de gran ayuda para el procesamiento, regresión, clasificación y selección de modelos.

Al menos serían 10 los modelos de clasificación basados en una casada que primero aplica ***YOLO*** para la identificación del objeto y luego una ***CNN*** para clasificación binaria. A modo de ejemplo, en el caso de los rodillos, ***YOLO*** primero lo identificaría y luego la ***CNN*** lo clasificaría como desgastado y dañado o no.

YOLO (You Only Look Once) es una familia de modelos de redes neuronales convolucionales utilizadas para la detección de objetos en tiempo real. La clave de ***YOLO*** es su velocidad y precisión, ya que es capaz de detectar objetos en una imagen con una sola pasada a través de la red. Cabe reiterar que es menester realizar un proceso de ***fine-tuning*** para customizar el modelo a los requerimientos del proyecto.

Aguas abajo, al objeto identificado se aplicaría una red convolucional para clasificación. Las redes neuronales convolucionales (CNN) son un tipo de red neuronal especialmente

diseñada para procesar datos con una estructura de cuadrícula, como las imágenes. Las ***CNN*** son muy eficaces para tareas de visión por computadora, como clasificación de imágenes, detección de objetos y segmentación de imágenes, debido a su capacidad para captar características espaciales y patrones jerárquicos en los datos. La figura siguiente entrega una arquitectura típica de una ***CNN***.



Su arquitectura considera las llamadas capas convolucionales, para extraer patrones o características de la imagen y las capas de ***pooling*** para reducir la dimensionalidad y conservar las características más importantes de la misma. Posteriormente se añaden capas que utilizan funciones de activación no lineales como ***ReLU*** (Rectified Linear Unit) que alimentan capas completamente conectadas. La última capa suele ser una capa densa con activación ***softmax*** para problemas de clasificación multiclase.

c) Desafíos a Considerar

Se describe los desafíos más importantes identificados:

- **Generación de Data**

Se requerirá un trabajo muy arduo en la generación de imágenes y en el etiquetado de ellas. En efecto, al menos 10 clases que deberían ser etiquetadas. Esto implica la revisión de imágenes por expertos para su clasificación, lo cual es un proceso laborioso.

- **Data Lake y Repositorio en la Nube**

La solución propuesta incorpora el uso intensivo de cámaras de alta resolución, operativas 24x7, lo que arroja un estimado diaria por cámara de 216 GB (con compresión). Dados los requerimientos de almacenamiento, un ***data lake*** en la nube surge como camino natural. En la Minería 4.0, la data generada debe ser considerada como otro activo. La integración de tecnologías avanzadas en las operaciones mineras genera una gran cantidad de datos que, cuando se gestionan y analizan adecuadamente, pueden proporcionar beneficios significativos a las empresas mineras. Por tal motivo,

cobra mucho sentido la alternativa de conformar un ***data lake***, como un repositorio en frío, que resida en la nube. Este es un tipo de repositorio centralizado que permite almacenar grandes volúmenes de datos en su formato original, sin necesidad de estructurarlos previamente. A diferencia de los almacenes de datos tradicionales (data warehouses), donde los datos deben ser estructurados y organizados antes de almacenarse, los ***data lakes*** permiten almacenar datos en bruto, ya sean estructurados, semiestructurados o no estructurados.

- **Red de Comunicaciones**

La red de comunicaciones es el medio por el cual se transmite la data capturada en sensores y cámaras. Para estas últimas el requerimiento de ancho de banda es significativo. Sin embargo, la tecnología actual permite resolver estos requerimientos a costos razonables.

- **Paso a Producción, Seguimiento y Mantención de los modelos**

En la nube se debe configurar un ambiente adecuado para el proceso de desarrollo, construcción y despliegue en producción de los modelos de AI que se construyan. Este ambiente se debe basar en el enfoque ***MLOps*** (Machine Learning Operations), esquema de trabajo colaborativo que busca automatizar y optimizar el ciclo de vida de los sistemas de ***ML***, posibilitando la continuidad operacional de los procesos, desde el trabajo de construcción y validación en test hasta su despliegue en producción.

Este enfoque de trabajo es vital para mejorar la eficiencia, calidad y escalabilidad en el desarrollo de los modelos. Se incluye también la tarea de monitoreo de la capacidad predictiva de los modelos a construir, en el entendido de que estos sufren un proceso natural de obsolescencia por lo que el seguimiento y monitoreo en el tiempo de sus predicciones es vital para asegurar la calidad y confiabilidad de éstos.

En el contexto de la solución propuesta el paso a producción se entiende como despliegue de modelos en ***dispositivos Edge*** que acompañarán las cámaras.

Los modelos de visual computing que vivirán en el repositorio master de la nube, deben ser desplegados localmente en los ***dispositivos*** **Edges**. Estos disponen de procesadores ***GPU*** que posibilitan computación de altas prestaciones, pudiendo analizar en tiempo real el ***stream*** de imágenes provenientes de la cámara asociada, de tal manera de generar las alarmas pertinentes.

d) Desafíos a Considerar

Desde el punto de vista de los beneficios del proyecto, se estima un costo para CODELCO por hora de detención no planificada de la correa transportadora de aproximadamente MUSD/hr 248. Y si esta detección es por accidente, la detención de ella por protocolo es de al menos 5 horas, con lo cual este costo llega a MUSD 1.240.

Según las estimaciones que se dispone del costo de inversión asociado al proyecto, con sólo un episodio de detención no planificada de la correa éste se paga casi en un 93%.