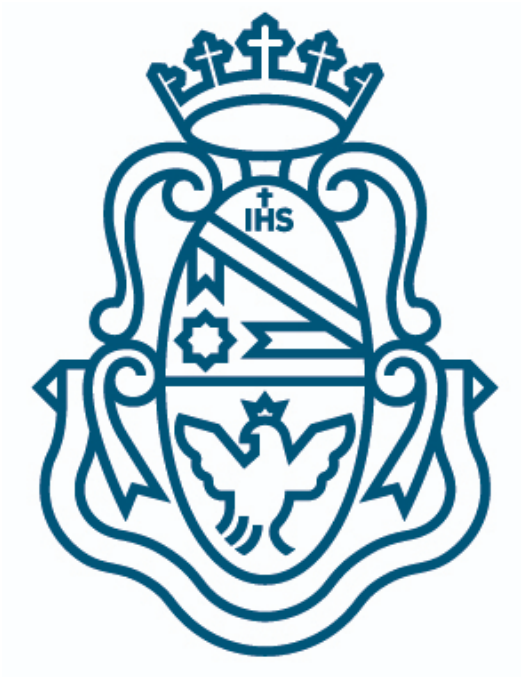
Universidad Nacional de Córdoba
Facultad de Ciencias Exactas, Físicas y Naturales



Proyecto Final Integrador

"Predicción de cantidad de defectos graves en vehículos utilitarios en planta automotriz"

Anexo

Gerardo A. COLLANTE
Matrícula: 39.022.782
Email: gerardo.collante@unc.edu.ar
Cel: 54 (03574) 650490

Supervisor
Dr. Ing. Orlando Micolini

11 de noviembre de 2021

Índice general

1 Inteligencia Artificial

> La *Inteligencia Artificial (Artificial Intelligence)* se define como el estudio de los "agentes inteligentes", i.e. cualquier dispositivo que perciba su entorno y tome medidas que maximicen sus posibilidades de lograr con éxito sus objetivos.
>
> **?**]

Esta definición nos da la idea de que la IA es un sistema reactivo, que reacciona a cambios externos y actúa en consecuencia.

1.1 Aprendizaje automático

> El *aprendizaje automático (Machine Learning)* es el estudio científico de algoritmos y modelos estadísticos que los sistemas informáticos utilizan para realizar una tarea específica sin utilizar instrucciones explícitas, sino que se basan en patrones e inferencia. Es visto como un subcampo de inteligencia artificial. Los algoritmos de aprendizaje automático crean un modelo matemático basado en datos de muestra, conocidos como "datos de entrenamiento", para hacer predicciones o decisiones sin ser programado explícitamente para realizar la tarea.
>
> **?**]

Es importante destacar la independencia del aprendizaje automático al momento de tomar decisiones a partir de los datos proporcionados sin intervención externa, es decir que no hay una especificación de reglas que dictan cómo deben ser tomadas estas decisiones. A su vez, los modelos obtenidos a partir de los algoritmos de *Machine Learning* deben tener la capacidad de predecir a partir de nuevos datos, nunca antes procesados por el modelo, a esto se lo conoce como **generalización**.

1.2 Preprocesamiento

Este punto es vital para cualquier proyecto que utiliza algoritmos de *Machine Learning*, debido a que los datos incluidos en los conjuntos conocidos como *datasets*, no suelen presentarse en condiciones para obtener el óptimo rendimiento de los algoritmos de aprendizaje. Estos datos suelen estar desbalanceados, con faltantes o ruidosos, entre otras cuestiones.

Una vez obtenido el *dataset* de entrada es primordial investigar, limpiar y transformar los datos con diversas técnicas. Una vez que los datos estén en

condiciones procedemos al entrenamiento de nuestro modelo, con el propósito de lograr un desempeño óptimo.

Para lograr este objetivo se aplican técnicas tales como normalización, reescalado, reducción de dimensionalidad, discretización, tratamiento de anomalías y *outliers*.

1.3 Aprendizaje no supervisado

En este caso no disponemos de una salida deseada, tampoco se disponen datos etiquetados o con estructuras definidas. Por lo que el objetivo principal del Aprendizaje no Supervisado es generar esas etiquetas a partir de la información que se extrae de datos proporcionados en el *dataset*, sin tener una referencia de salida.

Un ejemplo de aprendizaje no supervisado podría ser si nos encontramos con un texto extenso y queremos obtener una especie de resumen, de los temas o tópicos relevantes, probablemente de antemano no se sabe cuales son o su cantidad, por lo tanto nos enfrentamos a la situación de no conocer cuales serian las salidas esperadas del modelo. Otros ejemplos clásicos pueden ser agrupar fotografías similares o separación de diferentes fuentes que originan un determinado sonido.

Como se comentó anteriormente en la sección 1.2, en la etapa de preprocesamiento, es muy útil aplicar las técnicas del aprendizaje no supervisado ya que se cuentan con grandes cantidades de datos en contextos no conocidos y lo más importante, sin etiquetar. Entonces suele ser una buena práctica dar un primer paso mediante algoritmos de aprendizaje no supervisado antes de pasar los datos a un proceso de aprendizaje supervisado. Como por ejemplo cuando se realizan transformaciones de datos mediante reescalado o estandarización.

En las próximas subsecciones explicaremos brevemente cada una de las tareas que comprenden el Aprendizaje no Supervisado.

1.3.1 Detección de anomalías

Uno de los primeros pasos a realizar cuando se nos presenta un conjunto de datos, es proceder con la tarea llamada detección de anomalías (*anomaly detection*, AD), o identificación de *outliers* o datos fuera de rango.

Un *outlier* puede ser considerado como un dato atípico en un dataset o que parece ser inconsistente con el resto del conjunto.

Tipos de entornos en los que se produce la detección de anomalías:

- AD supervisada:
  - Las etiquetas están disponibles, tanto para casos normales como para casos anómalos.
  - En cierto modo, similar a minería de clases poco comunes o clasificación no balanceada.
- AD semi-supervisada (detección de novedades, *Novelty Detection*)
  - Durante el entrenamiento, solo tenemos datos normales.
  - El algoritmo aprende únicamente usando los datos normales.
- AD no supervisada (detección de *outliers*, *Outlier Detection*)
  - No hay etiquetas y el conjunto de entrenamiento tiene datos normales y datos anómalos.
  - Asume que los datos anómalos son poco frecuentes.
  - Algunos ejemplos típicos de detección de anomalías pueden ser, cuando se quiere detectar intrusos en tráfico de red o bien detectar acciones fraudulentas en transacciones con tarjetas de crédito.

	Ventajas	**Desventajas**
One Class SVM	• Eficiente cuando la dimensionalidad de los datos es demasiado alta.	• Es sensible ante los outliers.
Isolation Forest	• Bajo costo computacional, debido a que no usa medidas de distancia, similitud o densidad del conjunto de datos. • La complejidad crece linealmente gracias al submuestreo del dataset. • Muy útil para escalar grandes conjuntos de datos con variables irrelevantes. • Las anomalías suelen quedar en las partes altas del árbol, por lo que no es necesario construirlo completamente.	

Figura 1: Ventajas y desventajas de los algoritmos de detección de anomalías.

1.3.2 Reducción de dimensionalidad

A menudo las muestras disponibles contienen una gran variedad de características, que pueden dar como resultado un sobreajuste del modelo utilizado, por lo tanto es necesario reducir la dimensionalidad de dichos datos pero manteniendo la información relevante. Al reducir la dimensionalidad no solo se evita el sobreajuste, sino que también se obtiene una mejor visualización de los datos y se reduce el costo computacional.

Uno de los modelos más conocidos y que requieren menor costo computacional es *Principal Component Analysis* (`PCA`), pero si lo que estamos buscando es una mejor visualización de los datos y además las características no son lineales, sería recomendable usar `T-SNE`.

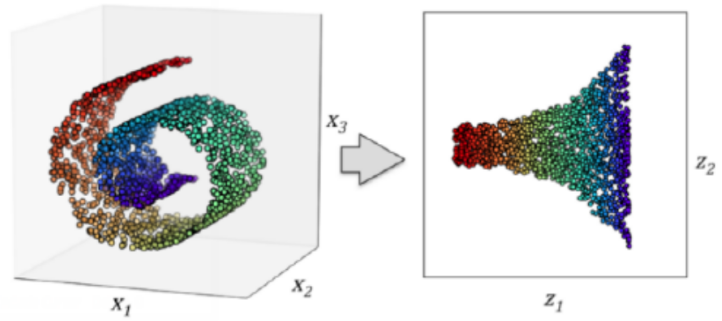


Figura 2: Reducción de dimensionalidad de 3D a 2D.

En la Fig. 2 [**?**] se muestra un ejemplo de cómo la reducción de dimensionalidad facilita la visualización de un dataset de alta dimensionalidad en una proyección de 1, 2 o 3 dimensiones.

Enumeramos en la Fig. 3 las ventajas y desventajas de los algoritmos disponibles para esta tarea.

	Ventajas	**Desventajas**
PCA	• Simple de implementar. • Es uno de los algoritmos más rápidos de reducción de dimensionalidad.	• No puede detectar características no lineales. • Los datos necesitan ser normalizados.
Isomap	• Puede detectar características no lineales.	• Lento en grandes cantidades de datos.
T-SNE	• Puede detectar características no lineales. • Recomendable cuando se quiere obtener una mejor visualización de un dataset con alta dimensionalidad.	• Computacionalmente costoto y lento.

Figura 3: Pro y contras de algoritmos de reducción de dimensionalidad.

1.3.3 *Clustering*

El *clustering* es una técnica que conceptualmente es simple de comprender, consiste en agrupar objetos con características similares. Por lo tanto, obtenemos diferentes grupos llamados clústers, donde en cada uno de ellos están contenidos los datos que son más similares entre ellos que con los que pertenecen a otros clústers, obteniendo de esta forma una útil subdivisión del *dataset*. Como no suele tenerse conocimiento sobre los datos, es decir no están etiquetados, esta técnica pertenece al Aprendizaje no Supervisado.

El algoritmo más simple de *clustering* es K-means, el cual funciona para agrupar datos que se distribuyen en formas esféricas, si se usa la distancia euclídea. y a su vez hay que proporcionar la cantidad k de grupos en los cuales queremos distribuir el *dataset*, por ello se debe tener un conocimiento previo de cuantos clúster se espera tener. Otras alternativas pueden ser, realizar *clustering* jerárquico o *clustering* basados en densidades.

En *clustering* jerárquico, vemos como resultado un dendograma, es decir un diagrama de árbol. A partir de esto, se decide un umbral de profundidad, donde se corta el árbol y de esta forma se obtiene un agrupamientos, por lo tanto a diferencia con K-means, no necesitamos tener información para poder decidir la cantidad de grupos. En la Fig. 4 podemos observar como quedan evidenciados a través del dendograma los diferentes clusters.

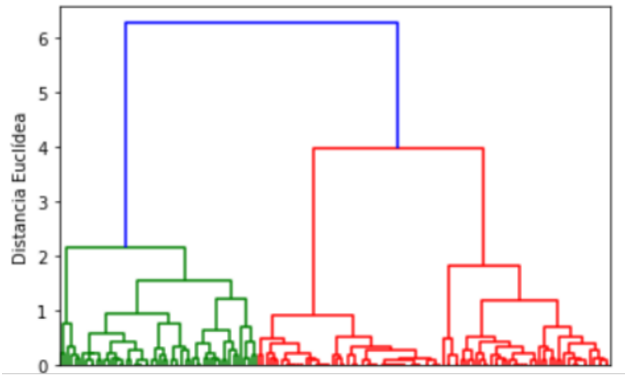


Figura 4: Dendograma generado por *clustering* jerárquico.

En cambio DBSCAN (*Density-based Spatial Clustering of Applications with Noise*), divide el *dataset* buscando las regiones densas de puntos, como podemos observar claramente en Fig. 5. Con esta técnica, tampoco especificamos el número de parámetros a priori, sino que se establecen hiperparámetros adicionales, como lo son la cantidad mínima de puntos y un radio ϵ, para lograr un óptimo funcionamiento.

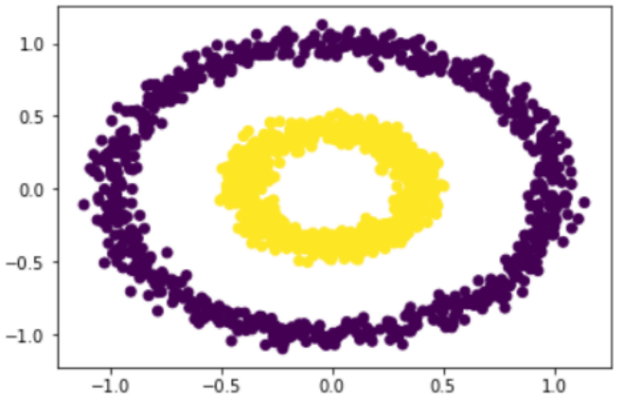


Figura 5: Clustering basado en densidades.

Enumeramos en la Fig. 6 las ventajas y desventajas de los algoritmos disponibles para esta tarea.

	Ventajas	**Desventajas**
K-Means	• Fácil de implementar. • Rápido.	• Hay que conocer el número de grupos y asumir que los datos están normalizados. • Utiliza la distancia euclídea, por lo que debemos estar seguros de que las variables estén en la misma escala. • Sensible al ruido.
Agglomerative Clustering	• No es necesario indicar el número de grupos a priori. • No es sensible a la elección de la métrica de distancia.	• No es muy eficiente.
Mini Batch K-Means	• Puede agrupar conjuntos de datos masivos. • Reduce el tiempo de cómputo gracias a los mini-batches.	• La calidad de los resultados podría verse reducida respecto a K-means.
Mean Shift	• No es necesario indicar el número de grupos a priori.	• No es escalable para muchos datos.
DBSCAN	• No hay que especificar el número de clusters a priori. • Puede detectar grupos con formas irregulares. • Puede detectar outliers. • Robusto al ruido.	• Sensible a datos con alta dimensión. • No funciona bien cuando los clusters son de densidad variable.

Figura 6: Pros y contras de los algoritmos de *clustering*.

1.4 Selección de algoritmo en base al *dataset*

En la Fig. 7 obtenemos una vista general sobre que hacer con nuestros datos en función a nuestro objetivo.

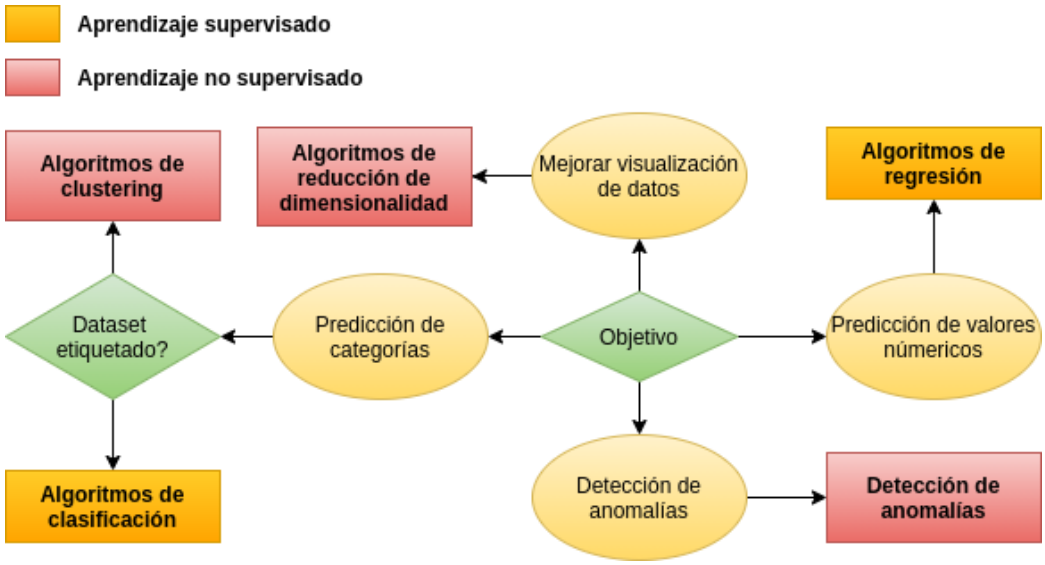


Figura 7: Diagrama general de algoritmos de aprendizaje supervisado y no supervisado.

1.4.1 Algoritmos no supervisados

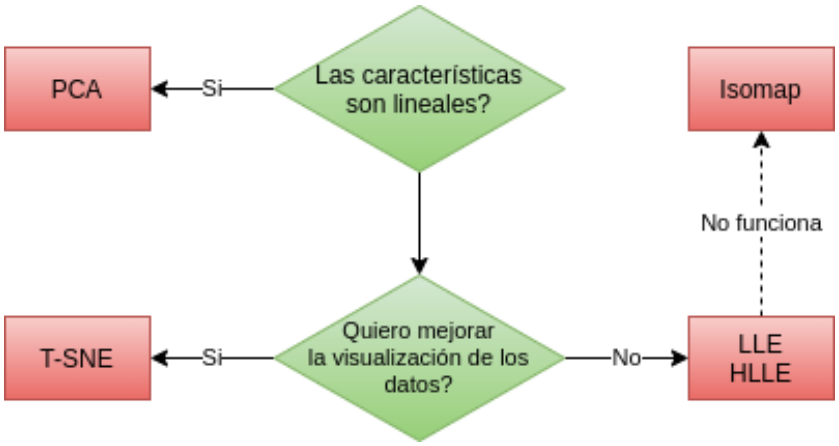


Figura 8: Diagrama general de los algoritmos de reducción de dimensión.

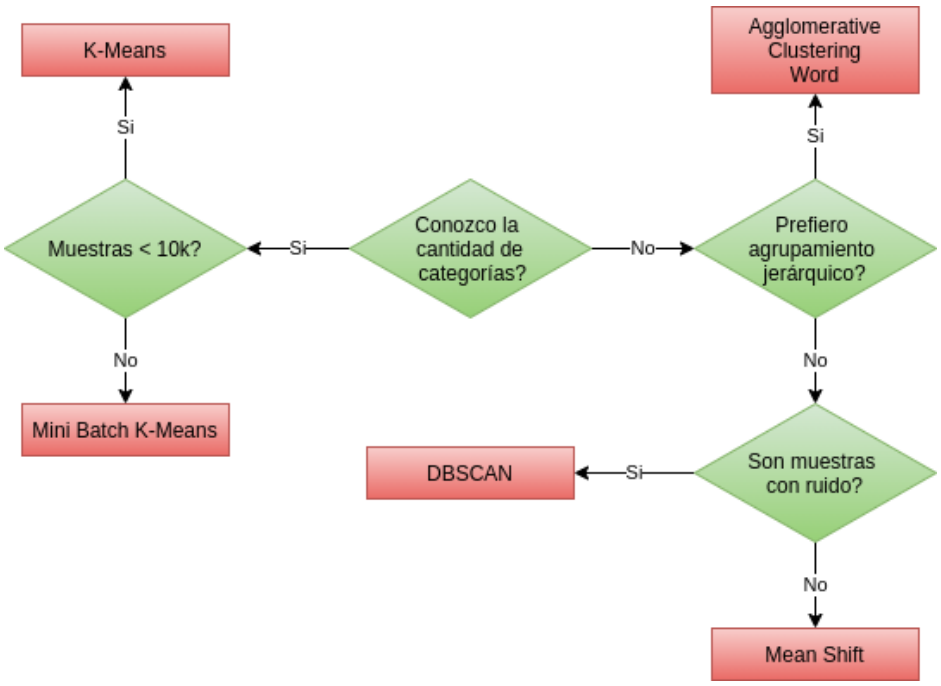


Figura 9: Diagrama general de los algoritmos de *clustering*.

Figura 10: Diagrama general de los algoritmos de detección de anomalías.

Algoritmo	Ventajas	Desventajas
Naive Bayes	- No necesita una gran cantidad de datos de entrenamiento. - Rápido.	- Supone que cada característica es independiente. - Sufre al tener características irrelevantes.
Regresión logística	- Rara vez existe sobreajuste. - Rápido de entrenar.	- Es muy difícil lograr que se ajuste a datos no lineales. -Los valores atípicos alteran la precisión del modelo.
KNN	- Eficaz en *datasets* de varias clases. - Entrenamiento rápido.	- La dimensionalidad del *dataset* merma el rendimiento. - Lento en inferencia.
Árbol de decisión *(Decision Tree)*	- Robusto en muestras con ruido. - Fácil interpretación. - Resuelve problemas no lineales.	- Cuando hay muchas etiquetas de clases, los cálculos pueden ser complejos. - Puede sufrir sobreajuste.
Clasificador de bosque aleatorio *(Random Forest Classifier)*	- No sufre sobreajuste como el árbol de decisión. - Funciona muy bien en grandes *datasets*. - Maneja automáticamente los valores faltantes.	- Consume mucho tiempo y recursos computacionales. - Difícil de interpretar. - Se debe elegir la cantidad adecuada de árboles.
Máquinas de vectores de soporte *(Support Vector Machines)*	- Eficaz en espacios de gran dimensión. - Puede manejar soluciones no lineales. - Robusto al ruido.	- Lento para entrenarse con grandes conjuntos de datos. - Ineficaz si las clases se superponen. - Se debe elegir correctamente la función de *kernel*. - Difícil de interpretar al aplicar *kernels* no lineales.

Cuadro 8: Pros y contras de cada algoritmo supervisado.

Algoritmo	Ventajas	Desventajas
Ridge	- El costo computacional no es mayor que otros algoritmos. - Permite evitar el sobreajuste.	- Se necesita una excelente selección del hiperpaametro α. - Incrementa el sesgo.
LASSO	- Evita el sobreajuste. - Selecciona características tendiendo a que sus coeficientes sean 0.	- Las características seleccionadas tienen demasiado sesgo. - Si tenemos n datos y p características, *LASSO* solo selecciona como máximo n características. - El rendimiento de la inferencia es peor que para *Ridge*.
ElasticNet	- Eficaz con muestras de gran dimensión.	- Alto costo computacional en comparación a *LASSO* o *Ridge*.
KNN Regressor	- No tiene período de entrenamiento. - Fácil interpretación. - Permite agregar datos al modelo sin inconvenientes en la precisión del algoritmo.	- La dimensionalidad del *dataset* influye mucho en el rendimiento. - Es sensible a datos ruidosos, valores faltantes y *outliers*. - En grandes *datasets* se vuelve muy alto el costo computacional para calcular distancias.
Regresor de árbol de decisión *(Decision Tree Regressor)*	- No sufre sobreajuste como el árbol de decisión. - Funciona muy bien en grandes *datasets*. - Maneja automáticamente los valores faltantes.	- Al trabajar con variables continuas, se pierde mucha información al categorizar. - Necesita variables correlacionadas. - Alto tiempo de entrenamiento. - Se puede volver demasiado complejo.
Máquinas de vectores de soporte *(Support Vector Machines)*	- Útil cuando las clases no son linealmente separables.	- Suelen ser ineficientes al momento del entrenamiento.

Cuadro 9: Pros y contras de cada algoritmo supervisado.