|  |  |
| --- | --- |
| Proyecto final  Árbol de decisión para predecir discos duros fallidos | Abstract  Detectar fallas en dispositivos de almacenamiento es crucial para proteger datos valiosos y evitar pérdidas significativas. Permite a las empresas mantener la continuidad operativa y ahorrar costos asociados con la recuperación de datos y el tiempo de inactividad. Además, facilita una planificación de mantenimiento más eficiente y reduce el estrés relacionado con la seguridad de la información. En este trabajo, presentamos innovadoras metodologías para la detección de fallas en dispositivos, empleando un modelo de Decision Tree para clasificar discos fallidos  Iñaki Sebastián Orozco García  Aprendizaje automático |

# Introducción

Detectar fallas en dispositivos de almacenamiento es crucial para proteger datos valiosos y evitar pérdidas significativas. Permite a las empresas mantener la continuidad operativa y ahorrar costos asociados con la recuperación de datos y el tiempo de inactividad.

La fuerte dependencia de los discos duros en la industria de la tecnología le da un valor muy alto a la prevención de fallas. Este trabajo de obtención de grado de la materia de aprendizaje automático se centra en asentar las bases necesarias en un proyecto de predicción de fallas datos que involucran series de tiempo, lo más importante en esta etapa del proyecto es tener un set de datos funcional que aporte información valiosa a los modelos que se van a utilizar, se utilizará un set de datos muy útil que contiene información SMART (Self-Monitoring, Analysis, and Reporting Technology) de discos duros y es el que se ha limpiado y manipulado para hacerlo funcional.

Este proyecto no solo es un proyecto exploratorio para poner en práctica los conocimientos que obtuvimos en el semestre, es la primera parte del proyecto de tesis que he estado desarrollando para la maestría y los resultados obtenidos van a aportar directamente a mi tesis final.

# Justificación, objetivos

A partir de la fuerte dependencia de la industria hacia el almacenamiento confiable se busca que para la problemática de predicción de fallas y específicamente en hardware se pueda crear un modelo que ayude con la clasificación de discos fallidos, esto con la finalidad de posteriormente usar el modelo desarrollado y mejorado en este proyecto para clasificar los datos predichos de un LSTM.

Hay un campo de estudio amplio en el que me ayudaré para desarrollar el trabajo de clasificación, en este proyecto se exploraron diferentes métodos para decidir en cuál era mejor y los atributos para los que se busca que posteriormente la LSTM genere los datos a futuro.

El objetivo final del proyecto es encontrar el mejor modelo con el mejor set de atributos del dataset de discos duros de Backblaze que sea posible comparando. Esto explorando con distintos modelos y herramientas que vimos a lo largo del curso y que el profesor recomendó.

# Adquisición de Datos y Preprocesamiento

Para la adquisición de datos se obtuvo un conjunto de datos de discos duros del dataset más usado para predicción de fallas de estos, este dataset es de una empresa llamada Backblaze que tiene servidores y cientos de discos duros, han recopilado la información de estos de manera ordenada y clasificada, y está disponible públicamente para que la gente los utilice.

Este dataset utiliza datos SMART (Self-Monitoring, Analysis, and Reporting Technology) de una gran variedad de discos duros de distintos fabricantes, cada disco duro es registrado una vez al día y registra 255 atributos SMART (dependiendo del fabricante del disco y el soporte de SMART que tenga), tanto en su versión raw como en su versión normalizada.

Además, el dataset incluye una columna denominada "failure", que tiene valores binarios referentes al fallo del disco al día siguiente, esta columna es la clave para el buen desempeño del modelo ya que nos proporciona información clasificada (ósea que buscaremos utilizar modelos supervisados para aprovechar).

Este set de datos se divide por cuatrimestres y están recopilados desde 2013, cada carpeta de cuatrimestre contiene los días pertenecientes a esos cuatrimestres en un archivo csv por día con los siguientes headers:

date,serial\_number,model,*capacity\_bytes,*failure,datacenter,cluster\_id,vault\_id,**pod\_id,**pod\_slot\_num,is\_legacy\_format,smart\_1\_normalized,smart\_1\_raw,*smart\_2\_normalized,*smart\_2\_raw,smart\_3\_normalized,smart\_3\_raw,smart\_4\_normalized,**smart\_4\_raw,**smart\_5\_normalized,smart\_5\_raw,smart\_7\_normalized,smart\_7\_raw,*smart\_8\_normalized,*smart\_8\_raw,smart\_9\_normalized,smart\_9\_raw,smart\_10\_normalized,**smart\_10\_raw,**smart\_11\_normalized,smart\_11\_raw,smart\_12\_normalized,smart\_12\_raw,*smart\_13\_normalized,*smart\_13\_raw,smart\_15\_normalized,smart\_15\_raw,smart\_16\_normalized,**smart\_16\_raw,**smart\_17\_normalized,smart\_17\_raw,smart\_18\_normalized,smart\_18\_raw,*smart\_22\_normalized,*smart\_22\_raw,smart\_23\_normalized,smart\_23\_raw,smart\_24\_normalized,**smart\_24\_raw,**smart\_27\_normalized,smart\_27\_raw,smart\_71\_normalized,smart\_71\_raw,*smart\_82\_normalized,*smart\_82\_raw,smart\_90\_normalized,smart\_90\_raw,smart\_160\_normalized,**smart\_160\_raw,**smart\_161\_normalized,smart\_161\_raw,smart\_163\_normalized,smart\_163\_raw,*smart\_164\_normalized,*smart\_164\_raw,smart\_165\_normalized,smart\_165\_raw,smart\_166\_normalized,**smart\_166\_raw,**smart\_167\_normalized,smart\_167\_raw,smart\_168\_normalized,smart\_168\_raw,*smart\_169\_normalized,*smart\_169\_raw,smart\_170\_normalized,smart\_170\_raw,smart\_171\_normalized,**smart\_171\_raw,**smart\_172\_normalized,smart\_172\_raw,smart\_173\_normalized,smart\_173\_raw,*smart\_174\_normalized,*smart\_174\_raw,smart\_175\_normalized,smart\_175\_raw,smart\_176\_normalized,**smart\_176\_raw,**smart\_177\_normalized,smart\_177\_raw,smart\_178\_normalized,smart\_178\_raw,*smart\_179\_normalized,*smart\_179\_raw,smart\_180\_normalized,smart\_180\_raw,smart\_181\_normalized,**smart\_181\_raw,**smart\_182\_normalized,smart\_182\_raw,smart\_183\_normalized,smart\_183\_raw,*smart\_184\_normalized,*smart\_184\_raw,smart\_187\_normalized,smart\_187\_raw,smart\_188\_normalized,**smart\_188\_raw,**smart\_189\_normalized,smart\_189\_raw,smart\_190\_normalized,smart\_190\_raw,*smart\_191\_normalized,*smart\_191\_raw,smart\_192\_normalized,smart\_192\_raw,smart\_193\_normalized,**smart\_193\_raw,**smart\_194\_normalized,smart\_194\_raw,smart\_195\_normalized,smart\_195\_raw,*smart\_196\_normalized,*smart\_196\_raw,smart\_197\_normalized,smart\_197\_raw,smart\_198\_normalized,**smart\_198\_raw,**smart\_199\_normalized,smart\_199\_raw,smart\_200\_normalized,smart\_200\_raw,*smart\_201\_normalized,*smart\_201\_raw,smart\_202\_normalized,smart\_202\_raw,smart\_206\_normalized,**smart\_206\_raw,**smart\_210\_normalized,smart\_210\_raw,smart\_218\_normalized,smart\_218\_raw,*smart\_220\_normalized,*smart\_220\_raw,smart\_222\_normalized,smart\_222\_raw,smart\_223\_normalized,**smart\_223\_raw,**smart\_224\_normalized,smart\_224\_raw,smart\_225\_normalized,smart\_225\_raw,*smart\_226\_normalized,*smart\_226\_raw,smart\_230\_normalized,smart\_230\_raw,smart\_231\_normalized,**smart\_231\_raw,**smart\_232\_normalized,smart\_232\_raw,smart\_233\_normalized,smart\_233\_raw,*smart\_234\_normalized,*smart\_234\_raw,smart\_235\_normalized,smart\_235\_raw,smart\_240\_normalized,**smart\_240\_raw,**smart\_241\_normalized,smart\_241\_raw,smart\_242\_normalized,smart\_242\_raw,*smart\_244\_normalized,*smart\_244\_raw,smart\_245\_normalized,smart\_245\_raw,smart\_246\_normalized,**smart\_246\_raw,**smart\_247\_normalized,smart\_247\_raw,smart\_248\_normalized,smart\_248\_raw,*smart\_250\_normalized,*smart\_250\_raw,smart\_251\_normalized,smart\_251\_raw,smart\_252\_normalized,**smart\_252\_raw,**smart\_254\_normalized,smart\_254\_raw,smart\_255\_normalized,smart\_255\_raw

Este dataset tiene un problema y es que los discos duros han ido mejorando a través de los años y por lo tanto no son muy propensos a fallar, gracias a esto hay un desbalance enorme entre los datos de discos que no fallaron al siguiente día y los que si fallaron, por eso también se decidió trabajar en este proyecto, por que es valioso saber cuándo van a fallar ya que es una tecnología muy usada en la industria, pero para el modelo no nos va a ser útil ser entrenado con muchísimos datos de discos que no fallan si lo que nosotros queremos es detectar fallas.

Primero se implementó un algoritmo de Isolation Forest para ver si podía funcionar con este dataset tan desbalanceado pero no fue muy efectivo, elaboro por qué en la sección de modelos predictivos en la página 5.

Por lo tanto lo primero que se hizo para alimentar al DT fue aplicar SMOTE, Microsof describe SMOTE en su artículo cómo: “técnica estadística para incrementar el número de casos en el dataset de manera balanceada, esto lo hace generando nuevas instancias de casos de minoría que le provees a esta técnica como entrada.”[1]

Una vez teniendo esto ya podemos elegir los discos que no fallan para tener una equivalencia de datos no tan diferente y ya podemos empezar a limpiar los datos para utilizar los atributos que serán valiosos para el modelo.

Lo que sigue es obtener un dataframe con atributos útiles, para poder cargar todos los datos en un dataframe que les sirva a los modelos se deben eliminar atributos que no aportan como fecha, serial, etc… También se reemplazan los valores NaN de las columnas Smart con 0.

Apoyándome en artículos del estado del arte, encontré el artículo de Hard Drive Failure Prediction Using Classification and Regression Trees [2] donde tienen un set de atributos del mismo dataset que estoy usando y voy a comparar estos atributos que en su artículo tuvieron buen desempeño con artículos que obtendré usando algoritmos de correlación.

Se compararon resultados de distintos algoritmos para ver la correlación entre datos, utilicé una matriz de confusión y guardé los 10 mejores atributos que tuvieran una correlación con ‘failure’ y también utilicé un algoritmo de selección de atributos univariante llamado KBest, una vez que se compararon esos 3 resultados se vio que los atributos usados en el documento del estado del arte fueron los que tuvieron mejor desempeño que los resultantes de la correlación y el KBest, entonces esos fueron elegidos como los atributos que se usarán.

Lo primero que haremos es quedarnos con los atributos más útiles que previamente vimos que eran los que mejores resultados tienen, algo interesante es que estos atributos solo tienen una décima más de precisión que los atributos de los otros métodos pero no comparten casi ningún atributo entre ellos.

Los atributos elegidos son estas columnas: 1, 3, 5, 7, 9, 187, 189, 190, 195, 197 que corresponden a los atributos SMART de:

* Raw Read Error Rate
* Spin Up Time
* Reallocated Sectors Count
* Seek Error Rate
* Power On Hours
* Reported Uncorrectable Errors
* High Fly Writes
* Temperature Celsius
* Hardware ECC Recovered
* Current Pending Sector Count
* Reallocated Sectors Count (raw value)
* Current Pending Sector Count (raw value)

Ahora si se van a cargar los 3 meses del dataset para entrenar y evaluar el DT, esto se hace utilizando chunks al agregar al dataframe que previamente fueron limpiados de las columnas que no nos interesan para no sobre cargar de basura la memoria de la computadora que posteriormente va a eliminarse de todas maneras.

Una vez que tenemos los datos limpios con los atributos que mejor desempeño tienen se van a separar estos datos para test y train, ya no se necesita que los datos estén ordenados como linea de tiempo por que lo único que necesitamos predecir es si los datos recibidos pertenecen a un disco sano o a un disco que fallará al siguiente día, se separan los datos con 20% test y 80% train

# Modelos Predictivos

Los modelos que se probaron en este proyecto fueron decisión tree e isolation forest, el primero es un modelo supervisado y el segundo no. Los árboles de decisión modelan un esquema incremental donde clasifican datos supervisados, los árboles resultantes de este modelaje dependen del tamaño de los atributos que se introducen.

El modelo de Isolation Forest en contraparte no es supervisado y no aprovecha el atributo failure que está presente en el dataset, este modelo nos puede haber sido útil por que encuentra anomalías y tiene muy buenos resultados con información muy sesgada como en nuestro caso, donde tenemos muy pocos failures y muchísimos discos que no fallaron.

También se probaron métodos de RDD (reducción de dimensionalidad) y clustering como un extra para explorar si alguno de estos métodos aportaba algo útil, los métodos usados fueron:

* t-SNE (RDD)
* LDA (RDD)
* KMeans (Clustering, no supervisado)
* DBSCAN (Clustering)

# Evaluaciones, Métricas

Lamentablemente el isolation forest no fue muy útil en este caso, es un algoritmo que detecta anomalías pero como el dataset es en carácter de tiempo y probablemente un disco tiene métricas similares un dos días antes de fallar con un día antes de fallar (cuando failure es 1) el día que falla ni siquiera se considera anomalía y por eso creo que Isolation Forest no tuvo el desempeño que esperaba.

Este fue la comparación de valores clasificados correctamente y no correctamente:

Anomaly Failure

False 0 2336618

True 0 263858

False 1 65

True 1 58

Aquí podemos ver que incorrectamente clasificó 263,858 elementos como fallos y 65 fallos los clasificó como discos funcionales.

Continuando con el DT, una vez que se separaron los datos con SOTA, se compararon los resultados de los 3 DT diferentes que fueron:

## Matriz de confusión:

|  |
| --- |
| Confusion Matrix:  [[1375   54]   [  73  207]]  Accuracy Score: 0.9256875365710943  Classification Report:                precision    recall  f1-score   support         False       0.95      0.96      0.96      1429          True       0.79      0.74      0.77       280      accuracy                           0.93      1709     macro avg       0.87      0.85      0.86      1709  weighted avg       0.92      0.93      0.92      1709 |

## Univariate Feature Selection:

|  |
| --- |
| Confusion Matrix:  [[1370   59]   [  67  213]]  Accuracy Score: 0.9262726740784084  Classification Report:                precision    recall  f1-score   support         False       0.95      0.96      0.96      1429          True       0.78      0.76      0.77       280      accuracy                           0.93      1709     macro avg       0.87      0.86      0.86      1709  weighted avg       0.93      0.93      0.93      1709 |

### Atributos del estado del arte:

|  |
| --- |
| Confusion Matrix:  [[1377   52]   [  59  221]]  Accuracy Score: 0.9350497366881217  Classification Report:                precision    recall  f1-score   support         False       0.96      0.96      0.96      1429          True       0.81      0.79      0.80       280      accuracy                           0.94      1709     macro avg       0.88      0.88      0.88      1709  weighted avg       0.93      0.94      0.93      1709 |

Donde el mejor desempeño lo tuvo el set de atributos del estado del arte, con solo 59 falsos positivos de los 280 datos.

## Clustering y RDD exploratorio

Como trabajo exploratorio como ya comenté en la sección anterior se implementaron también varios algoritmos de clustering y RDD que lamentablemente no tuvieron resultados que fueran muy útiles. Estos fueron sus resultados:

* t-SNE (RDD)

A red and blue dots

Description automatically generated

* LDA (RDD)

A graph with red and blue lines

Description automatically generated

* KMeans (Clustering, no supervisado)

A graph with red and blue lines

Description automatically generated

* DBSCAN (Clustering)

A graph with red and blue dots

Description automatically generated

# Conclusiones y Trabajo Futuro:

Por algo los artículos del campo de predicción de fallas a discos duros usan ese set de atributos, a pesar de haber intentado alternativas podemos ver que son los que mejor se desempeñan y además tienen los menos falsos positivos que es lo que nos interesan, clasificar discos fallidos.

Aún así esto solo abre el panorama para poder refinar más y dar ese elemento diferenciador a la tesis que busca aportar al campo de estudio algo útil y no solamente replicar lo ya existente, seguiré explorando y mejorando el algoritmo propuesto con estas herramientas de clase para buscar el mejor desempeño posible de este árbol de clasificación.

Posteriormente este modelo va a ser usado en conjunto con una LSTM para poder predecir datos de un disco proyectados a un tiempo en el futuro y poder clasificar estos datos proyectados con el modelo que se desarrolló para esta clase.

Estoy satisfecho con el trabajo realizado para este proyecto ya que me ha sido extremadamente útil y ha mejorado el trabajo que tenía previo para mi tesis.

# Referencias

1. Li, B. (n.d.). *Smote - Azure Machine Learning*. Azure Machine Learning | Microsoft Learn. https://learn.microsoft.com/en-us/azure/machine-learning/component-reference/smote?view=azureml-api-2
2. J. Li et al., "Hard Drive Failure Prediction Using Classification and Regression Trees," 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Atlanta, GA, USA, 2014, pp. 383-394, doi: 10.1109/DSN.2014.44. keywords: {Predictive models;Prediction algorithms;Hidden Markov models;Training;Data models;Regression tree analysis;Reliability;Hard drive failure prediction;SMART;CART;Health degree},