UNIVERSIDAD NACIONAL DE CÓRDOBA

Facultad de Ciencias Exactas, Físicas y Naturales
Ingeniería en Computación

Práctica Profesional Supervisada
Informe Final

Selección de herramientas de Machine Learning aplicado a problemas de Ingeniería

Trejo, Bruno Guillermo
guille.trejo14@gmail.com

Supervisor: Ing. Micolini, Orlando
Tutor: Ing. Luis O. Ventre
Lugar: Laboratorio de Arquitectura de Computadoras

Indice

1 Motivación

El Machine Learning (ML o Aprendizaje Automático) es una de las tecnologías más trascendentes que han resurgido en los últimos 10 años impulsada principalmente por el crecimiento exponencial de los datos producidos por las personas que hacen uso de muchas tecnologías relacionadas a internet.[1]

Hoy en día disponemos de suficiente poder de cómputo a un precio accesible para extraer patrones de conjuntos masivos de datos (Big Data) y obtener valiosa información de los mismos, lo cual nos permite solucionar muchos problemas de ingeniería utilizando en ellos estos algoritmos de aprendizaje.

A lo largo del ejercicio de su actividad, un ingeniero muchas veces se encuentra en la situación de tener una gran variedad de herramientas, métodos o algoritmos que a priori podrían solucionar el problema que se le presenta. Sin embargo, cada una de estas herramientas tienen pros y contras en su desarrollo que llevan al profesional a tener que seleccionar la que a su criterio sea la mejor según el objetivo que busque, los datos y recursos que posea.

Este escenario nos motiva y genera una necesidad de encontrar un panorama claro y a la vez amplio que le permita al lector, y a nosotros mismos, identificar las características que tiene el proyecto y de esta forma ayudarnos en el proceso de selección de un algoritmo adecuado.

2 Objetivos

El objetivo principal de nuestra Práctica Profesional Supervisada es desarrollar una guía resumida que, mediante una lectura rápida, nos permita identificar las herramientas de machine learning que mejor se adapten al proyecto a desarrollar según los recursos y datos disponibles. Además, se busca hacer una introducción conceptual sobre estos algoritmos y casos tipicos.

Puntualmente, los objetivos específicos que se pueden distinguir en el trabajo son los siguientes:

- Desarrollar una tabla (cheat sheet) para facilitar la selección del mejor algoritmo que se adecue a las necesidades de un problema específico de ingeniería.
- Presentar los conceptos básicos de las distintas herramientas que se usan en Aprendizaje Automático.
- Ejemplificar con los casos reales de aplicación más representativos de cada herramienta.
- Seleccionar la bibliografía adecuada para tener un desarrollo más profundo de los distintos conceptos.

3 Introducción

El aprendizaje automático es el proceso de extraer conocimiento de los datos automáticamente, normalmente con la meta de hacer predicciones sobre datos nuevos desconocidos. Un ejemplo clásico es el de un filtro de spam, para el que el usuario etiqueta los correos que le llegan como spam o no spam. Un algoritmo de aprendizaje automático entonces "aprende" un modelo predictivo de los datos que distingue el spam de los correos normales, un modelo que puede predecir si los nuevos correos son spam o no.

Uno de los elementos centrales del aprendizaje automático es el concepto de **toma de decisiones automáticas** a partir de los datos **sin que el usuario especifique reglas** acerca de cómo esas decisiones deben ser tomadas. Para el caso de los correos, el usuario no proporciona una lista de las palabras o características de un correo spam. En lugar de eso, el usuario proporciona ejemplos de correos que son y no son spam. El segundo concepto central es el de la **generalización**. La meta de un modelo de aprendizaje automático es predecir en nuevos ejemplos nunca antes vistos. [2] [3] [4]

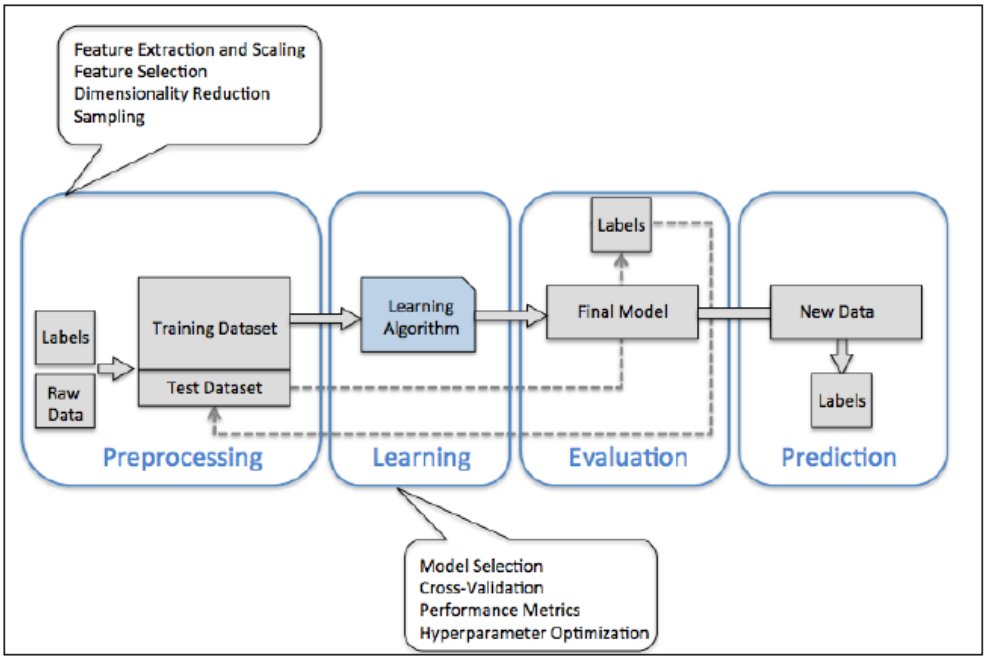


Fig. 1: Diagrama de flujo completo de una aplicación de Machine Learning.

Vale aclarar que el pre-procesamiento de los datos es uno de los pasos más cruciales antes de aplicar cualquier modelo. Raramente los datos vienen en el formato necesario para el rendimiento óptimo de un algoritmo de aprendizaje, por lo que se deben realizar una serie de transformaciones que generalmente se logran mediante técnicas de normalización, reescalado, etc. y a veces incluso se utilizan algoritmos de aprendizaje no supervisado.

4 Aprendizaje supervisado

En **aprendizaje supervisado**, tenemos un dataset que contiene tanto características de entrada como una salida deseada (etiqueta, target o label). El término **supervisado** se refiere a que se trabaja con un conjunto de muestras cuyas etiquetas son conocidas previamente.

El objetivo principal es entrenar un modelo con datos de entrenamiento etiquetados para poder realizar predicciones cuando se presenten datos nuevos (nunca antes vistos).

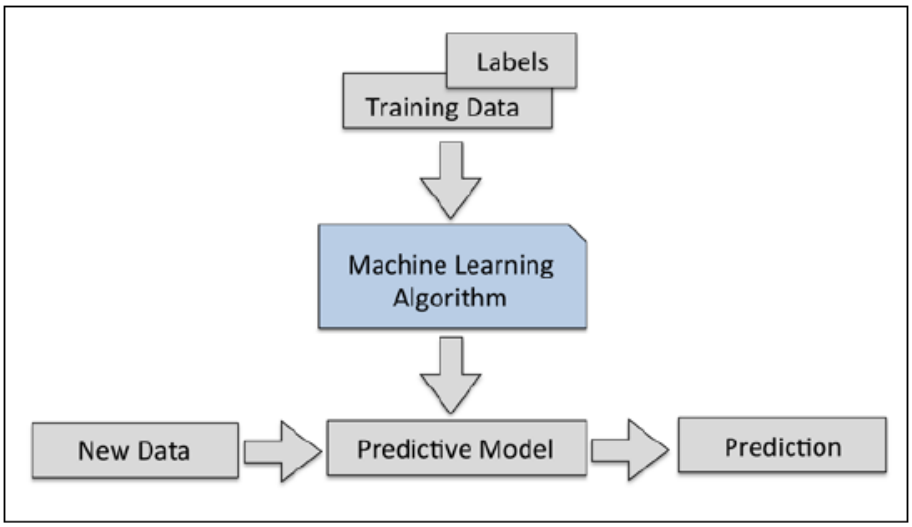


Fig. 2: Flujo simplificado de información de un sistema de Machine Learning.

Algunos ejemplos son:

- Dada una imagen RBG de un telescopio, determinar si el objeto es una estrella, un cuásar o una galaxia.
- Dada la fotografía de una persona, identificar a la persona en la foto.
- Dada una lista de películas que una persona ha visto y la puntuación que le he dado, recomendar una lista de películas que le podrían gustar.
- Dada la edad de una persona, su educación y su posición, deducir su salario.

Lo que tienen en común todas estas tareas es que tenemos una o más cantidades no determinadas sobre un objeto que deben ser predichas usando otras cantidades.

El aprendizaje supervisado se divide a su vez en dos categorías, **clasificación** y **regresión**.

4.1 Clasificación

En clasificación, la etiqueta es discreta, por ejemplo *spam* y *no spam*. En otras palabras, se proporciona una distinción clara entre las categorías. Es más, es importante indicar que estas categorías son nominales y no ordinales. Las variables nominales y ordinales son ambas subcategorías de las variables categóricas. Las variables ordinales tienen asociado un orden, por ejemplo, las tallas de las camisetas "XL >L >M >S". Por el contrario, las variables nominales no implican un orden, por ejemplo, no podemos asumir (en general) "naranja >azul >verde".

El ejemplo anterior de detección de spam representa un ejemplo típico de una tarea de *clasificación binaria*, donde el algoritmo aprende una serie de reglas con el objetivo de distinguir entre dos clases posibles: spam y no spam. Un ejemplo de una *clasificación multi-clase* es el reconocimiento de caracteres manuscritos. Es importante mencionar que si una clase no aparece en el dataset de entrenamiento, el modelo no será capaz de predecir nunca esa salida (por ejemplo no se podrían reconocer dígitos del 0 al 9 si al entrenar el modelo no había ningún ejemplo etiquetado de esos números).

La *Figura 3* ilustra el concepto de una tarea de clasificación binaria dadas 30 muestras de entrenamiento: 15 muestras son etiquetadas como *clase negativa* (círculos) y las otras 15 como *clase positiva* (signos más). En este escenario, nuestro dataset es bi-dimensional, lo cual significa que cada muestra tiene 2 valores asociados: x_1 y x_2. Luego, podemos usar un algoritmo de aprendizaje supervisado para aprender una regla (el umbral de decisión representado como una línea punteada) que pueda separar esas 2 clases y clasificar nuevos datos en cada una de esas 2 categorías dados sus valores x_1 y x_2.

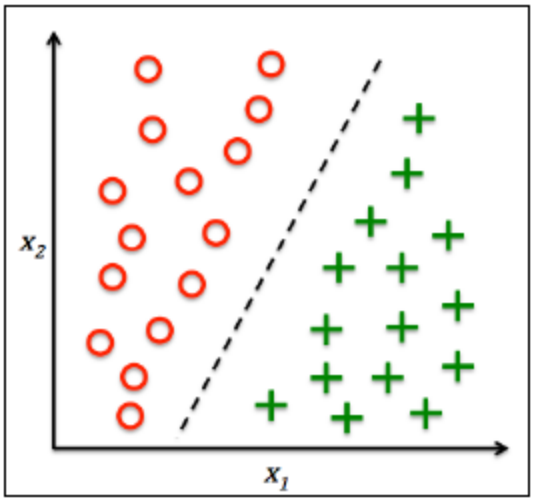


Fig. 3: Clasificación binaria.

4.2 Regresión

En regresión, la etiqueta es continua, es decir una salida real. Por ejemplo, en astronomía, la tarea de determinar si un objeto es una estrella, una galaxia o un cuásar es un problema de clasificación: la etiqueta viene de tres categorías distintas. Por otro lado, podríamos querer estimar la edad de un objeto basándonos en su imagen: esto sería regresión, porque la etiqueta (edad) es una cantidad continua.

Generalmente, en el análisis de regresión, tenemos variables predictoras (explicativas) y una variable continua de respuesta (salida), y tratamos de encontrar una relación (por ejemplo una función lineal) entre estas variables que nos permita predecir una salida.

La *Figura 4* ilustra el concepto de *regresión lineal*. Dada una variable predictora **x** y una variable de respuesta **y**, ajustamos una línea recta que minimice la distancia (generalmente la distancia euclídea) entre los puntos de muestra y la misma recta. Ahora podemos usar la pendiente y la ordenada al orígen (coeficientes) aprendidas de estos datos para predecir la variable de salida de nuevos datos.

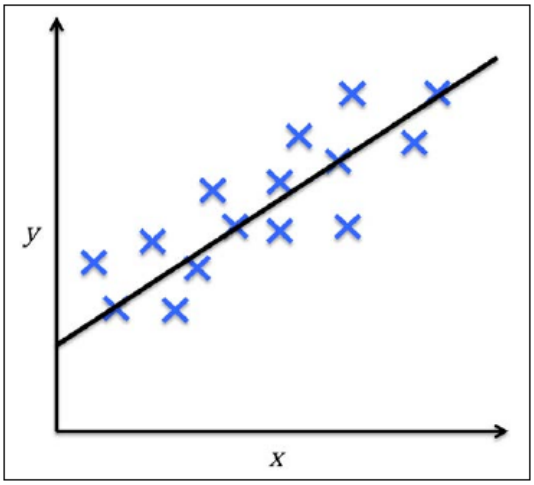


Fig. 4: Ajuste de un modelo de regresión lineal.

5 Aprendizaje no supervisado

En aprendizaje supervisado sabemos la *respuesta correcta* de antemano cuando entrenamos nuestro modelo, en cambio en *aprendizaje no supervisado* se tienen datos sin etiquetar o de estructuras desconocidas. Usando estos modelos, podemos explorar la estructura de los datos para extraer información significativa sin tener la ayuda de una variable de salida. En cierto sentido, se podría pensar que el aprendizaje no supervisado es una forma de descubrir etiquetas a partir de los propios datos. Este tipo de aprendizaje suele ser más difícil de entender y evaluar.

El aprendizaje no supervisado engloba tareas tales como *clustering, reducción de la dimensionalidad y detección de anomalías*. Algunos otros ejemplos son:

- Dado un conjunto de observaciones detalladas sobre galaxias lejanas, determinar qué combinación o combinaciones de características resumen mejor la información.
- Dada un sonido que proviene de la mezcla de dos fuentes distintas de sonido (por ejemplo, una persona que habla por encima de una canción), separar ambas fuentes de sonido (a esto se le llama blind source separation).
- Dado un vídeo, aislar un objeto móvil y categorizarlo en función de otros objetos móviles que hayan sido vistos anteriormente.
- Dada una larga colección de artículos de noticias, encontrar temas recurrentes en estos artículos.
- Dada una colección de imágenes, agrupar las imágenes que son más similares (por ejemplo, para visualizar una colección de imágenes).

Algunas veces los dos tipos de aprendizaje se combinan: por ejemplo, el aprendizaje no supervisado puede ser utilizado para encontrar características útiles en datos heterogéneos y luego usar estas características en un *framework* de aprendizaje supervisado.

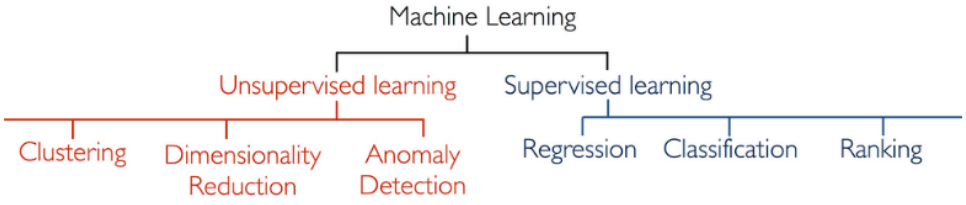


Fig. 5: Taxonomía simplificada de aprendizaje automático.

Muchas formas de aprendizaje no supervisado encuentran una nueva representación de los datos de entrada sin ninguna variable adicional, lo que se suele denominar como **transformación** de los datos (preprocesamiento que se realiza a los datos para poder operar con ellos de una manera más sencilla). Ejemplos de *transformadores* son el reescalado y la estandarización (transformar los datos para que tengan una media de 0 y una varianza unitaria).

5.1 Clustering

El *clustering* es una técnica de análisis de exploración de datos que nos permite dividir un dataset en subgrupos (clusters) significativos sin tener ningún conocimiento previo acerca de la pertenencia de los datos a alguno de estos. Cada uno de los clusters que pueda surgir durante el análisis define un grupo de objetos que comparten cierto grado de similitud pero que se diferencian en un grado mayor de los objetos en los otros clusters (por esta razón a veces el clustering es denominado *clasificación no-supervisada*). El clustering es una gran técnica para estructurar información y derivar relaciones significativas entre los datos (por ej. descubrir grupos de clientes a partir de sus intereses para desarrollar distintos programas de marketing).

La *Figura 6* ilustra como el clustering puede ser aplicado para organizar datos sin etiquetar en 3 grupos distintos basándose en la similitud de sus características x_1 y x_2.

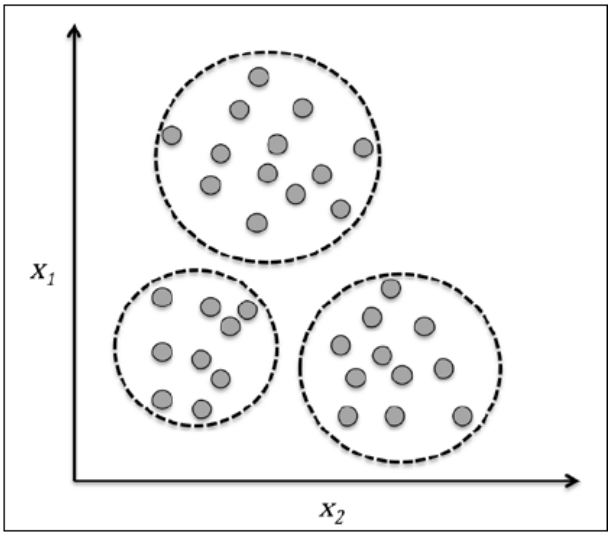


Fig. 6: Clustering de un dataset de prueba.

El agrupamiento siempre viene con suposiciones: un algoritmo de agrupamiento encuentra grupos haciendo suposiciones acerca de cómo debería agruparse los ejemplos. Esta técnica a menudo puede ser vista a través de una mirada probabilística o de optimización. Esto quiere decir que a la llegada de una nueva entrada existe la probabilidad de que integre uno de estos clusters.

Cada algoritmo hace suposiciones distintas y la calidad e interpretabilidad de nuestros resultados dependerá de si estas suposiciones son correctas para nuestro objetivo. Por ejemplo, una suposición que hace el modelo *K-means* es que todos los grupos tienen la misma varianza esférica (si se usa la distancia euclídea).

5.2 Reducción de dimensionalidad

Otro campo del aprendizaje no supervisado es la *reducción de dimensionalidad*. Con frecuencia se trabaja con datos de una alta dimensionalidad (o sea, cada instancia de un dato viene con un gran número de características), lo cual puede ser un obstáculo si tenemos un espacio de almacenamiento limitado así como una potencia computacional no muy alta.

La reducción de dimensionalidad no-supervisada es un enfoque común para pre-procesamiento de características, permitiendo remover el ruido de los datos (lo cual puede también degradar el rendimiento predictivo de ciertos algoritmos) y comprimirlos en un subespacio dimensional más pequeño sin perder la información relevante. A veces, la reducción de dimensionalidad puede ser útil para visualizar un dataset de alta dimensionalidad en una proyección de 1, 2 o 3 dimensiones. Ejemplo en la *Figura 7*.

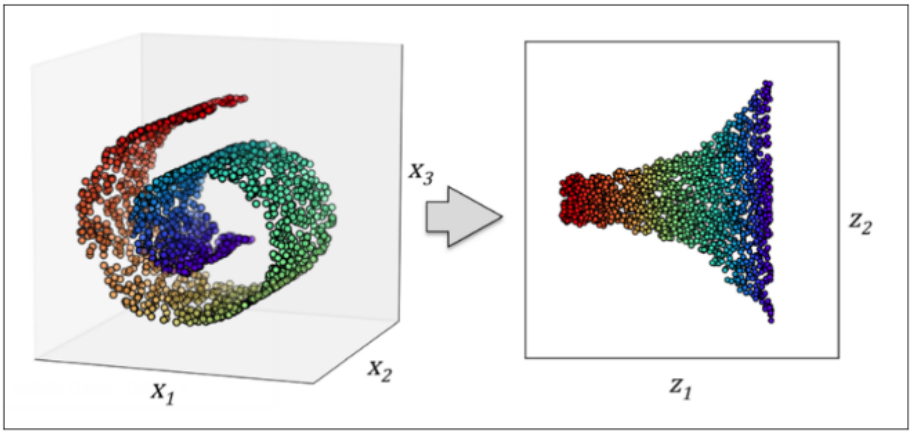


Fig. 7: Reducción de dimensionalidad de 3D a 2D.

5.2.1 Análisis de Componentes Principales (PCA)

Una técnica muy usada para reducir la dimensionalidad es el Análisis de Componentes Principales (*Principal Component Analysis*, PCA), la cual crea una proyección lineal. Es decir, encontramos características nuevas para representar los datos que son una combinación lineal de los datos originales (lo cual es equivalente a rotar los datos). Podemos pensar en el PCA como una proyección de nuestros datos en un nuevo espacio de características, la forma en que logra esto es buscando la direcciones de máxima varianza (*Figura 8*), capturando la mayor parte de la información.

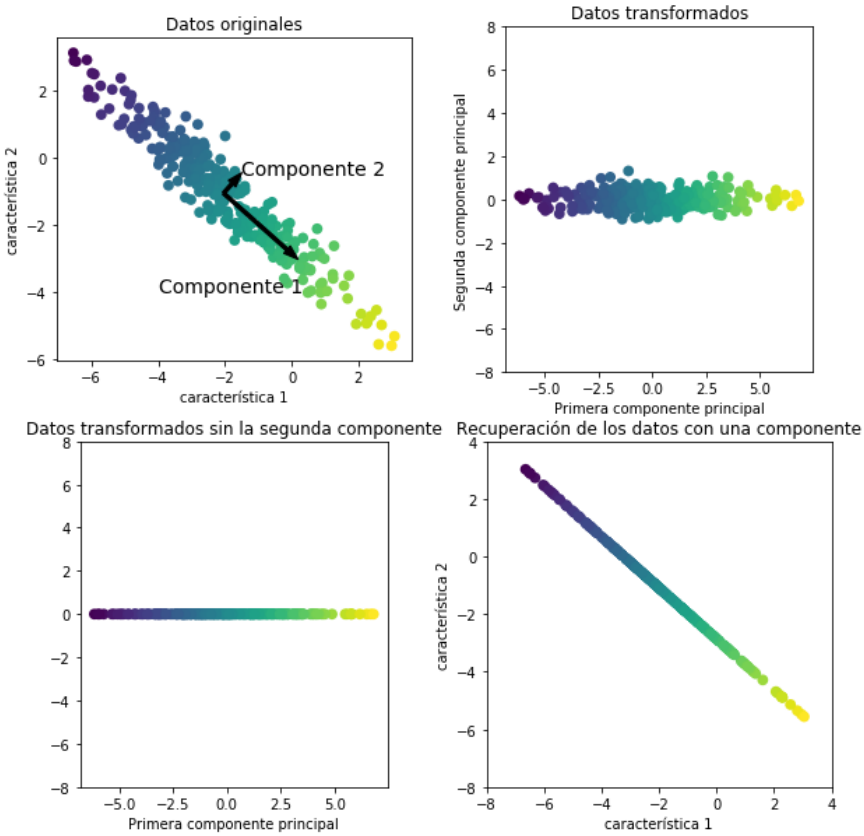


Fig. 8: Proyección sobre componentes principales.

5.3 Detección de anomalías

La detección de anomalías (*anomaly detection*, AD) es una tarea de aprendizaje automático que consiste en detectar *outliers* o datos fuera de rango.

Un outlier es una observación en un dataset que parece ser inconsistente con el resto del conjunto. Johnson 1992

Un outlier es una observación que se desvía tanto de las demás que levanta sospechas de que fue generada por un mecanismo diferente. Outlier/Anomaly Hawkins 1980

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.
  - Asunción: los datos anómalos son poco frecuentes.

Algunas aplicaciones son:

- Detección de intrusos (identificar patrones extraños en el tráfico de red que podrían ser señal de un hackeo).
- Sistema de monitoreo de salud (identificar un tumor maligno en una resonancia magnética).
- Detección de fraudes en transacciones con tarjetas de crédito.
- Detección de fallas en entornos operativos.

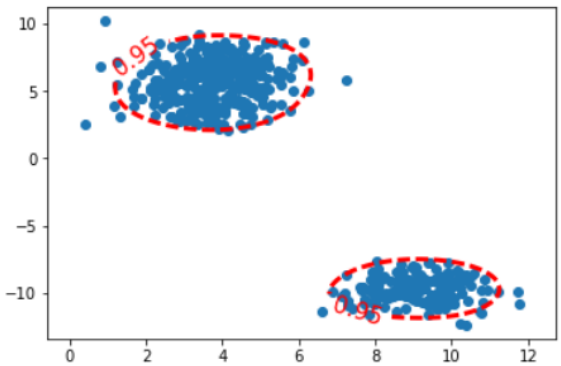


Fig. 9: Detección de outliers (puntos rojos) en dos clusters.

6 Redes Neuronales

Una red neuronal es un sistema de computación compuesto por un gran número de elementos simples altamente interconectados, los cuales procesan información por medio de su estado dinámico como respuesta a entradas externas. Los datos ingresan por medio de la "capa de entrada", pasan a través de la "capa oculta" y salen por la "capa de salida". Cabe mencionar que la capa oculta puede estar constituida por varias capas *(Figura 10).* [5]

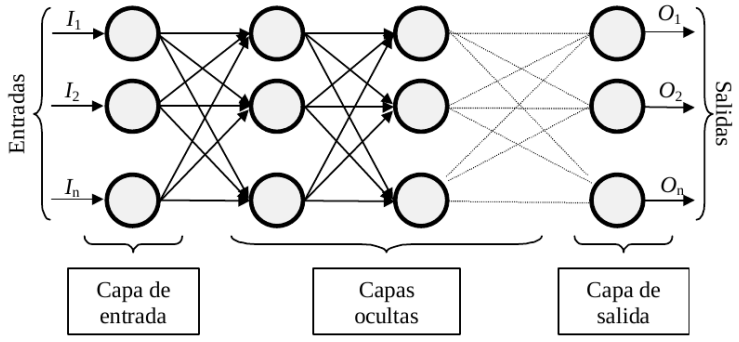


Fig. 10: Ejemplo de una red neuronal totalmente conectada.

Las redes neuronales se entrenan de una forma similar a los demás algoritmos de aprendizaje automático, brindando un dataset con ciertas características de entrada y las etiquetas correspondientes (salidas). La optimización de los pesos internos en las capas ocultas se realiza con distintos algoritmos de "reducción de costo" (por ej. *gradient descent*).

Función de entrada (input function)

La neurona trata a muchos valores de entrada como si fueran uno solo; esto recibe el nombre de entrada global. Por lo tanto, ahora nos enfrentamos al problema de cómo se pueden combinar estas simples entradas (I_1, I_2,...,I_n) dentro de la entrada global. Esto se logra a través de la función de entrada, la cual se calcula a partir del vector entrada. La función de entrada puede describirse como sigue:

$$\text{input} = (I_1 \text{ x } w_1) * (I_2 \text{ x } w_2) *...* (I_n \text{ x } w_n)$$

donde: * representa al operador apropiado (por ejemplo: máximo, sumatoria, productoria, etc.), n al número de entradas a la neurona N_i y w_i al peso.

Función de activación (activation function)

La función activación calcula el estado de actividad de una neurona; transformando la entrada global en un valor (estado) de activación, cuyo rango normalmente va de (0 a 1) o de (–1 a 1). Esto es así porque una neurona puede estar totalmente inactiva (0 o –1) o activa (1). La función activación, es una función de la entrada global, menos un umbral. Las funciones de activación más comúnmente utilizadas son: función lineal, función sigmoid, función tangente hiperbólica.

Función de salida (output function)

El último componente que una neurona necesita es la función de salida. El valor resultante de esta función es la salida de la neurona i; por ende, la función de salida determina que valor se transfiere a las neuronas vinculadas. Si la función de activación está por debajo de un umbral determinado, ninguna salida se pasa a la neurona subsiguiente. Normalmente, no cualquier valor es permitido como una entrada para una neurona, por lo tanto, los valores de salida están comprendidos en el rango [0, 1] o [-1, 1]. También pueden ser binarios 0, 1 o -1, 1. Dos de las funciones de salida más comunes son: ninguna (la salida es la misma que la entrada, también llamada función identidad) y binaria (1 si la función de activación es mayor que el umbral y 0 en caso contrario).

7 Redes Neuronales Convolucionales

Las redes neuronales convolucionales (CNNs, en inglés) se aplican principalmente para desarrollar modelos supervisados y no supervisados cuando los datos de entrada son imágenes. En general, las convoluciones bi-dimensionales (2D) son aplicadas a las imagenes, mientras que las convoluciones uni-dimensionales (1D) pueden ser usadas en una entrada secuencial para capturar dependencias temporales. Este enfoque generalmente se utiliza para desarrollar modelos de pronósticos en una serie temporal. [6] [7]

Convolución 2D

Las CNNs aprovechan la estructura 2D de las imágenes, las cuales tienen una dimensión rectangular. El valor de color de cada pixel, sería una característica de entrada al modelo. Un problema que se presenta es la potencia computacional requerida para entrenar los pesos de las capas ocultas cuando hay muchas muestras (ej. dataset de caracteres manuscritos tienen dígitos de 32x32, lo que en una red con 100 neuronas de entrada serian 307200 pesos a calcular en cada capa). Para solucionar esto, se divide la imagen en partes y cada neurona se conecta solo a una parte (local), limitando el costo computacional.

Convolución 1D

Las capas de convolución 1D pueden ser usadas para desarrollar modelos de pronósticos temporales. Una serie de tiempo que tiene 1 x m observaciones es como una imagen de dimensión p con una altura de un solo píxel. En este caso, la convolución 1D puede ser aplicada como un caso especial de una convolución 2D.

En algunos casos de estudio, como el del pronóstico meteorológico, es útil combinar tanto la convolución 2D como la 1D para poder ver la evolución de las imágenes captadas a lo largo del tiempo y poder hacer predicciones a futuro.

8 Antecedentes

En publicaciones científicas no se encuentran muchos casos de trabajos que se hayan realizado con el objetivo que nos proponemos (confeccionar una tabla/guía para seleccionar el algoritmo adecuado), no así en distintos blogs pertenecientes a "Data Scientists" los cuales han realizado contribuciones Open Source (librerías o algoritmos específicos) y han incluido en su trabajo un diagrama de flujo que sirve de ayuda para elegir qué herramienta usar para cada proyecto.

En una búsqueda de Google Académico, se encuentra el siguiente libro:

Machine Learning for Dummies - J.P Mueller [8], Luca Massaron [9]

En el cual se brinda como anexo una "Cheat Sheet", que es una guía para facilitar el acceso a las necesidades más comunes a la hora de encarar un proyecto con Machine Learning (para qué es mejor cada algoritmo, ventajas y desventajas de cada uno, librerías que lo implementan, etc.). A continuación se adjunta el enlace a dicha tabla:

https://www.dummies.com/programming/big-data/data-science/machine-learning-dummies-cheat-sheet/ [10]

En cuanto a las publicaciones en blogs de los creadores de librerías específicas (por ej. SciKit Learn para Python), nos encontramos con:

Dr. Andreas Christian Müller [11]:

http://peekaboo-vision.blogspot.com/2013/01/machine-learning-cheat-sheet-for-scikit.html [12]

El cual se basa en las herramientas de la librería sklearn, por lo que no incluye todos los algoritmos (por ej. las redes neuronales). Un enfoque interesante que plantea en la publicación es el siguiente:

"Basically, start simple first. If this doesn't work out, try something more complicated. The chart above includes the intersection of all algorithms that are in scikit-learn and the ones that I find most useful in practice. Only that I ***always*** *start out with "just looking". To make any of the algorithms actually work, you need to do the* ***right preprocessing*** *of your data - which is much more of an art than picking the right algorithm imho."*

"Básicamente, primero empezar simple. Si esto no funciona, probar algo más complicado. El diagrama incluye la intersección de todos los algoritmos que se encuentran en scikit-learn y aquellos que me parecen más útiles en la práctica. ***Siempre*** *comienzo "solo mirando"* (hace referencia a un punto de la tabla que lleva a reducción de dimensionalidad para visualización del dataset). *Para hacer que los algoritmos realmente funcionen, necesitas realizar el correcto* ***preprocesamiento*** *de los datos, lo cual es mucho más difícil que elegir el algoritmo correcto, en mi humilde opinión."*

Dr. Hui Li [13]:

The SAS Data Science Blog [14]

https://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/

Este caso es muy similar al anterior, solo que se hace un desarrollo más intenso de cada uno de los tipos de aprendizaje y sus algoritmos. En palabras de la autora, se propone lo siguiente:

"This is the work flow which is easy to follow. The takeaway messages when trying to solve a new problem are:

- *Define the problem. What problems do you want to solve?*
- *Start simple. Be familiar with the data and the baseline results.*
- *Then try something more complicated."*

"Este es el diagrama de trabajo más fácil de seguir. Los mensajes clave cuando se intenta resolver un nuevo problema son:

- *Definir el problema. ¿Qué problemas quieres resolver?*
- *Comenzar simple. Familiarizarse con los datos y los resultados de base.*
- *Luego, intentar algo más complicado."*

9 Cheat Sheet

Basado en los antecedentes antes nombrados, se construyó la siguiente cheat sheet. Es un diagrama que flujo que guía al lector a la herramienta mas recomendada para realizar su trabajo. Tiene su inicio en la pagina siguiente y se descompone en las continuas, una por cada clase. En cada bloque se nombran distintas librerías que implementan tales modelos, ademas se referencia tanto un ranking [15] (*Figura 11*) como una planilla [16] que ayudan a decidir entre cual usar según conveniencias. El apartado STRUCTURE no esta incluido en este trabajo pero alienta a una continuidad del mismo, consultando las librerías **SVM-Struct** o **pytorch** (de los desarrolladores de scikit-learn).

En cuanto al apartado ANOMALY DETECTION, se debe tener en cuenta que este tipo de aprendizaje se utiliza en distintos entornos: supervisado, semi-supervisado y no supervisado. El caso específico de no-supervisado es en el que más nos enfocamos, ya que sirve para *detección de outliers*, en donde no hay etiquetas y el conjunto de entrenamiento tiene datos normales y datos anómalos, y además se asume que los datos anómalos son poco frecuentes. No se incluye un diagrama para este algoritmo debido a que no es necesario tomar tantas decisiones para seleccionar un modelo específico, sino que generalmente se usan los siguientes: **KernelDensity(kernel='gaussian')**, **OneClassSVM** e **Isolation Forest**.

En la salida de los bloques no supervisados (verdes) el flujo retorna porque generalmente se utilizan estos algoritmos como pasos previos de pre-procesamiento del dataset para un posterior uso de los algoritmos de aprendizaje supervisado; sin embargo, también pueden ser considerados como puntos finales de un flujo de trabajo.

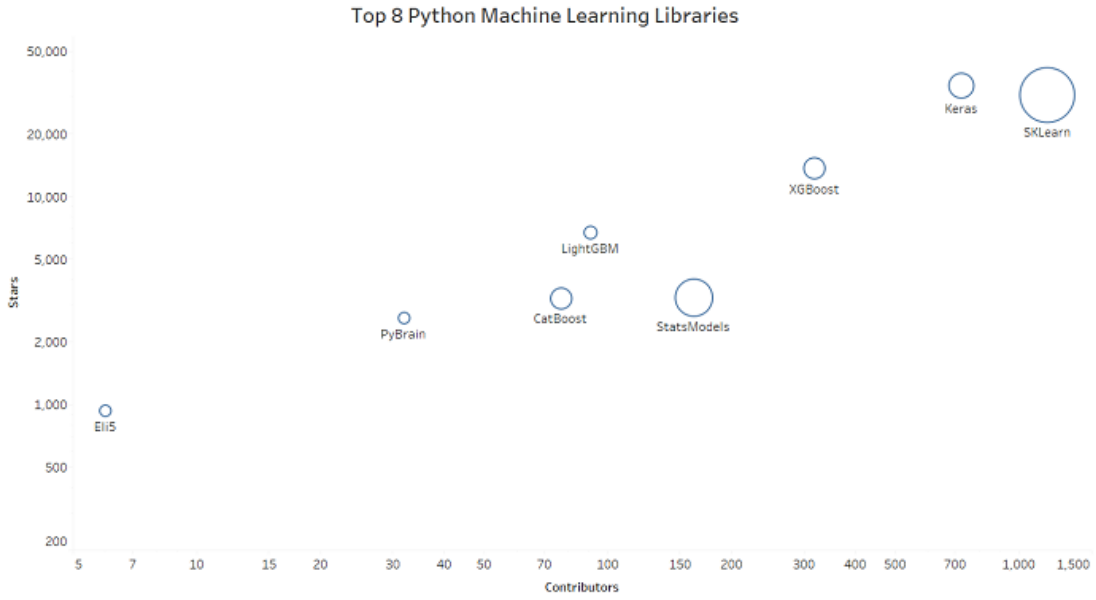
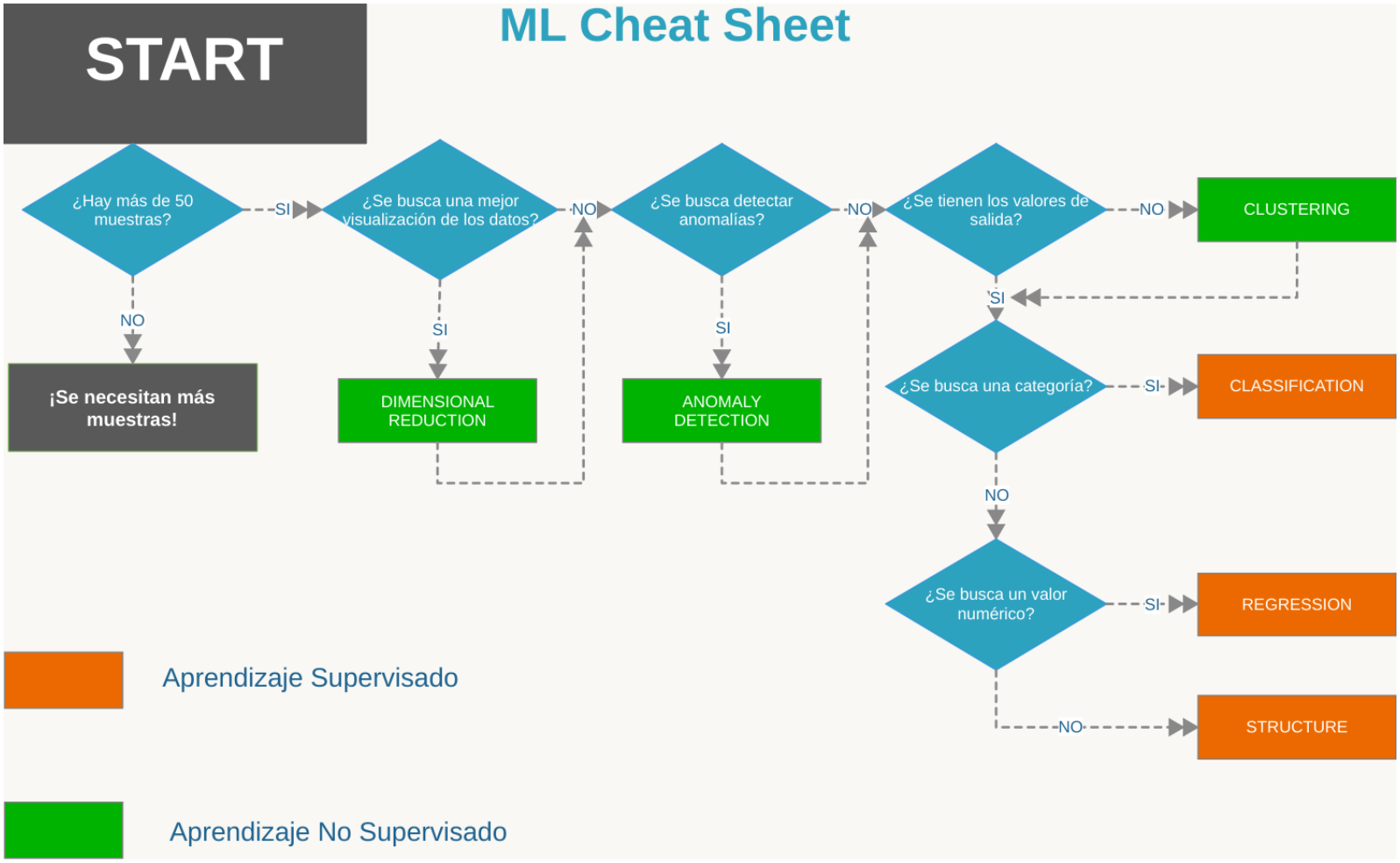
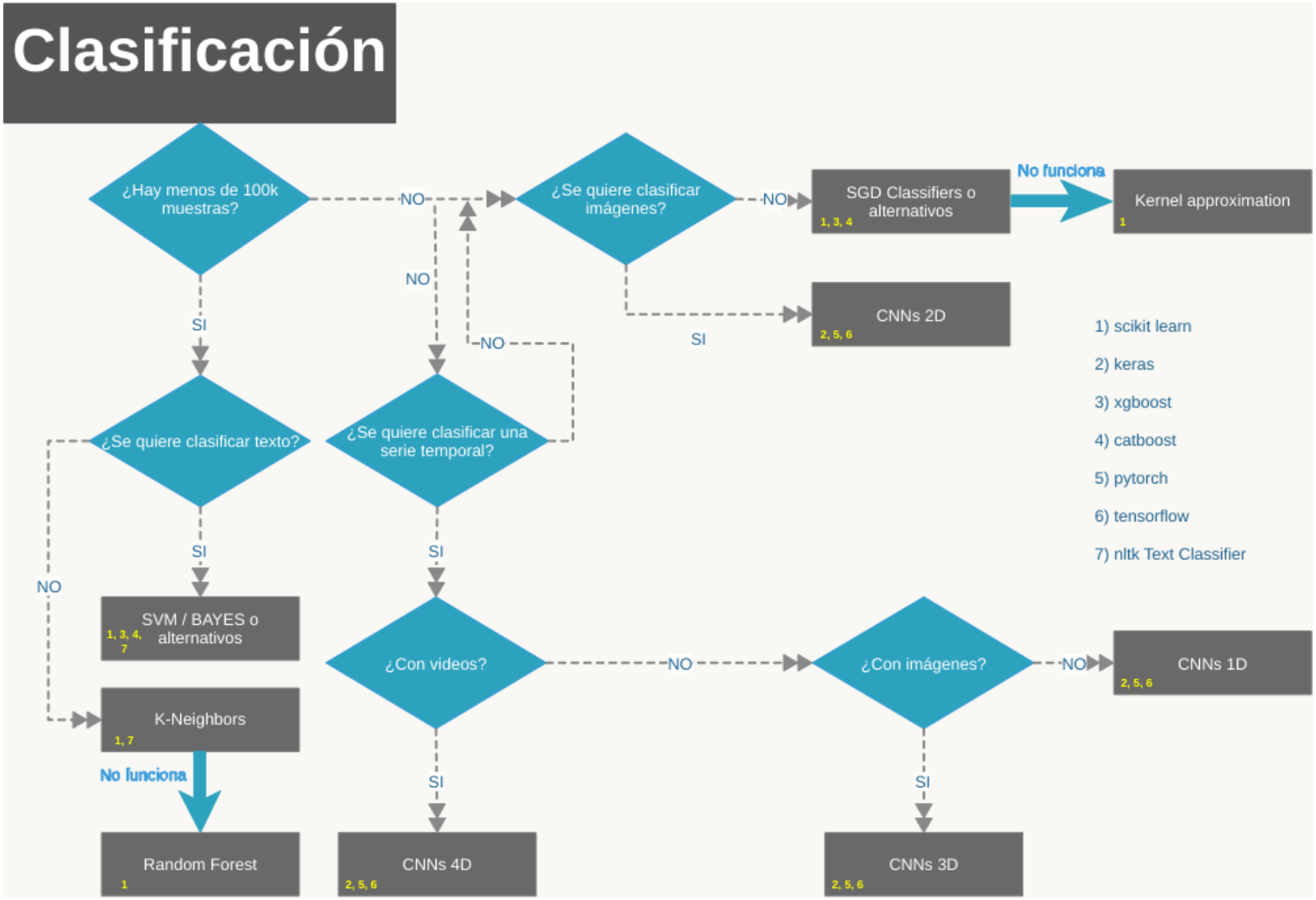
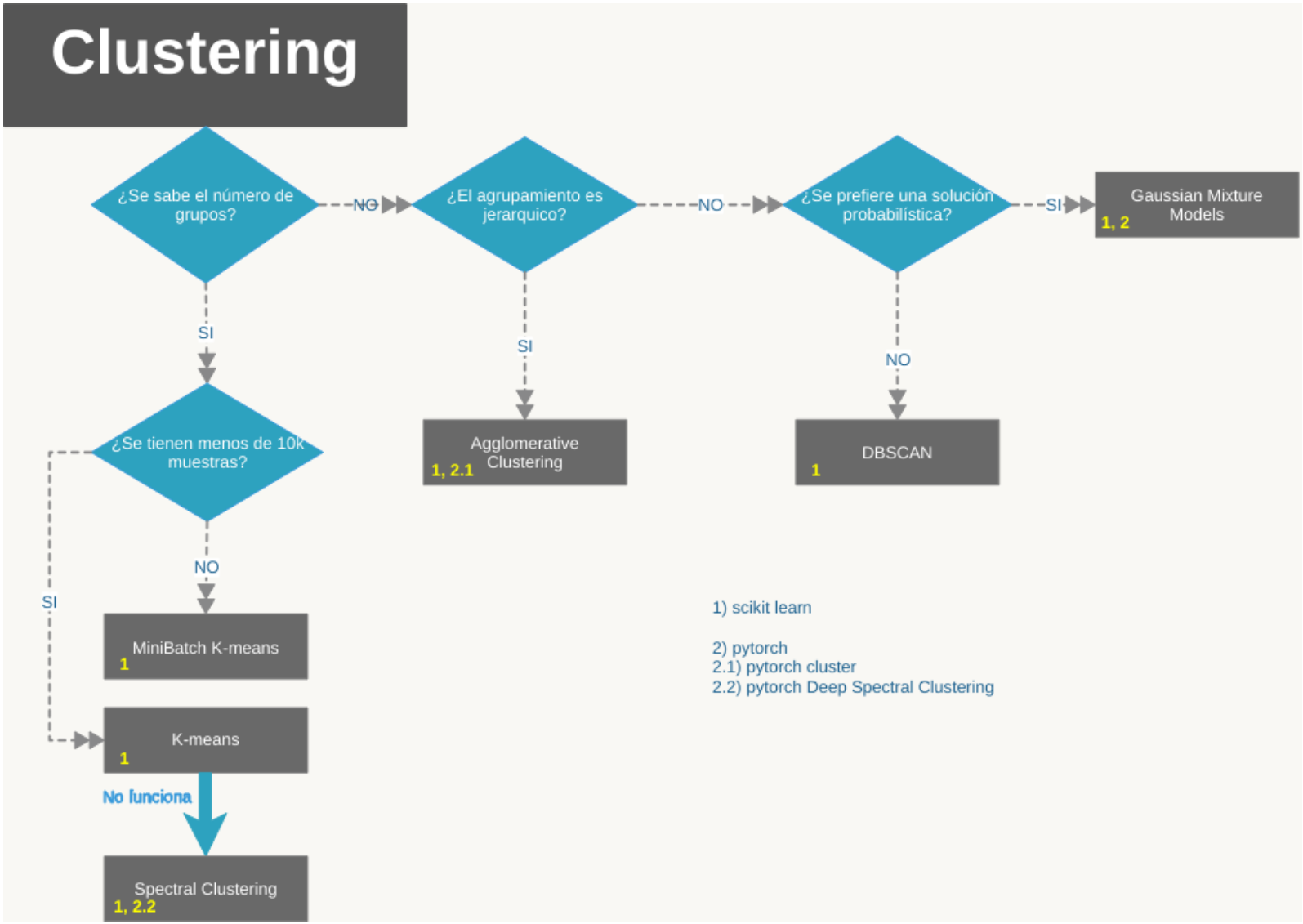
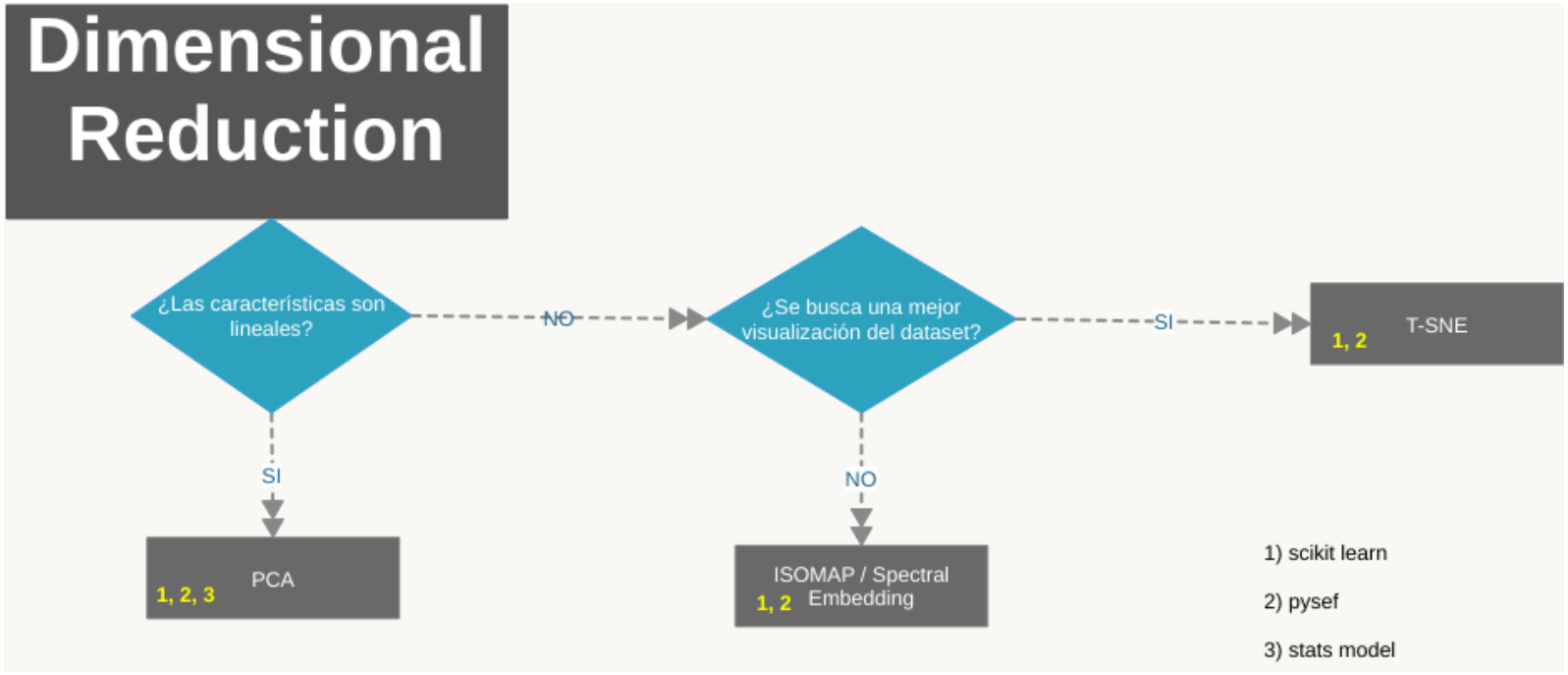


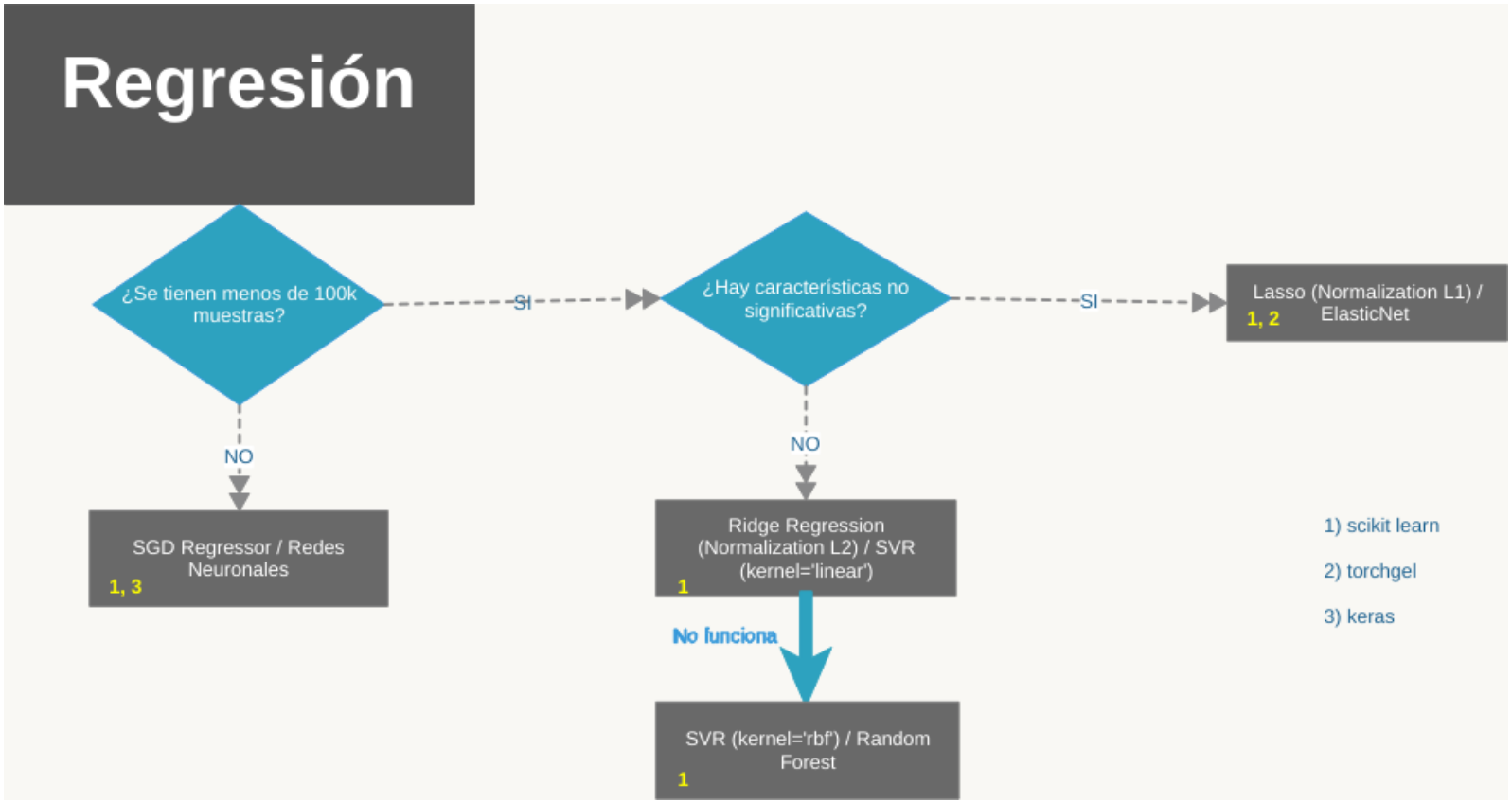
Fig. 11: Basado en colaboradores, estrellas y commits de GitHub (tamaño del circulo).


START
ML Cheat Sheet
¿Hay más de 50 muestras?
SI
¿Se busca una mejor visualización de los datos?
NO
¿Se busca detectar anomalías?
NO
¿Se tienen los valores de salida?
NO
CLUSTERING
NO
¡Se necesitan más muestras!
SI
DIMENSIONAL REDUCTION
SI
ANOMALY DETECTION
SI
¿Se busca una categoría?
SI
CLASSIFICATION
NO
¿Se busca un valor numérico?
SI
REGRESSION
NO
STRUCTURE
Aprendizaje Supervisado
Aprendizaje No Supervisado


Clasificación
¿Hay menos de 100k muestras?
NO
¿Se quiere clasificar imágenes?
NO
SGD Classifiers o alternativos
1, 3, 4
No funciona
Kernel approximation
1
SI
CNNs 2D
2, 5, 6
SI
NO
NO
¿Se quiere clasificar texto?
¿Se quiere clasificar una serie temporal?
SI
SI
NO
SVM / BAYES o alternativos
1, 3, 4, 7
K-Neighbors
1, 7
No funciona
Random Forest
1
¿Con videos?
NO
¿Con imágenes?
NO
CNNs 1D
2, 5, 6
SI
SI
CNNs 4D
2, 5, 6
CNNs 3D
2, 5, 6
1) scikit learn
2) keras
3) xgboost
4) catboost
5) pytorch
6) tensorflow
7) nltk Text Classifier


Clustering
¿Se sabe el número de grupos?
NO
¿El agrupamiento es jerarquico?
NO
¿Se prefiere una solución probabilística?
SI
Gaussian Mixture Models
1, 2
SI
SI
NO
¿Se tienen menos de 10k muestras?
Agglomerative Clustering
1, 2.1
DBSCAN
1
NO
SI
MiniBatch K-means
1
K-means
1
No funciona
Spectral Clustering
1, 2.2
1) scikit learn
2) pytorch
2.1) pytorch cluster
2.2) pytorch Deep Spectral Clustering


Dimensional Reduction
¿Las características son lineales?
NO
¿Se busca una mejor visualización del dataset?
SI
T-SNE
1, 2
SI
NO
PCA
1, 2, 3
ISOMAP / Spectral Embedding
1, 2
1) scikit learn
2) pysef
3) stats model


Regresión
¿Se tienen menos de 100k muestras?
SI
¿Hay características no significativas?
SI
Lasso (Normalization L1) / ElasticNet
1, 2
NO
SGD Regressor / Redes Neuronales
1, 3
NO
Ridge Regression (Normalization L2) / SVR (kernel='linear')
1
No funciona
SVR (kernel='rbf') / Random Forest
1
1) scikit learn
2) torchgel
3) keras

10 Conclusión

En este trabajo se introdujeron los conceptos fundamentales relacionados al aprendizaje automático de manera que el lector se familiarice con los mismos y pueda tener el marco teórico necesario para solucionar un problema utilizando las librerías existentes. Se realizó un análisis del estado del arte de las herramientas en el área de Machine Learning, investigando las distintas librerías, aplicaciones y alternativas más usadas en estos últimos años.

De esta manera, se hace evidente lo útil que es tener un punto de partida y una guía que permita bajar a tierra los conocimientos aprendidos. La gráfica desarrollada posibilita obtener esto, brindando una clasificación de los distintos algoritmos a utilizar y un método de selección rápido y preciso.

Con este panorama claro y general y con la ayuda visual para seleccionar la herramienta adecuada en base a los requerimientos del proyecto, se puede avanzar en la resolución de cualquier problema específico donde se quiera implementar Machine Learning. De todos modos, se pretende seguir mejorando y actualizando la gráfica a medida que surjan nuevas herramientas, para mantenerse a la vanguardia de la tecnología.

References

[1] Gerardo Ortega. Gerardo Ortega - ¿Qué es Machine Learning y porqué éste es el mejor tiempo para comenzar a aprender del tema?

[2] Andreas C Müller, Sarah Guido, et al. *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.", 2016.

[3] Sebastian Raschka. *Python machine learning*. Packt Publishing Ltd, 2015.

[4] Trent Hauck. *scikit-learn Cookbook*. Packt Publishing Ltd, 2014.

[5] Damián Jorge Matich. Redes neuronales: Conceptos básicos y aplicaciones. 2001.

[6] Pablo Rozas Larraondo, Inaki Inza, and Jose A Lozano. Automating weather forecasts based on convolutional networks. In *Proceedings of ICML Workshop on Deep Structured Predictions*, 2017.

[7] Dr. Avishek Pal and Dr. PKS Prakash. *Practical Time Series Analysis*. Packt Publishing Ltd., 2017.

[8] John Paul Mueller. John Paul Mueller - O'Reilly Media.

[9] Luca Massaron. Luca Massaron - Scholar Google.

[10] John Paul Mueller and Luca Massaron. *Machine learning for dummies*. John Wiley & Sons, 2016.

[11] Andreas Mueller. Andreas C. Muller - Machine Learning Scientist.

[12] Andreas Mueller. Machine Learning Cheat Sheet (for scikit-learn).

[13] Dr. Hui Li. Hui Li, Author at SAS Blogs.

[14] Dr. Hui Li. Which machine learning algorithm should I use? - The SAS Data Science Blog.

[15] Dan Clark. Top 8 Python Machine Learning Libraries.

[16] Wikipedia. Comparison of deep-learning software.

Diagrama de Gantt

ID	Task Name	Start	Finish	Duration	Complete
1	Capacitación Scikit Learn	10/12/18	21/12/18	60 h.	100%
2	Investigación series temporales	1/2/19	6/2/19	24 h.	100%
3	Investigación Redes Neuronales Convolucionales (CNNs)	7/2/19	11/2/19	18 h.	100%
4	Puesta en marcha de modelo de aplicación de CNNs	12/2/19	18/2/19	30 h.	70%
5	Desarrollo teorico de las herramientas de Machine Learning	19/2/19	20/2/19	12 h.	100%
6	Desarrollo de Cheat Sheet	21/2/19	26/2/19	24 h.	100%
7	Confección de informe en lenguaje Latex	27/2/19	1/3/19	18 h.	100%
8	Feedback Cheat Sheet	6/3/19	8/3/19	18 h.	100%