Investigación Corta #3: Aprendizaje No Supervisado (Unsupervised Learning).

Carlos Brenes Jiménez, 9002170

***Resumen*—En este trabajo de investigación se aborda el tema de "Aprendizaje No Supervisado", su propuesta de valor, algoritmos existentes, colaterales requeridos y principales retos. La parte práctica se complementa con un cuaderno en *"Jupyter Lab"* que muestra con ejemplos prácticos sus diferentes modos de uso.**

***Palabras Claves*—Unsupervised Learning, bibliotecas, algoritmos, usos.**

I. Aprendizaje de Máquina ("Machine Learning"): [1] [2]

Definiciones:

- **Modelo de Aprendizaje**: es aquel que aprende de los datos de entrenamiento (Experiencia) para mejorar su tasa de error (Desempeño) en una tarea, y el criterio de éxito final es qué tan bien su experiencia generaliza a algo nuevo, datos nunca antes vistos (error de generalización). La entrada vectorial $\bar{X}$ es transformada en una salida vectorial $\bar{Y}$. Figura 1.

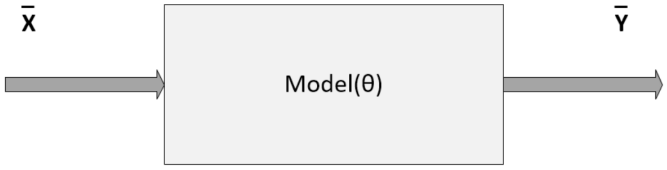

Fig. 1: **Esquema de un modelo genérico con vector** θ [2]

  En la tarea de aprendizaje de máquina, la meta es alcanzar la máxima exactitud, iniciando con el conjunto de entrenamiento y depués con el conjunto de validación. Se busca mejorar el modelo para llegar lo más cerca a la exactitud de Bayes. La exactitud de Bayes es un límite teórico superior que se puede alcanzar usando un estimador. En la figura 2 se representa el proceso.
- **Aprendizaje Supervisado**: El agente de **AI ("Inteligencia Artificial")** tiene acceso a etiquetas, las cuales pueden usarse para mejorar su desempeño en alguna tarea.
- **Aprendizaje No Supervisado**: las etiquetas no están disponibles. Por lo tanto la tarea del agente de *AI* no está bien definida, y su desempeño no puede ser tan claramente medido. El problema del Aprendizaje No Supervisado está menos claramente definido que el problema de Aprendizaje Supervisado y tiene una complejidad mayor para el agente de *AI* de poder resolverlo. Un contraste entre los modelos de aprendizaje se muestra en la tabla I.

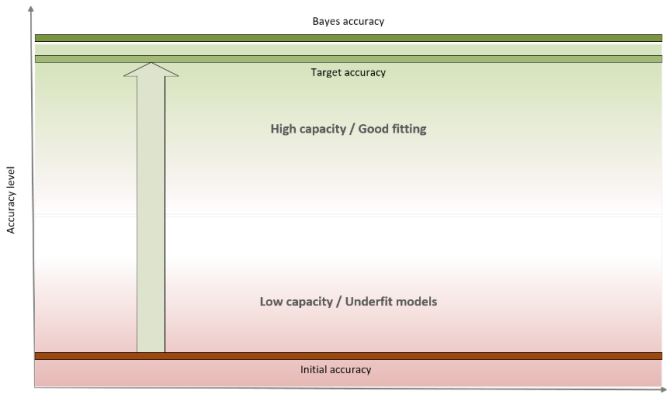

Fig. 2: **Diagrama de los niveles de Exactitud** [2]

- **Variables de entrada:** alimentan los modelos de aprendizaje. Por ejemplo en un sistema de detección de "Spams", los correos electrónicos son la variable de entrada. Se cononocen también como *Características*, *"Features"*, *Predictores* o *Variables Independientes*. La naturaleza de las muestras se puede definir como un proceso estocástico de generación de datos asociada a una distribución de probabilidad conjunta.

$$p_{data}(\bar{x}, \bar{y}) = p(\bar{y}|\bar{x})p(\bar{x})$$

  Si se muestrea *N* valores independientes e idénticamente distribuídos (i.i.d.) de p_{data}, se puede crear un conjunto finito de datos $\mathbf{X}$ compuesto de vectores reales de *k-dimensiones*:

$$\mathbf{X} = \{\bar{x}_0, \bar{x}_1, ..., \bar{x}_n\} \; donde \; \bar{x}_i \in \mathcal{R}^k$$

- **Variable de Salida:** es lo que se busca predecir. Por ejemplo, la etiqueta de "Spam" y "No Spam". Se conoce también como *Variable Objectivo "Target"*, *Variable dependiente*, *Variable de respuesta* o *Clase*.

$$\mathbf{Y} = \{\bar{y}_0, \bar{y}_1, ..., \bar{y}_n\} \; donde \; \bar{y}_i \in \mathcal{R}^t$$

- **Conjunto de Entrenamiento:** es el conjunto de ejemplos en donde la *Inteligencia Artificial* se entrena. Cada ejemplo individual se conoce como *Instancia de Entrenamiento o Muestra*. En el proceso de entrenamiento, el **AI** intenta minimizar su *función de costo o tasa de error* o convertirla más positivamente, para maximizar su *función de valor*. Un ejemplo es la razón de correos correctamente clasificados. El *AI* activamente optimiza una *tasa mínima de error* durante el entrenamiento. Su tasa de error se

determina al calcular la etiqueta predecida con la etiqueta verdadera.

- **"Generalización":** es qué también el modelo generaliza/predice de su entrenamiento cuando se enfrenta ante muestras no antes vistas.
- **Conjunto de Prueba (Test set o conjunto retenido del conjunto de entrenamiento)**: se busca tener un conjunto de prueba para medir el error de generalización durante el entrenamiento.
- **Conjunto de Validación (Validation Set)**: son conjuntos intermedios de datos retenidos que se utilizan para evaluar el progreso realizado antes del conjunto final de prueba. En otras palabras, se utiliza para "medir" el desempeño del modelo sin ninguna parcialidad ("bias") con muestras nunca antes vistas.
- **Dimensiones del conjunto de entrenamiento y validación**: deacuerdo a la naturaleza del problema, es posible escoger una distribución porcentual de un 70% a un 30% (para conjuntos de datos relativamente pequeños), o más porcentaje en el conjunto de entrenamiento (80%, 90% y hasta 99%) para aprendizaje profundo donde el número de muestras es muy grande.
- **Validación Cruzada**: Cabe la posibilidad de seleccionar un conjunto de prueba incorrecto. Un método válido para detectar este problema es esta técnica. Particularmente, se usa el acercamiento de validación cruzada (K-pliegues o K-fold). La idea es de dividir el conjunto entero de datos **"X"** en un conjunto de prueba y entrenamiento dinámicos (el conjunto restante). El tamaño del conjunto de prueba se determina por el número de pliegues, tal que durante las "k" iteraciones, el conjunto de prueba cubra el conjunto original de los datos como se muestra en la figura 3.

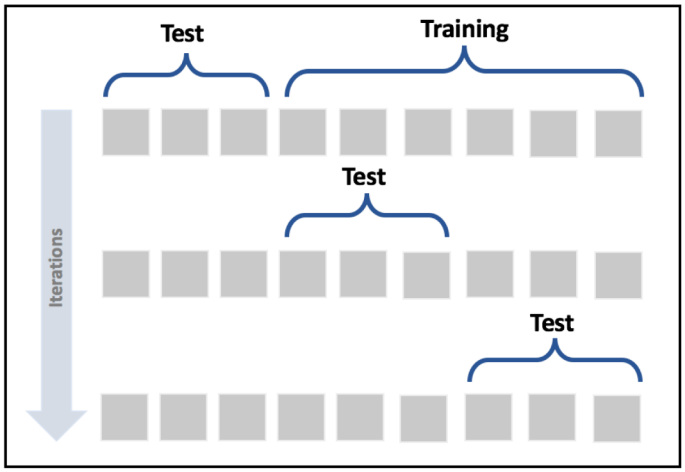

Fig. 3: **Esquema de Validación Cruzada de "K-pliegues"** [2]

II. Aprendizaje No Supervisado ("Unsupervised Learning" - UL): [1] [3]

Éxitos recientes en el Aprendizaje de Máquina se han debido por la disponibilidad de una gran cantidad de datos, avances en el hardware computacional y en los recursos en la nube, y en los avances de los algoritmos del apredizaje de máquina. Pero estos éxitos han sido mayormente en problemas estrechos de AI como lo son la clasificación de imágenes, visión por computadora, reconocimiento del habla, procesamiento natural del lenguaje, y la traducción de máquina. Para resolver problemas más ambiciosos de AI, se necesita liberar el valor del Aprendizaje No Supervisado.

- **Fortalezas**:
  1) Altamente efectivo en problemas donde los patrones son desconocidos o que cambian constantemente o no hay un conjunto de datos etiquetados en forma abundante.
  2) Aprende la estructura subyacente de los datos en los que se entrena. Esto lo hace al tratar de representar los datos en lo que se entrena con un conjunto de parámetros que es significativamente más pequeño que las muestras disponibles en el conjunto de datos original.
  3) Al ejecutar esta representación de aprendizaje, es capaz de identificar distintos patrones en el conjunto de datos. Por ejemplo en un conjunto de datos de imágenes(sin etiquetas), el *AI* de Aprendizaje No Supervisado puede ser capaz de identificar y agrupar imágenes con base a similitudes entre ellas y qué tan diferente son las mismas del resto. Por ejemplo, todas las imágenes que se parecen a sillas serán agrupadas entre sí, y las imágenes que se parecen a perros serán agrupadas entre sí y así sucesivamente. Después del entrenamiento inicial, si el *AI* de Aprendizaje No Supervisado encuentra imágenes que no pertenecen a ninguno de los grupos etiquetados, el *AI* creará grupos separados para las imágenes no clasificadas, para que un humano etiquete los grupos de imágenes todavía sin etiquetar.
  4) Permite resolver problemas más complicados y es más ágil en encontrar patrones ocultos tanto en datos históricos disponibles para entrenamiento y en datos futuros.
  5) Es mejor en atacar problemas abiertos y en generalizar este conocimiento.
  6) Se puede etiquetar automáticamente ejemplos no etiquetados. Por ejemplo, se agrupan las muestras en un "cluster" y después se aplican las etiquetas de muestras etiquetadas a las muestras no etiquetadas en el mismo "cluster". Las muestras no etiquetadas recibirían las etiquetas de aquellas muestras a las que son similares.
- **Debilidades**:
  1) No se puede obtener información precisa acerca de la clasificación de los datos, y la salida de los datos usada como entradas de *UL* es etiquetada y no conocida.
  2) Exactitud menor en los resultados ya que los datos de entrada son desconocidos y no etiquetados anteriormente. El algoritmo lo tiene que hacer por su cuenta.

TABLA I: Comparación del Aprendizaje No Supervisado y el Aprendizaje Supervisado. [3]

Parámetros	**No Supervisado**	**Supervisado**
Datos	*No etiquetados*	*Etiquetados*
Implementación Computacional	*Compleja*	*Simple*
Exactitud/Confiabilidad	*Menor*	*Alta*

3) Las clases espectrales no siempre corresponder a las clases de información.
4) El usuario necesita invertir tiempo en interpretar y etiquetar las clases provenientes del proceso de clasificación.
5) Las propiedades espectrales del las clases pueden cambiar con el tiempo, por lo tanto no se puede mantener la misma información de la clase para casos futuros.

A. *Retos del Aprendizaje No Supervisado*

1) **Overfitting**: es cuando un algoritmo de aprendizaje de máquina tiene un desempeño en datos nunca antes vistos (conjuntos de datos de validación y prueba). La causa es que el algoritmo ha extraído mucho del ruido en los datos y tiene un error de generalización muy pobre. En otras palabras el algoritmo está "memorizando" el conjunto de entrenamiento en lugar de aprender cómo generalizar el conocimiento basado en él. Figura 4

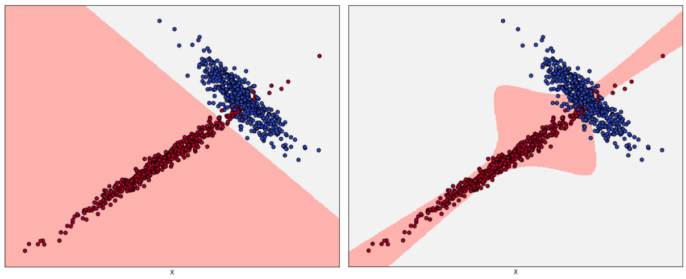

Fig. 4: **Ajuste Aceptable (Izquierda), Clasificador Sobre-ajustado (Derecha)** [2]

   – **Uso de UL**: como *Regularizador*. El proceso de regularización es un proceso usado para reducir la complejidad de un algoritmo de aprendizaje de máquina. Permita capturar la señal en los datos sin ajustar demasiado hacua el ruido. En lugar de que se alimente directamente los datos al algoritmo supervisado, se alimenta una nueva representación de los datos generados por al algoritmo *No Supervisado*. Se conoce también como *Extracción de Características o "Feature Extraction"*.
2) **"Maldición" de la Dimensionalidad**: Entre más características se tengan, el entrenamiento se complica. Por ejemplo, en espacios dimensionales muy grandes, los algoritmos supervisados, la clasificación se vuelve muy costosa, desde el punto de tiempo y cálculos computacionales. Causando que sea imposible encontrar una solución en forma rápida.
   – **Uso de UL**: se trata de encontrar las caraterísticas más importantes del conjunto de datos original. Al hacer esto, se reduce el número de dimensiones a un número más manejable perdiendo muy poca información en el proceso para después aplicar algoritmos supervisados para un desempeño más eficiente en encontrar una función de aproximación
3) **Ingeniería de Características ("Feature Engineering")**: sin las características correctas, los algoritmos de *ML* no son capaces de separar puntos en el espacio en forma correcta y por lo tanto no pueden tomar decisiones en datos nunca antes vistos.
   – **Uso de UL**: Se usa la representación de aprendizaje de los algoritmos *No Supervisados* para automáticamente aprender los tipos de representación de las características correctas para resolver una tarea en particular.
4) **Valores Atípicos u "Outliers"**: La calidad de los datos es muy importante. Si un algoritmo se entrena con datos atípicos su error de error será alto.
   – **Uso de UL**: Se detectan los valores atípicos al usar la reducción de dimensionalidad y se crean soluciones específicas para los "outliers" y otra para los datos normales.
5) **Desplazamiento de Datos o "Data Drift"**: Los algoritmos de ML necesitan detectar cuando hay un desplazamiento en los datos. Esto quiere decir que si en los datos que el modelo hace predicciones son estadísticamente diferentes a los datos en que el modelo se entrenó, el modelo necesita re-entrenarse en datos que sean más representativos de los datos actuales. Si no se re-entrena el modelo no es capaz de identificar el desplazamiento, la calidad en la predicción se verá impactada.
   – **Uso de UL**: En la construcción de distribuciones de probabilidad, se puede evaluar en cuánto difieren los datos actuales con respecto a los datos de entrenamiento, si la diferencia es significativa, se inicia un proceso de re-entrenamiento.

B. *Bibliotecas o Marcos de Referencia para Aprendizaje de Máquina [4]*

Una descripción de las bibliotecas más utilizadas en Aprendizaje de Máquina (tanto No Supervisado como Supervisado) se muestra a continuación.

- **NumPy** [5]: paquete de próposito general para el procesamiento de arreglos multi-dimensionales y matrices. Es muy útil para el manejo de álgebra lineal, transformadas de Fourier y números aleatorios. Tensor Flow la utiliza para manipular tensores.Tiene herramientas para integrar código c/c++ y Fortran.
- **SciPy** [6]: Originalmente fue desarrollada por Travis Oliphant, Eric Jones, and Pearu Peterson en 2001. En la actualidad el desarrollo es soportado por una comunidad abierta de desarrolladores y se distribuye bajo una licencia gratuita BSD. Tiene módulos de álgebra lineal, optimización de imágenes, integración de interpolación, funciones especiales, Transformada Rápida de Fourier (FFT), procesamiento de señales e imágenes, solución de ecuaciones diferenciales ordinarias entre muchas más tareas. La estructura de datos subyacente que usa es de arreglos multi-dimensionales provenientes de **numpy**. Tiene una interface amigable con el usuario.
- **Scikit-learn** [7]: fue desarrollada por David Cournapeau en el 2007 como parte del proyecto código de verano en Google. Se construyó con base en **NumPy** y **SciPy**. Tiene una amplia variedad de algoritmos de aprendizaje no supervisado y supervisado que se ejecutan en un ambiente Python. La biblioteca se utiliza también en minería de datos y análisis de datos. Con respecto al Aprendizaje de Máquina puede manejar tareas de Clasificación, Regresión, Agrupamiento, Reducción de dimensionalidad, selección de modelos, y pre-procesamiento.
- **Theano** [8]: puede actuar como compilador de optimización para evaluar y manipular expresiones matemáticas y cálculos de matrices. Su base es **NumPy** y su sintáxis es muy similar. Puede trabajar tanto en Unidades de Procesamiento Gráficas(GPU) como en CPUs. Tiene un alto desempeño en arquitecturas de GPU (140x con respecto a CPUs). Puede evitar errores y fallos en operaciones que involucran funciones logarítmicas y exponenciales. Tiene herramientas para prueba y validación de unidades funcionales.
- **TensorFlow** [9]: desarrollada originalmente para uso interno en Google(2015). Permite la creación de modelos de aprendizaje de máquina. Tiene soporte de diferentes conjunto de herramientas para construir modelos con diferentes niveles de abstracción. Tiene APIs estables para Python y C++. Tiene una arquitectura flexible lo que le permite funcionar con una variedad de plataformas de CPUs, GPUs y TPUs (Unidad de Procesamiento de Tensores).
- **Keras** [10] : tiene más de 200 mil usuarios. Es una biblioteca de código abierto ampliamente usada en redes neuronales y aprendizaje de máquina. Keras puede ejecutarse sobre TensorFlow, Theano, Microsoft Cognitive Toolkit, R, o PlaidML. Puede ejecutarse eficientemente tanto en CPUs como GPUs y trabaja con bloques o elementos de redes neuronales, como lo son capas, objetivos, funciones de activación y optimizadores. Tiene la capacidad de trabajar en imágenes e imágenes de texto que se utilizan al escribir código de Redes Neuronales Profundas. También soporta Redes Neuronales Convolucionales y Recurrentes.
- **PyTorch** [11]: tiene muchas herramientas y bibliotecas para el soporte de visión por computadora, aprendizaje de máquina, procesamiento natural del lenguaje. Es de código abierto y su base es la librería **Torch** [12]. Es muy fácil de aprender y usar. Integra fácilmente a **NumPy**. Permite también el cálculo de Tensores. Tiene un marco de trabajo que permite construir gráficas en tiempo de ejecución. Soporta GPUs, Pre-procesadores y carga de datos.
- **Pandas** [13]: Es una biblioteca rápida, flexible y con estructuras claras de datos para facilitar el trabajo con datos relacionales o etiquetados. Es altamente estable y de un óptimo desempeño. Su código es **C** o **Python**. Los dos tipos principales de estructuras de datos son:
  1) **Series**: una dimensión.
  2) **Dataframe**: dos dimensiones.

  En conjunto pueden manejar una vasta variedad de requerimientos de datos de diferentes areas: ciencia, estadística, social, financiera, analítica y otras ciencias de ingeniería.
- **Matplotlib** [14]: es una biblioteca usada para visualización de datos 2D y figuras en una gran variedad de formatos. La biblioteca ayuda a generar histogramas, gráficos de error, dispersión, barras, etc. Provee una interface similar a **Matlab**. Utiliza herramientas GUI como GTK+, wxPython, Tkinter y Qt para proveer APIs orientadas a objectos.

C. Algoritmos No Supervisados usando Scikit-Learn (Unsupervised Learning Algorithms - ULA)

1) **Reducción de Dimensionalidad**: Estos algoritmos proyectan los datos de entrada originales de alta dimensionalidad a un espacio dimensional menor al filtrar las características no relevantes y por lo tanto mantiene las más relevantes tanto como sea posible. Esto permite al UL *AI* más efectivamente identificar patrones y eficientemente resolver problemas que implican una escala más amplia con costos computacionales más altos (como por ejemplo imágenes, video, habla y texto). Hay dos tipos de algoritmos de dimensionalidad: 1) Proyección Lineal y 2) Reducción Dimensional No lineal.
   - **Proyección Lineal**:
     a) **Análisis de Componentes Principales (PCA)**: Se enfoca en identificar cuáles características del conjunto de datos son más importantes en explicar variabilidad (permitiendo así que el modelo pueda separar los datos posteriormente). Aquellas características que varían poco no ayudan en explicar el conjunto de datos. Figura 5. Es

claro que se perderá una parte de la variación al moverse a un espacio dimensional menor, pero la estructura subyacente de los datos será fácil de identificar, permitiendo así hacer tareas como el agrupamiento ("Clustering") más eficiente. Algunos ejemplos de algoritmos de esta familia son: PCA Incremental y variantes no lineales como es el "Kernel" PCA y variantes dispersas como lo es el "Sparse" PCA.

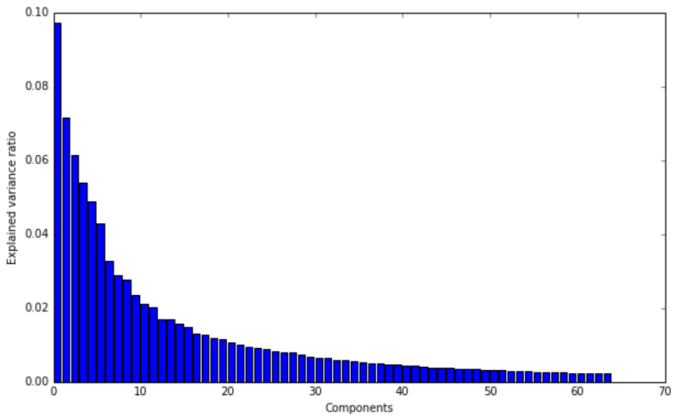

Fig. 5: **Razón de Varianza explicada vs. Número de Componentes** [2]

b) **Descomposición en Valores Singulares (SVD)**: se fundamenta en la reducción del rango ("rank") de la matriz original de características a una matriz de rango más pequeño tal que la matriz original puede generarse al usar una combinación lineal de algunos vectores en la matriz de rango menor. Esto se conoce como *SVD*. Al generar la matriz de rango menor, el *SVD* mantiene los vectores de la matriz original que tengan la mayor parte de la información (los valores con los valores singulares más altos.

c) **Proyección Aleatoria** usa el mismo pricipio de proyectar puntos de un espacio dimensional mayor a uno menor en donde la escala de las distancias entre puntos se mantiene. Se puede usar una matriz aleatoria Gaussiana o una matriz aleatoria Dispersa para lograr la reducción en dimensión.

– **Reducción Dimensional No lineal o Aprendizaje de Colección o Múltiple ("Manifold Learning"**:

a) **ISOMAP**: Este algoritmo aprende la geometría de los datos colectados al estimar la *geodesia* o la distancia curva entre cada punto y sus vecinos en lugar de la distancia Euclideana. Esta característica se usa para transformar de un espacio dimensional a uno menor. Figura 6. Un ejemplo de ISOMAP aplicado en imágenes se muestran en las figuras 7 y 8.

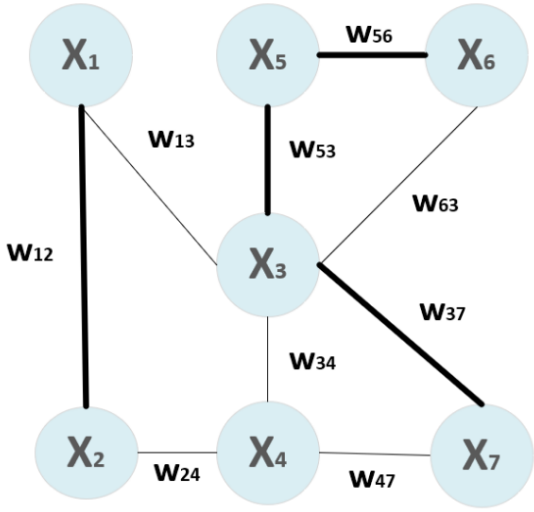

Fig. 6: **Ejemplo de un grafo con las distancias más cortas marcadas** [2]

Fig. 7: **Subconjunto del conjunto de datos de rostros Olivetti** [2]

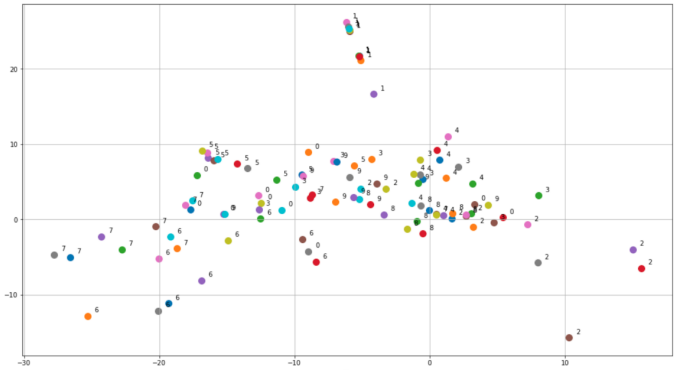

Fig. 8: **Ejemplo de Isomap aplicado a 100 muestras del conjunto de datos de rostros Olivetti** [2]

b) **Embeber el t-vecino estocástico distribuído (t-SNE)**: Este algoritmo embebe datos de alta dimensionalidad en un espacio de dos o tres dimensiones, permitiendo así que los datos transformados puedan ser visualizados. Las instancias similares se modelan juntas y las que no se modelan separadas. Figura 9.

– **Aprendizaje por Diccionario**: Se aprende la representación de los datos subyacentes. Los elementos representativos son vectores simples, y

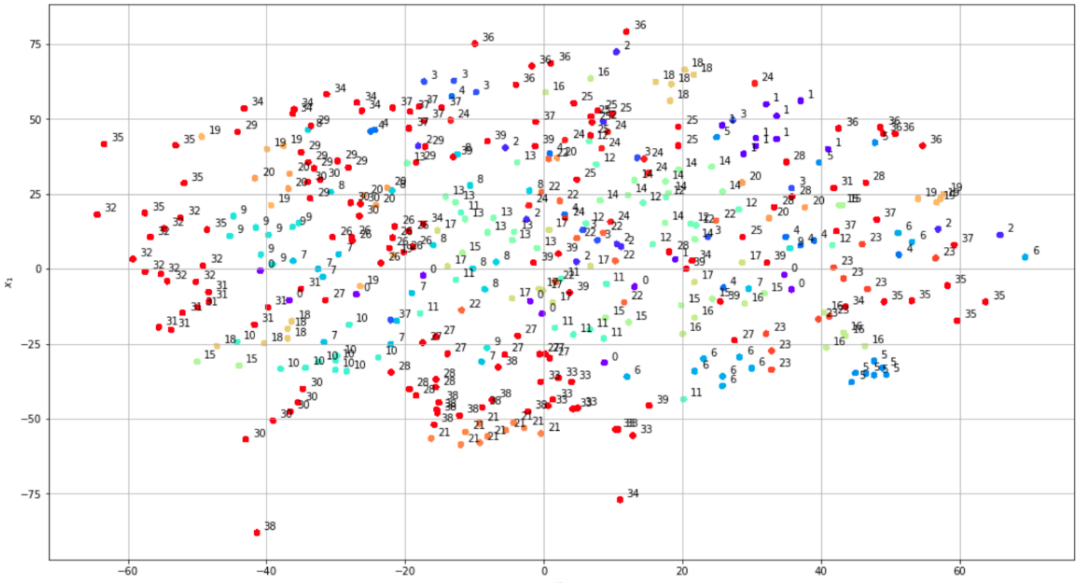

Fig. 9: **Ejemplo de t-SNE aplicado al conjunto de datos de rostros Olivetti** [2]

cada instancia en el conjunto de datos se representa como un vector con peso, y puede reconstruirse como una suma de pesos de los elementos representativos. Los elementos representativos generados se conocen como el *diccionario*. Este algoritmo puede identificar los elementos más representativos del espacio dimensional original (con la mayor cantidad de pesos que no sean cero). Los elementos menos importantes son los que tienen pocos pesos diferentes a cero. Esto permite que el algoritmo pueda aprender la estructura subyacente de los datos, lo que ayuda a separar los datos y poder identificar patrones.

– **Análisis de Componentes Independientes (ICA)**: un problema con los datos sin etiquetar es que hay muchas señales independientes embebidas junto a las características dadas. Al usar este algoritmo, se pueden separar las señales mezcladas en sus componentes individuales. Figuras 10 y 11. Después de realizar la separación, se puede reconstruir cualquier característica original al sumar en conjunto alguna combinación de los componentes generados individualmente. Este algoritmo se usa en tareas de procesamiento de señales (por ejemplo, identificar las voces individuales en una grabación de audio de un lugar como una cafetería).
– **Asignación Latente Dirichlet(LDA)**: UL también puede explicar un conjunto de datos al aprender el por qué algunas partes del conjunto de datos son similares a otras. Esto requiere aprender elementos que no se observan en el conjunto de datos. Por ejemplo, un documento con muchas palabras. Estas palabras no son aleatorias; las palabras tienen un estructura. Esta estructura puede modelarse como elementos no

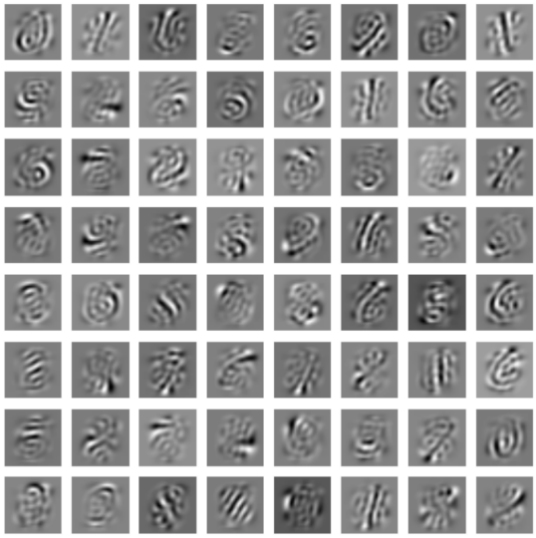

Fig. 10: **Componentes Independientes del conjunto de datos MNIST extraídas por el algoritmo FastICA (64 componentes)** [2]

observados y se conoce como *tópicos*. Después del entrenamiento, LDA es capaz de explicar dado un documento con un conjunto pequeño de tópicos, dónde está cada tópico, hay un pequeño conjunto de palabras que son usadas. Esta estructura oculta es la que *LDA* puede capturar, y por lo tanto ayudar a explicar una parte no estructurada del texto.

2) **Agrupamiento (“Clustering”)**: Una vez que se reduce el conjunto de características originales a un conjunto más manejable, se pueden encontrar patrones interesantes al agrupar instancias similares. Esto se conoce como agrupamiento y se logra por

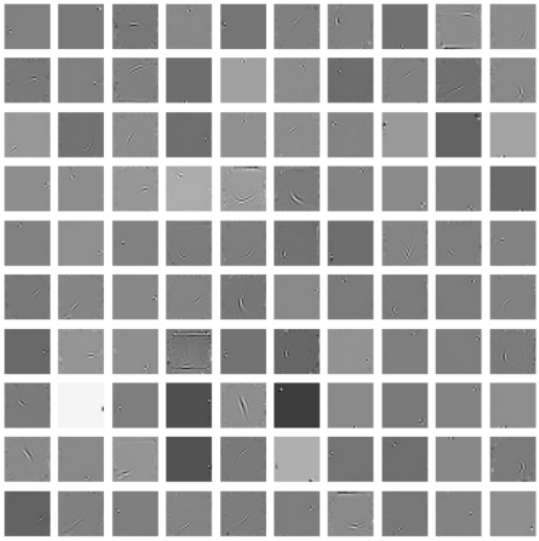

Fig. 11: **Componentes Independientes del conjunto de datos MNIST extraídas por el algoritmo FastICA (640 componentes)** [2]

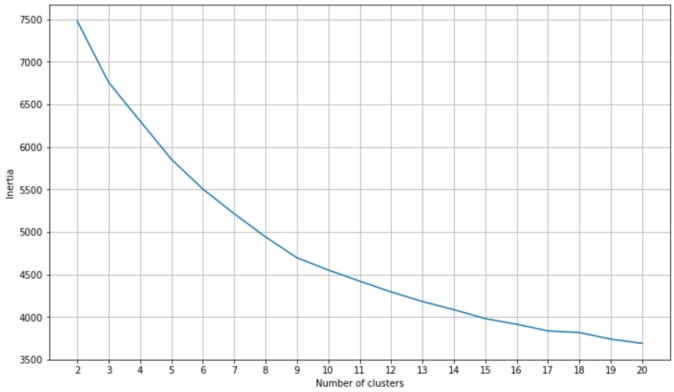

Fig. 12: **Inercia como función del número de los grupos o "clusters".** [2]

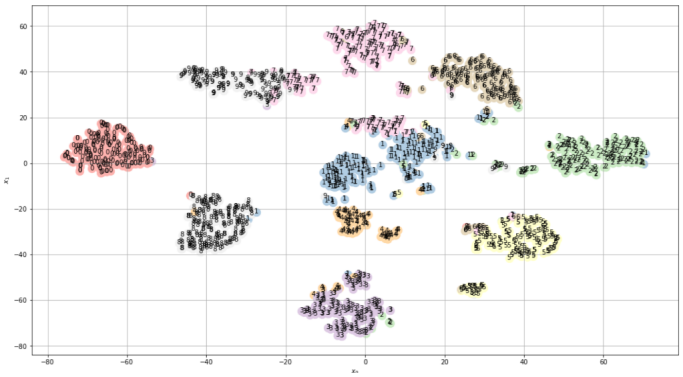

Fig. 13: **Representación t-SNE del conjunto de datos MNIST; las etiquetas corresponden a los grupos o "clusters".** [2]

medio de algoritmos no supervisados y tiene varias aplicaciones (por ejemplo en la segmentación de mercados).

– **K-Means**: Se especifica el número deseable de grupos K, y el algoritmo asignará cada instancia exactamente a uno de los K grupos. Esto optimiza el agrupamiento al minimizar la variación dentro del "cluster" (esto se conoce como *Inercia* Figura 12) tal que la suma de las variaciones en los diferentes grupos K es la más pequeña posible. Para acelerar el proceso de agrupamiento, *K-Means* asigna aleatoriamente cada observación a uno de los K grupos y después re-asigna estas observaciones para minimizar la distancia Euclideana entre cada observación y el punto central del grupo conocido como *Centroide*. El resultado, al realizar varias corridas de cada *K-Means*, con un inicio aleatorio dará una asignación aleatoria de las observaciones. De las diferentes corridas, se selecciona la que tenga la mejor separación, definida como la que tenga la menor suma dentro del grupo de variaciones para todos los *k-grupos*. Figura 13.
– **Agrupamiento Jerárquico**: no se requiere precomprometer a un número particular de grupos. Una versión del algoritmo se conoce como *Agrupamiento Aglomerado*. El mismo se basa en usar en un agrupamiento tipo árbol y construye un *Dendrograma*. El Dendrograma se puede describir como un árbol invertido, donde las hojas están en la parte inferior y el tronco del árbol en la parte superior. Las hojas son instancias individuales en el conjunto de datos. El agrupamiento jerárquico, une las hojas (deacuerdo a similitudes) en sentido del movimiento vertical hacia arriba. Las instancias similares se unen más pronto. Las instancias no similares se unen más tarde. Al ser un proceso iterativo, todas las instancias se unirán formando un sólo tronco del árbol. Una vez que el algoritmo termina su corrida, se decide dónde "cortar" el árbol. Entre más bajo se corte, habrá una mayor cantidad de ramas individuales (más grupos). Si se quieren pocos grupos, se corta el dendrograma cerca del tronco. Esto es similar a escoger el número de *k-grupos* en el algoritmo de *K-Means*.
– **DBSCAN -"Agrupamiento de Aplicaciones con Ruido Espaciales de Densidad"**: Es un algoritmo de agrupamiento (basado en la densidad de los puntos). Dada todas las instancias en el espacio, *DBSCAN* agrupará aquellos puntos cercanos (esto es el número mínimo de instancias que deberían existir dentro de cierta distancia. Se especifican el mínimo número de instancias requeridas y la distancia. Si una instancia está dentro de la distancia de muchos grupos, será agrupada con el grupo en donde se encuentre densamente localizada. Cualquier instancia que no esté dentro de la distancia de otro grupo se

etiquetará como un valor atípico. No se necesita pre-especificar el número de grupos. Los grupos pueden tener cualquier forma y el algoritmo no es propenso a distorsiones causada por valores atípicos en los datos.
- **HDBSCAN**; es el DBSCAN jerárquico. Convierte el algoritmo DBSCAN a un algoritmo de agrupamiento jerárquico. En otras palabras, los grupos se crean con base a densidad y después los une iterativamente con base a la distancia.

D. Aplicaciones del Aprendizaje No Supervisado:

1) **Detección de Anomalías**: Son aquellos puntos conocidos como atípicos ("outliers"). Algunos ejemplos son la detección de fraudes en tarjetas de crédito, fraudes de transferencias, fraudes cibernéticos y de seguros, intrusos en dispositivos conectados, fallas de mantenimiento en medios de transporte, financiación de terroristas, lavado de dinero, tráfico de humanos, narcóticos y armas, etc. Los sistemas supervisados encuentran patrones conocidos con un alto nivel de exactitud, mientras los sistemas no supervisados descubren nuevos patrones que podrían ser interesantes. Una vez que los patrones son descubiertos por el *UL AI*, los patrones se etiquetan por humanos, transicionando más los datos de no etiquetados a etiquetados.
2) **Segmentación de Grupos**: Con el agrupamiento, se pueden segmentar grupos basados en similitudes en comportamiento para areas como el Mercadeo, Retención de Clientes, Diagnóstico de Enfermedades, Compras "On-line", escucha de música, videos, citas "on-line", actividad de medios sociales, y clasificación de documentos. Los datos disponibles son muchos y parcialmente etiquetados.

E. Aprendizaje No Supervisado usando TensorFlow y Keras:

- **Auto Codificadores o "Autoencoder"**: se generan nuevas representaciones de las características al usar una alimentación hacia adelante "feedforward" por medio de redes neuronales no-recurrentes para desempeñar el aprendizaje de representación, donde el número de nodos en la capa de salida se parea con el número de nodos en la capa de entrada. Esto permite re-construir las características originales, aprendiendo así una nueva representación por medio de las capas ocultas en el medio. Cada capa oculta del "Autoencoder" aprende una representación de las características originales, las capas siguientes construyen en la representación aprendida por las capas anteriores. Capa por capa, el "autoencoder" aprende representaciones más complicadas a partir de simples. La capa de salida es la representación final aprendida. Esta representación aprendida se puede convertir en la entrada de un modelo supervisado con el objectivo de mejorar el error de generalización.

F. Aprendizaje Profundo No Supervisado usando TensorFlow y Keras:

- **Extracción de Características**: Se basa en aprender nuevas representaciones de las características originales de los datos. Esto permite reducir el número de características originales a un subconjunto más pequeño, dándose asi una reducción de dimensionalidad. Pero también se pueden generar nueva representación de características para así mejorar el desempeño en problemas de aprendizaje supervisado.
- **Aprendizaje Profundo No Supervisado ("Deep Neural Networks - DNN"):** Hasta ahora, el entrenamiento de DNN no se podía rastrear computacionalmente. En estas redes neuronales, las capas ocultas aprenden representaciones internas para resolver problemas. Las representaciones se mejoran con el tiempo basado en cómo la red neuronal usa la función del *gradiente del error* en cada iteración de entrenamiento para actualizar los pesos de los diferentes nodos. Esta actualización es computacionalmente costosa y dos problemas que pueden occurrir son: 1) La función del gradiente del error se hace muy pequeña y ya que la propagación hacia atrás se fundamenta en multiplicar estos pesos en conjunto, los pesos de la red se actualizarán muy lentamente o no del todo, causando que el entrenamiento de la red no suceda. Esto se conoce como el *desvanecimiento del gradiente*. El otro problema es que la función del gradiente de error se vuelva muy grande, causando que los pesos sean actualizados con altos incrementos, causando que el entrenamiento de la red sea inestable. Esto se conoce como el problema de *explosión del gradiente*.Para resolver estos problemas se recurre a :
  - **Pre-Entrenamiento No Supervisado**: el objetivo es de entrenar las redes neuronales en estados sucesivos múltiples. Cada estado involucra una red neuronal poca profunda. La salida de la red neuronal poca profunda se usa como entrada de la siguiente red neuronal. Típicamente, la primera red poco profunda es una red neuronal No Supervisada. Las siguientes son supervisadas.La parte no supervisada se conoce como "Greedy layer-wise Unsupervised Pre-training". En 2006, Geoffrey Hinton demonstró la aplicación de este concepto. Este acercamiento se conoce como "Greedy" porque cada porción de la red neuronal es entrenada independientemente. El término "Layer-wise" se refiere a las capas de la red. En la mayoría de las redes neuronales modernas, el pre-entrenamiento usualmente no es necesario. En su lugar, todas las capas son entrenadas en conjunto usando la propagación hacia atrás. El entrenamiento No Supervisado también facilita

la transferencia de aprendizaje (por medio del uso de algoritmos de aprendizaje de máquina) se almacena el conocimiento ganado de resolver una tarea para resolver otra en forma rápida y con menos datos.

– **Máquinas Restringidas Boltzmann - RBM**: es una red neuronal poca profunda de dos capas. La primera capa es la capa de entrada, y la segunda capa es la capa oculta. Cada nodo está conectado a cada nodo en la otra capa, pero los nodos no están conectados a nodos en la misma capa. Aquí es donde sucede la restricción. Los RBMs pueden desempeñar tareas no supervisadas como *reducciones de dimensionalidad* y *extracción de características* para proveer pre-entrenamiento no supervisado a soluciones de aprendizaje supervisado.
– **"Deep Belief Networks - DBN**: se componen de RBMs conectados. La capa oculta de cada RBM es usada como entrada de la siguiente red RBM y así sucesivamente. Esto permite que las DBNs puedan aprender representaciones más complicadas que muchas veces son usadas como detectores de características.
– **Redes Adversarias Generativas - GAN**: el concepto lo presentó Ian Goodfellow y sus compañeros investigadores en la Universidad de Montreal en 2014. Las GANs pueden usarse para crear datos sintéticos casi reales, como imágenes y habla, o detectar anomalías. En las GANs, se tienen dos redes neuronales. Una red conocida como el *Generador* (genera datos basados en un modelo de distribución de datos creado a partir de muestras de datos reales recibidos). La otra red conocida como *Discriminador*, diferencia entre datos creados por el generador y datos de la verdadera distribución. Las dos redes están asociadas en un juego de suma cero. Figura 14. GANs son algoritmos No Supervisados de Aprendizaje porque el generador puede aprender la estructura subyacente de la verdadera distribución de los datos aunque no se tengan etiquetas. Las GANs aprenden la estructura subyacente en los datos por medio del proceso de entrenamiento y captura eficientemente la estructura usando un número pequeño y manejable de parámetros. El proceso es similar al aprendizaje que ocurre en Aprendizaje Profundo. Cada capa oculta en la red neuronal de un generador captura una representación de los datos subyacentes. Inicia con una estructura simple y sigue con representaciones más complicadas. Por lo tanto, el generador aprende la estructura subyacente de los datos y usando lo que aprendió, el generador intenta crear datos sintéticos que son cercanamente idénticos a la distribución de los datos verdaderos. Los datos sintéticos se parecerán a los reales.
– **Problemas de Datos Secuenciales usando Aprendizaje No Supervisado)**: El *UL* puede manejar datos secuenciales tales como datos en series de tiempo.

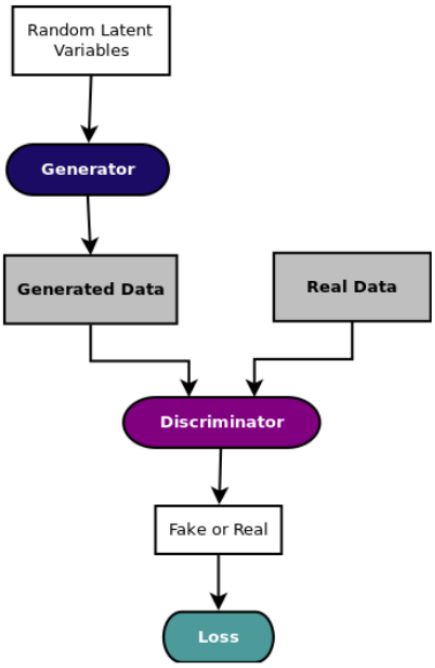

Fig. 14: **Esquema de Redes Adversarias Generativas (GANs)** [2]

Un acercamiento involucra aprender los estados ocultos del modelo de Markov. En un modelo de Markow Simple como en la figura 15, los estados son completamente observados y cambian estocásticamente (aleatoriamente). Estados futuros dependen solamente del estado actual y no dependen de estados anteriores. En un modelo oculto de Markov, los estados sólo son observables parcialmente, pero las salidas son totalmente observables. Figura 16. Los *UL* ayudan a descubrir los estados ocultos (ya que las observaciones son insuficientes). Los algoritmos del Modelo Oculto de Markov involucran el aprendizaje del estado probable siguiente, dado que se conoce la secuencia de estados parcialmente observables y salidas completamente observables. Estos algoritmos se usan en problemas que implican el habla, texto, y series de tiempo.

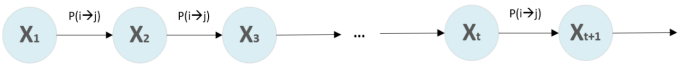

Fig. 15: **Estructura Genérica de una cadena de Markov** [2]

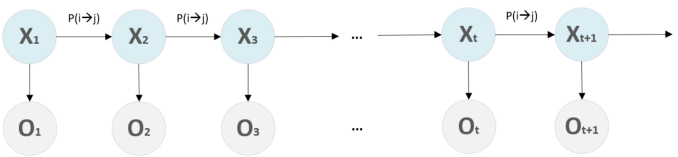

Fig. 16: **Estructura Genérica de un Modelo Oculto de Markov.** [2]

III. Puntos Principales

- **El Aprendizaje No Supervisado** es efectivo en problemas donde los patrones son desconocidos o que cambian constantemente o no hay un conjunto de datos etiquetados en forma abundante.
- **Los retos que enfrenta el Aprendizaje No Supervisado son** El Sobre ajuste, Dimensionalidad y Desplazamiento de los datos.
- **En los algoritmos de Reducción de Dimensionalidad**, se proyectan los datos de entrada originales de alta dimensionalidad a un espacio dimensional menor al filtrar las características no relevantes y por lo tanto mantienen las más relevantes tanto como sea posible.
- **Hay dos tipos de Algoritmos de Dimensionalidad:** *Proyección Lineal y Proyección No lineal*.
- **Los algoritmos de proyección lineal** son *Análisis de Componentes Principales(PCA), Descomposición en Valores Singulares (SVD) y Proyección Aleatoria.*
- **Los algoritmos no lineales o aprendizaje múltiple (manifold)** son el *ISOMAP, Escalamiento multidimensional, Embebido lineal local (LLE), Embebido del vecino estocástico distribuído (t-SNE), Aprendizaje por diccionario, Embebido de árboles aleatorios y Análisis de Componentes Independientes(ICA)*.
- **El Agrupamiento o "Clustering"** consiste en agrupar los datos con base a similitud. El agrupamiento se realiza sin usar ninguna etiqueta, comparando que tan similar son los datos para una observación con respecto a los datos de otras observaciones y grupos.
- **Algoritmo K-means** asignará cada instancia exactamente a uno de los K grupos. Esto optimiza el agrupamiento al minimizar la variación dentro del "cluster" (esto se conoce como *Inercia*) tal que la suma de las variaciones en los diferentes grupos K es la más pequeña posible.
- **Algoritmo de agrupamiento DBSCAN**, tiene su base en la densidad de los puntos. Dada todas las instancias en el espacio, agrupará aquellos puntos cercanos (esto es el número mínimo de instancias que deberían existir dentro de cierta distancia).
- **Algoritmo de agrupamiento HDBSCAN**, es el DBSCAN jerárquico. Convierte el algoritmo DBSCAN a un algoritmo de agrupamiento jerárquico. En otras palabras, los grupos se crean con base a densidad y después los une iterativamente con base a la distancia.
- **Aprendizaje Profundo No Supervisado** se fundamenta en *Extracción de Características, Pre-Entrenamiento No Supervisado, Máquinas Restringidas Boltzmann - RBM, Deep Belief Networks - DBN y Redes Adversarias Generativas - GAN*.
- **Problemas de Datos Secuenciales usando Aprendizaje No Supervisado** pueden tener un acercamiento en aprender los estados ocultos del modelo de Markov.

Bibliografía

[1] A. A. Patel, *Hands-On Unsupervised Learning Using Python: How to Build Applied Machine Learning Solutions from Unlabeled Data*. O'Reilly Media, 2019.

[2] G. Bonaccorso, A. Fandango, and R. Shanmugamani, *Python: Advanced Guide to Artificial Intelligence: Expert machine learning systems and intelligent agents using Python*. Packt Publishing Ltd, 2018.

[3] Guru99, "Unsupervised machine learning: What is, algorithms, example," 2020, accesado el 4 de Julio del 2020. [Online]. Available: https://www.guru99.com/unsupervised-machine-learning.html#10

[4] upgrad.com, "Top 9 python libraries for machine learning in 2020," 2020, accesado el 4 de Julio del 2020. [Online]. Available: https://www.upgrad.com/blog/top-python-libraries-for-machine-learning/

[5] NumPy-org, "Numpy library." 2020, accesado el 4 de Julio del 2020. [Online]. Available: https://numpy.org/

[6] ScyPy-org, "Scipy library." 2020, accesado el 4 de Julio del 2020. [Online]. Available: https://www.scipy.org/

[7] Scikit-learn, "Scikit-learn library." 2020, accesado el 4 de Julio del 2020. [Online]. Available: https://scikit-learn.org/stable/

[8] Lisa-lab., "Theano library." 2020, accesado el 4 de Julio del 2020. [Online]. Available: http://deeplearning.net/software/theano/

[9] TensorFlow-org., "Tensorflow library." 2020, accesado el 4 de Julio del 2020. [Online]. Available: https://www.tensorflow.org/

[10] Keras-org, "Keras library." 2020, accesado el 4 de Julio del 2020. [Online]. Available: https://keras.io/

[11] PyTorch-org, "Pytorch library." 2020, accesado el 4 de Julio del 2020. [Online]. Available: https://pytorch.org/

[12] R.Collobert-S.Bengio-J.Mariethoz, "Torch library." 2020, accesado el 4 de Julio del 2020. [Online]. Available: http://torch.ch/

[13] Pandas-org., "Pandas library." 2020, accesado el 4 de Julio del 2020. [Online]. Available: https://pandas.pydata.org/

[14] J.Hunter-D.Dale-E.Firing-M.Droettboom, "Matplotlib library." 2020, accesado el 4 de Julio del 2020. [Online]. Available: https://matplotlib.org/index.html