Machine Learning Projects

Carlos Alejandro Cornielle Guzmán

Ing. Mecatrónico

Santo Domingo, RD

Abstract— **Para este documento se desarrollan distintas prácticas donde se analizarán y ejecutarán distintos tipos de algoritmos de Machine Learning como, Bayes, K- NN, etc... Hasta aplicar redes neuronales, así como métodos para simplificar la cantidad de información tanto para reducir la cantidad de muestras de la data como métodos para reducir la cantidad de features, todo esto con sus respectivas comparaciones y análisis a partir del set de datos original.**

I. Introducción

Hoy en día interactuamos de manera continua con algoritmos de machine learning, esta inteligencia que se les otorga a las máquina gracias de su poder de procesamiento y de memoria, estos métodos y algoritmos están detrás de toda la gran de cantidad de sugerencias y recomendaciones que nos proporcionan los dispositivos inteligentes, un ejemplo muy común pero acertado es el caso de Netflix donde a partir de las series que ya has visto y otros parámetros como si termina de verla o cuantas series y películas relacionadas a ese género has visto, logran de una manera certera recomendar series y películas si son del gusto de la persona, esta acción causó una revolución en este mundo de las películas y las series en directo, ya que las personas encontraron una aplicación donde encontrar series y películas que les guste es tan sencillo como entrar y ver las recomendaciones. No obstante aplicaciones como máquinas para resolver cubos Rubik, detectar si un correo es spam o no, los carros autónomos, la compresión de las imágenes, etc… es realizado mediante distintos tipos de aprendizaje de machine learning [1].

Pero ¿Qué es machine learning? Esto puede resumirse de manera muy compacta como "*La capacidad de las máquinas para aprender a partir de los datos"* [1]. Este como se mencionó anteriormente solemos verlo en recomendaciones de películas, renacimiento de voz de asistencia virtual, coches autónomos, este incluso se utiliza para diagnósticos médicos y detección de fraude con tarjetas de débito o crédito. Un punto importante es que la base fundamental del machine learning es la estadística [1]. Los tres grandes tipos o corrientes del machine learning que se emplean en la actualidad son:

- **Aprendizaje no supervisado:** Las distintas máquinas y computadoras no nos capaces de detectar los distintos patrones que se presentan incrustados en las distintas bases de datos que se encuentran clasificadas, esto se debe que no aprenden de la data por lo que se basan principalmente en encontrar similitudes. En este caso, los algoritmos no están programados para detectar un tipo específico de datos, como sucedió en las imágenes de perros, pero buscan ejemplos similares y se pueden agrupar. Esto es lo que sucede, por ejemplo, en el caso del reconocimiento facial, donde el algoritmo no busca características específicas, sino más bien un conjunto de patrones comunes que se dice que son la misma cara. Esto se puede ver en aplicaciones como regresión y clasificación [2].

- **Aprendizaje supervisado:** Consiste en entrenar a las máquinas con datos etiquetados. Por ejemplo, fotos con descripciones de los elementos que aparecen en ellas. Se utilizan distintos algoritmos capaces de seleccionar las etiquetas deseadas en otras bases de datos que no se haya utilizado para entrenar con el algoritmo. De esta manera si se ha etiquetado un grupo de imágenes en las que se muestran distintas imágenes de perros, la máquina debe ser capaz de aprender mediante un algoritmo a distinguir cuando en una imagen bajo esas características haya presencia de perros y poder identificarlos, así como poder identificar imágenes similares. Se observa en aplicaciones como análisis de grupos (clusters) y reducción de dimensionalidad [2].

- **Aprendizaje por refuerzo:** Consiste en la acción mediante la que una máquina aprende por mediante por la clásica técnica de prueba y error hasta lograr encontrar la mejor manera de realizar la tarea planteada. Un ejemplo clásico es, "*Microsoft el cual utiliza esta técnica en distintos entornos de juego como el famoso conocido Minecraft para ver cómo los distintos puntos del juego mejoran su desempeño. esto mediante de ella el sistema aprende a modificar su conducta a base de "recompensas" para que resuelva la tarea asignada, sin programarlo para esta en específico"* [2].

Otra corriente que suele utilizarse para aplicaciones tanto de aprendizaje supervisado como no supervisado son las redes neuronales. Este funciona bajo una idea muy simplificada de las neuronas del cerebro. Un conjunto de redes neuronales que tienen como objetivo resolver problemas difíciles de solucionar mediante algoritmos convencionales, a pesar de que son algoritmos complejos, son capaces de solucionar tantos los problemas convencionales como los complejos, pero se suelen utilizar cuando los métodos convencionales tienen complicaciones.

II. Clasificación por Bayes (1 feature)

A. *Problema:*

Esta práctica consiste en clasificar dos tipos de habichuelas, para este caso se plantearon los tipos Jacomelo y Pinta, a partir de estos tipos de habichuela se extrajo un feature de las imágenes y a partir de este feature se entrenó el algoritmo de Bayes para que en base a las probabilidades determinadas que tipo de habichuela se estaba en un set que este aún no había evaluado.

El objetivo general de esta práctica es el de clasificar dos tipos de habichuelas, precisamente habichuelas del tipo Jacomelo y Pinta mediante el algoritmo de Bayes utilizando una característica extraída de las imágenes.

B. *Método*

La práctica pide aplicar el método de Bayes como el algoritmo a utilizar para realizar las predicciones, el primer paso realizado para esta práctica fue el de identificar con que fondo para las imágenes se lograba obtener la mayor información de las habichuelas, varios intentos realizados después se concluyen que el fondo negro con un número de ISO bastante bajo para que el sensor de la cámara no detecte tanta luz como la existente al momento de realizar las fotografías.

Las fotografías fueron tomadas en una habitación de luz tenue, la luz fue reflejada de manera indirecta con lámparas, para evitar saturación de la luz en las imágenes que pueda causar pérdida de información en esta.

El próximo paso consistió en como extraer el feature a partir de las imágenes, el método que se utilizó fue utilizar dos imágenes que componían el set de entrenamiento para el algoritmo de Bayes y otra imagen no utilizada para entrenar el algoritmo para utilizarse como set de pruebas.

El feature extraído es la varianza de los pixeles de las imágenes, esto fue realizado a partir de la función de Matlab *Regionprops* que permitió extraer información de las imágenes después de realizar una conversión de estas de RGB a escalas de grises, a partir de la escala de grises las imágenes fueron convertidas a binarias para facilitar la extracción de ciertas informaciones como el área, tamaño y poder delimitar donde se encuentra cada habichuela. Una conversión de escala de grises a binaria se hace mediante un umbral o frontera que puede delimitarse para cada imagen, esto ayudó para a partir de cierto grado en la escala de grises se tomaba una decisión de que se debe considerar como blanco y que se debía considerar como negro, con esto se logró observar que por más intentos realizados el fondo siempre reflejaba una mínima luz por lo que se aplicó un límite para determinar que es parte del fondo y convertirlo en un *0* (negro) y que pixeles no componían el fondo para mantener su valor igual, se colocó una frontera donde cualquier pixel en la escala de grises que tenga un valor de (Este valor refleja la intensidad de luz de los pixeles o también puede indicar que tan cercano al blanco o al negro esta cada pixel). El valor utilizado fue de cien según lo observado en los pixeles de cada habichuela los pixeles reflejaban valores en la escala de grises que superaban en gran escala este número mientras que los pixeles del fondo reflejaban valores menores a este.

Se eliminó gran cantidad del ruido con ayuda de la función que se mencionó anteriormente *Rengionprops* ya que esta también detecta figuras sumamente pequeñas y a partir de una condición donde se elimine de la matriz todo lo que tenga un área menor como el área más pequeña de las habichuelas.

Ya logrado que se detecte cada una de las habichuelas, eliminado el ruido, se utilizó una función de Matlab llamada “Cropsize” que permitió extraer cada habichuela de la imagen original sin alterarse para a partir de aquí poder sacar cualquier feature.

El feature extraído fue la varianza de los pixeles en la escala de grises, la idea general de esto es poder reflejar que tan distantes pueden estar estos valores para cada tipo de habichuela. La varianza de las imágenes suele utilizarse en otras aplicaciones para poder determinar detalles o bordes para las imágenes, por lo que consideré que este sería un buen feature para comparar dos objetos tan parecidos como dos tipos de habichuela.

La función discriminante utilizada fue la de comparar las densidades probabilísticas de cada habichuela de que esta sea Jacomelo o sea Pinta, a partir de esta comparación el valor mayor fue el que se asoció para la clase de la habichuela, si la densidad probabilista se ser Jacomelo de la habichuela que se ubica en la primera columna y la primera fila es mayor a la densidad probabilística de ser pinta, se indicó que esta pertenece a la clase de las Jacomelo, esta comparación se realizó para cada habichuela.

C. *Análisis:*

Los datos obtenidos en esta práctica serán presentados a partir de cómo se desarrolló el método, punto explicado en el apartado anterior.

Los histogramas de cada clase y las campanas gaussianas correspondientes a estas se presentan como:

Fig. 1 Histogramas pertenecientes a cada tipo de habichuela y la gráfica de las campanas gaussianas estimadas

La distribución que modela la data extraída de cada tipo de habichuela es representada por los histogramas y la campana gaussiana que estos estiman, con esto se consiguió la información necesaria para conocer que las distribuciones son normales, por lo que se puede aplicar la función de *Normpdf* de Matlab para conseguir las distribuciones gaussianas, esto se debe realizar debido a que estos features son continuos.

Fig. 2 Histogramas de las varianzas de las habichuelas una al lado de otra.

Después de observar los histogramas presentados en la Fig. 2 juntos se comprendió que este puede ser un buen feature para poder clasificar estas habichuelas debido a que los histogramas no se encuentran muy solapados, por lo que cada habichuela podría ser bien clasificada a partir de este.

Con la función de Matlab *Regionprops* el primer resultado fue poder detectar cada una de las habichuelas en cada imagen del set de entrenamiento:

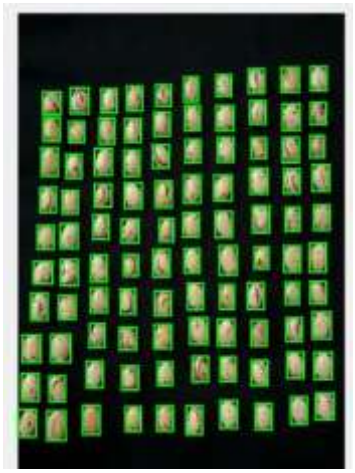

Fig. 3 Fotografía del set de entramiento para las habichuelas del tipo jacomelo detectando las 100 en la imagen

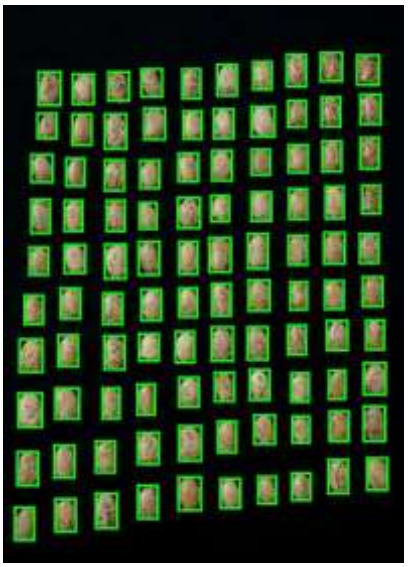

Fig. 4 Fotografía del set de entramiento para las habichuelas del tipo pinta detectando las 100 en la imagen

Los resultados obtenidos y que se pueden observar en la Fig. 4 que se logró reducir bastante la cantidad de ruido que suelen presentarse en las imágenes, así como se obtuvo una gran precisión al momento de encerrar cada habichuela por cada cuadro para indicar que se conocen las coordenadas y otras propiedades de estas.

Alcanzó este punto para determinar las densidades gaussianas debido a que se está utilizando un feature continuo, se utilizó el comando de Matlab *Normpdf*, para esta se obtiene la media de la varianza obtenida de cada clase (Cada tipo de habichuela) y se obtiene la desviación estándar igualmente de la varianza obtenida de clase. Mediante este comando se reduce la siguiente fórmula de la Fig. 5:

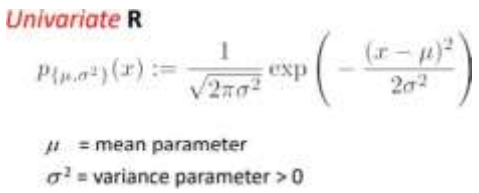

Fig. 5 Función de las densidades probabilísticas a partir de una distribución normal [2].

Estas densidades son donde el algoritmo de Bayes aprende de la data del set de entrenamiento, guardando estas densidades probabilísticas. El principio básico de como función el algoritmo de Bayes es:

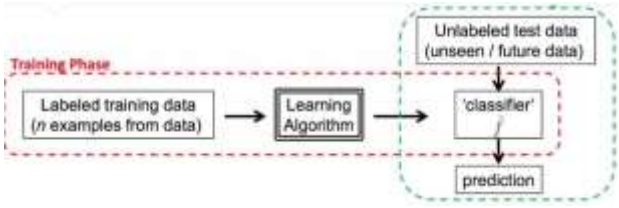

Fig. 6 Principio de funcionamiento del algoritmo de Bayes [3].

Las fotos originales tomadas organizadas en un arreglo de diez filas con diez columnas para cada uno de los casos encontrados en la Fig. 7, Fig. 8 y Fig. 9:

Fig. 7 Set de entrenamiento para las habichuelas jacomelo

Imagen del set de entrenamiento habichuelas Pinta tal como se observa en la Fig. 8.

Fig. 8 Set de entrenamiento para las habichuelas pintas.

Imagen set de pruebas de las habichuelas presentada en la Fig. 9.

Fig. 9 Set de pruebas, se incluyen ambos tipos de habichuelas.

Llegado el punto después de determinar las densidades probabilísticas e implementado el algoritmo Bayes, para poder hacer comparaciones primero se utilizó el set de entrenamiento para ver que como fue su funcionamiento y como el modelo a partir de este podría ajustarse a la misma data utilizada para el entrenamiento. En la Fig. 10 se observan los primeros resultados utilizando el set de entrenamiento.

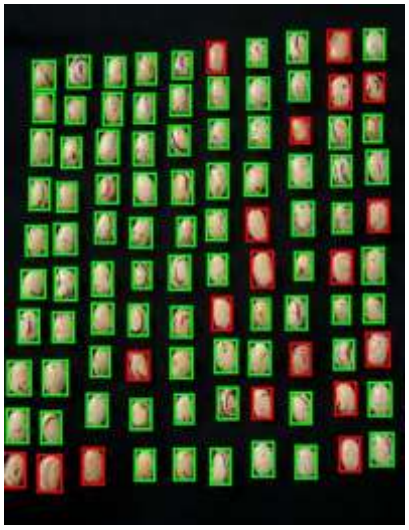

Fig. 10 Clasificación con el algoritmo de Bayes con el set de entrenamiento de la clase jacomelo

La tasa de aciertos que se obtuvo a partir de intentar clasificar la misma imagen utilizada para entrenar el algoritmo fue 81%, tal como se puede observar en la Fig. 10, donde los cuadros verdes muestran la clase Jacomelo y los cuadros rojos los que el algoritmo indicó que pertenecían a la clase pinta.

Esta misma prueba se realizó para el set de entrenamiento de la clase pinta como se observa en Fig. 11:

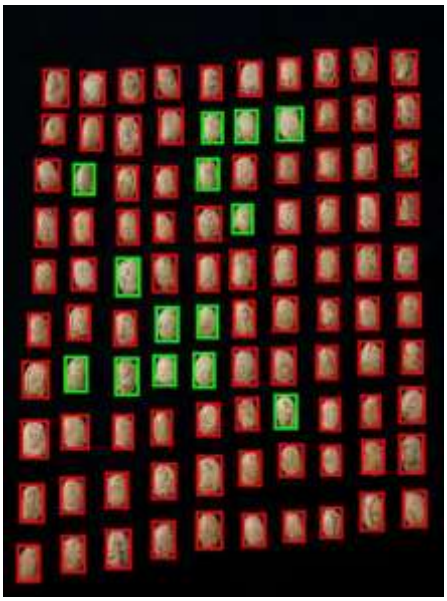

Fig. 11 Clasificación del set de entrenamiento de las habichuelas pinta.

La tasa de aciertos obtenida a partir de intentar clasificar la misma fotografía que la utilizada en para entrenar el algoritmo para la clase de la pinta fue de 86%, más adelante se observará que tendencias presenta el algoritmo y que conclusiones se pueden obtener a partir de estas dos clasificaciones.

El set de pruebas compuesto por ambos tipos de habichuelas presentó el siguiente resultado presentado en la Fig. 12:

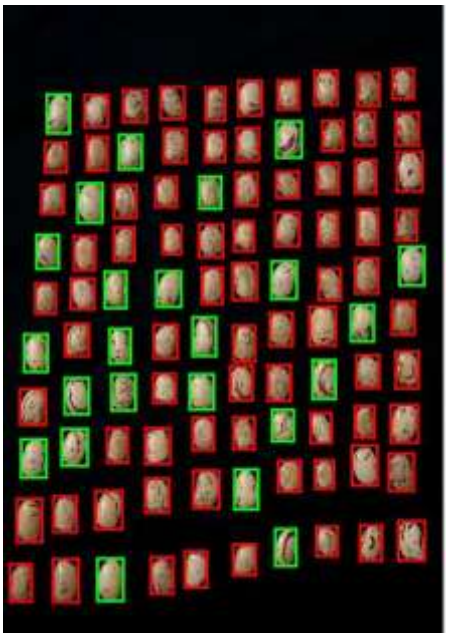

Fig. 12 Clasificación del set de pruebas, cuadro rojo considerado como pinta y cuadro verde como jacomelo

La tasa de aciertos obtenida fue de 81% y que se puede observar en la Fig. 12, resultado muy similar en comparación con la clasificación con el set de entrenamiento de las habichuelas Jacomelo. Las tasas de aciertos son muy parecidas, pero después de estudiar el modelo que resultó del algoritmo de Bayes se concluye que este tendía a predecir las habichuelas como pintas, al estudiar los histogramas de la Fig. 2. El histograma de las Jacomelo tiene más proporción de su gráfica compartida con las Pinta que la proporción que las Pintas tienen de su gráfica y debido a que la dispersión de estos histogramas es menor, dando densidades probabilísticas más acordes a la realidad de esta clase. Esas proporciones compartidas representarán densidades probabilísticas donde habrá una alta incertidumbre para poder realizar una simple comparación. A pesar de todo esto, el algoritmo logró clasificar con una tasa de aciertos mayor a un 80% cada tipo de habichuelas, pero esta tendencia y proporciones de gráfica compartida son las causaste los falsos positivos y verdaderos negativos.

En la Tabla 1 se presenta la matriz de confusión, en esta se recalca la tasa de aciertos obtenida para el set de pruebas, donde el resultado fue un 81%, pero como se mencionó en el apartado anterior el modelo tiene una tendencia reconocer mejor la clase de las pintas, esto se debe a que la varianza de sus pixeles no presentan una desviación estándar tan grande como la que presentan las Jacomelo, por lo que facilita al algoritmo aprenderse mejor la información de esta, debido a que un rango más pequeño de densidades gaussianas define esta clase, mientras un rango de densidades gaussianas más amplias definen la clase de las habichuelas Jacomelo.

TABLA I
MATRIZ DE CONFUSIÓN

Matriz de Confusión			
		Estimación	
		Positivo	Negativo
Realidad	Positivo	66	7
	Negativo	15	12
Tasa de aciertos		81%	

III. CLASIFICACIÓN POR BAYES (MULTIVARIABLE)

A. Problema

El problema para esta práctica es el mismo que para la práctica anterior de Bayes, consiste en clasificar dos tipos de habichuelas, para este caso se plantearon los tipos Jacomelo y Pinta, a partir de estos tipos de habichuela se extrajeron dos features de las imágenes y a partir de estos features se entrenó el algoritmo de Bayes para que en base a las probabilidades determinadas para cada tipo de habichuela según los features poder estudiar un set de pruebas no utilizado para entrenar el algoritmo y observar si este es capaz de clasificarlos o que tasa de aciertos obtiene en este caso.

El objetivo de la práctica al igual que la práctica anterior es el de clasificar dos clases de habichuelas, las Jacomelo y las Pinta, pero esta vez a partir de dos features extraídos de las imágenes.

B. Método

Para el mismo problema del punto anterior, se decide probar con otro feature extraído de las imágenes, se utilizó el feature de la media de los pixeles en conjunto del feature seleccionado en el apartado anterior de la varianza de los pixeles de la imagen en la escala de grises, estos features podrían visualizarse de manera práctica como, la varianza de los pixeles sería algo parecido al concepto de contraste, que indica la diferencia relativa en la intensidad entre distintos puntos de una imagen, como hace referencia a diferencia relativa si dos puntos o superficies en una imagen presentan igual brillo el contraste será nulo, es importante indicar que en casos donde la luminosidad de una superficie y un objeto es la misma si se está trabajando en la escala de grises no conseguirá un distinción entre el objeto examinado y la superficie, para facilitar este concepto de contraste esta la siguiente imagen:

Fig. 13 El cuadro de la derecha tiene más contraste que el objeto de la izquierda [2]

Con la representación de la imagen 1 es fácil hacer la relación de que la varianza de los pixeles en esta escala de grises indica que tan dispersos están los valores de intensidad de la luz de cada clase de habichuela por lo que en el próximo apartado se observará porque se consideró como buen feature a utilizar.

El otro feature utilizado fue la media de los pixeles de la imagen en la escala de grises, la media en la escala de grises irá más relacionado al color de cada una de las habichuelas, para de esta manera aprovechar que ambas tienen distintas tonalidades de color, así como franjas de color distintos, las Jacomelo presentan franjas rojizas y cercanas al color marrón, las Pintas presentan manchas y franjas algunas verdosas y otras un color tenue de amarrillo oscuro. La siguiente imagen que se observa en la Fig.14 puede ayudar a comprender mejor esta relación:

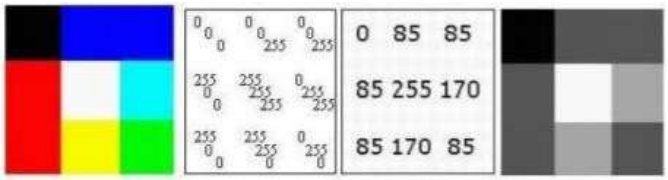

Fig. 14 Relación de los colores de una imagen, valores matriz RGB [3]

Tal y como la Fig. 14 lo representa al final la tonalidad y color de los objetos representará un número distinto en la matriz representada por la escala grises, se aprovecho está propiedad para mediante la media obtener una característica de la cual se observarán datos más exactos en el apartado de análisis.

Los métodos aplicados fueron exactamente los mismos que los utilizados en la práctica I, excluyendo que para las probabilidades del teorema de Bayes se utilizó otra fórmula que ayuda a determinar las densidades gaussianas cuando se tiene más de un solo feature. La fórmula presenta en el Fig. 15 utilizada para obtener esas densidades gaussianas y posteriormente las probabilidades fue:

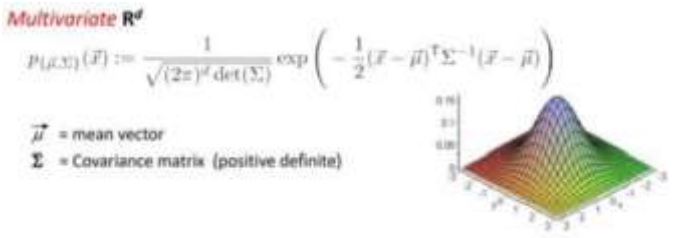

Fig.15 Formula para las densidades gaussianas multivariables [4]

La fórmula presenta en la Fig. 15 es la versión de distintas dimensiones que la aplicada en la práctica I que se resumía a un utilizar el comando Normpdf pero que realmente lo que está función realiza es la versión para un solo feature de la formula presentada en Fig. 15. Cabe considerar que ambas formulas se obtienen considerando MLE para las clases, dando buenos resultados para las aproximaciones mediante el teorema de Bayes.

Un punto importante es que en la práctica se indica que se considere los priors (estos representan la probabilidad general de que resulte ser cualquiera de las clases) de las clases como iguales por lo que al ser un factor común no se aplicó para la función discriminante (Se probó a utilizarse, pero los resultados de la clasificación eran los mismos por el motivo anteriormente mencionado).

En cuanto a la utilización de MLE (máxima verosimilitud), el fundamento de este método consiste en considerar la independencia de clases con otro para encontrar la función de densidad de las muestras (la mostrada en la Fig. 15). Este método se utilizó al obtener la media y la matriz de covarianza de cada clase y a partir de estos datos obtener las densidades probabilísticas.

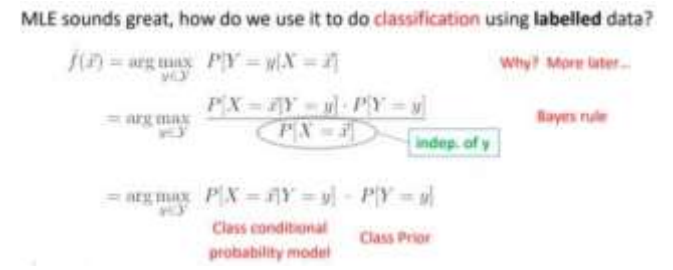

Fig. 16 Máxima verosimilitud, demostración de la independencia de clases [5]

Tal y como se muestra en la Fig. 16, donde indica como se utiliza la máxima verosimilitud en el teorema de Bayes, es como se explicó anteriormente, considerando la independencia entre las clases, para posteriormente obtener las densidades probabilísticas.

Las densidades probabilísticas de cada clase de deben dividir entre la sumatoria de ambas para obtener valores de probabilidad, esto facilitará la aplicación una función discriminante.

Después de utilizar estos métodos y algoritmos se debe aplicar una función discriminante, capaz de clasificar cada uno de los elementos que se examinen a partir de los datos obtenidos, se utilizará la misma implementada en la práctica I donde se realizaba una comparación directa de la probabilidad de pertenecer a una clase o a otra, también con esta información se obtiene la tasa de aciertos del algoritmo.

La tasa de aciertos será la métrica de error implementada, ya que solo es determinar si el algoritmo clasifico bien o mal cada elemento, para esto se tendrá una matriz con las clases a las que pertenece verdaderamente cada una de las muestras, es la misma métrica de error implementada a en la práctica I.

En cuanto a los features extraídos se presentarán los histogramas y estimaciones de las campanas gaussianas.

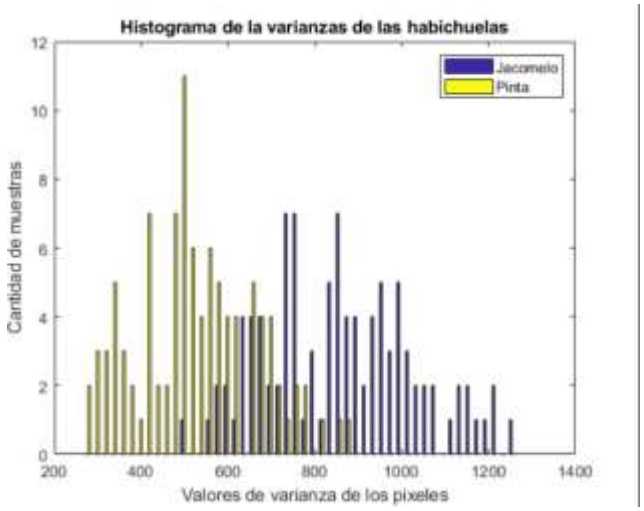

Fig. 17 Histogramas de la varianza de los pixeles de ambas clases

La varianza de los pixeles presentado en la Fig. 17 para ambas clases resultó tener rango de valores ligeramente alejados por lo que solo una pequeña proporción de sus histogramas se encuentran solapados, tal como a simple inspección se puede observar en estos histogramas, por lo que las muestras que presenten valores de varianza dentro de aproximadamente el rango de 650 a 750 difícilmente serán clasificados de manera correcta esto dependerá de cuál sea su valor de varianza en específico, pero la idea es que existe cierta incertidumbre para la estimación cuando los rangos son iguales, debido a la función discriminante creada para estimar la clasificación más adelante se podrá observar que existe una ligera tendencia a pertenecer a una clase en específico para estos casos.

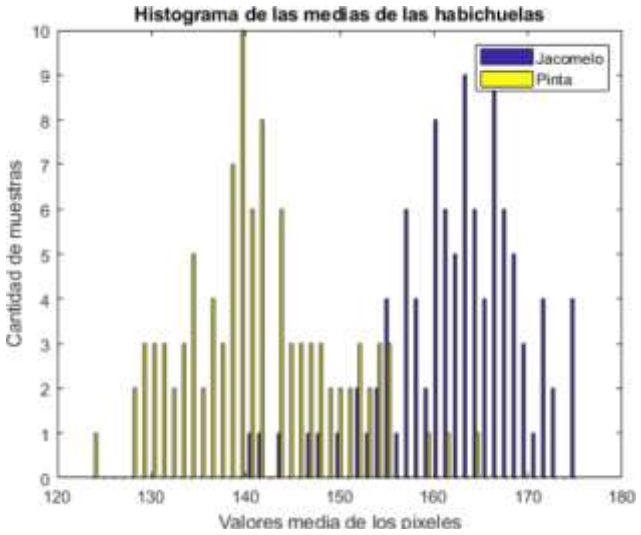

Fig. 18 Histogramas de la media de los pixeles de ambas clases

El caso de la Fig. 18 que muestra el histograma del segundo feature utilizado, la media de los pixeles de ambas clases, estos histogramas no presentan una proporción de estos tan grandes solapadas como la que se puede apreciar en la Fig. 19, por esta razón y que los valores de cada clase entre ellos no se encuentran muy dispersos, la media es considerado aún un mejor feature que la varianza para este caso de aplicación.

Mientras que un caso de un feature que no se deba considerar para realizar el entrenamiento del algoritmo a partir del cual se estimará la clasificación es el siguiente presentado en la Fig. 19:

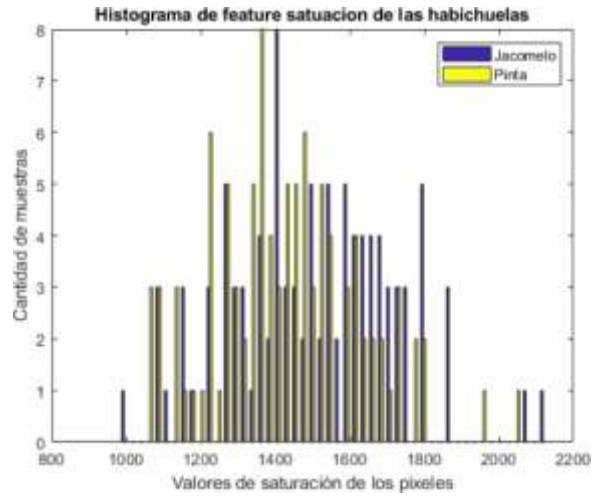

Fig. 19 Histogramas saturación de color de los pixeles de ambas clases

Una característica importante que se destacó tanto para la Fig. 17 como la Fig. 18 es que los valores del feature utilizado no podía solaparte, esto se debe a que al momento de entrenar un algoritmo para clasificar a partir de esta información no se obtendrá un buen modelo que pueda realizarlo, tendrá bastante incertidumbre en su información o puede darse el caso de que estime que todas las muestras pertenecen a una misma clase. Este es el ejemplo de un feature que no debe utilizarse para este caso, estos valores son extraídos de las imágenes utilizadas por lo que para otros casos este feature si puede ser de gran aporte para realizar la estimación.

Mientras que las campanas gaussianas estimadas fueron:

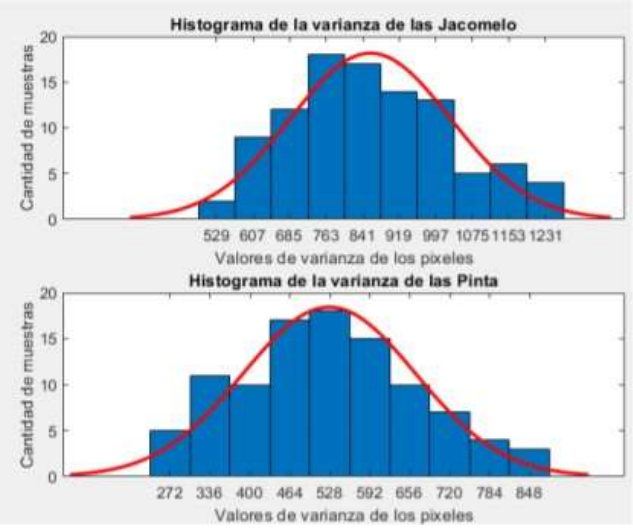

Fig. 20 Campanas gaussianas estimadas a partir de la varianza.

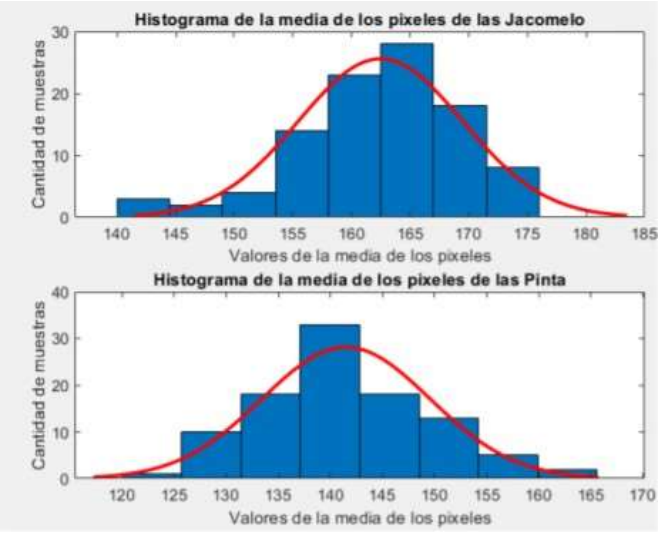

Fig. 21 Campanas gaussianas estimadas a partir de la media

Las campanas estimadas presentadas en la Fig. 19 para el feature de la varianza de los pixeles y la Fig. 20 para la media de los pixeles, esto para clase a partir de la data obtenida de la varianza y la media de los pixeles no representan a 100% una distribución gaussiana, pero a pesar de esto, tal y como se puede ver en la imagen se puede conseguir una buena estimación asumiendo que, si tienen distribuciones gaussianas, la mayor parte de los datos siguen el comportamientos de las campanas.

C. Análisis:

Se verifica que con este método de utilizar dos features se obtuvieron resultados similares, pero con un grado de mejoría en comparación a los obtenidas en la práctica I cuando se intentó clasificar las distintas muestras con únicamente un feature. Tal y como se mencionó en el apartado de método de obtuvieron las densidades probabilísticas de que cada muestra pertenezca a cada clase, posteriormente llevando estos datos a valores de probabilidad se observó cual es mayor y con esto se estima a la clase que pertenece cada muestra, el sesgo que presenta el algoritmo se debe a como se realiza la comparación donde si la probabilidad de ser Pinta es menor o igual a la probabilidad de que la muestra sea Jacomelo, el algoritmo que se realizó estima esta muestra como Pinta, por lo que para el caso en que las probabilidades sean iguales se estima como que pertenece a clase de Pintas, esto crea un pequeño sesgo en el algoritmo que es importante destacar, sin embargo, esto no presente mayores problemas al momento de realizar las clasificaciones donde siempre se obtuvieron tasas de aciertos mayores al 80%.

Al igual que como se realizó en la práctica I se presentarán las tasas de aciertos y la muestra de la estimación de la clasificación que realizó el algoritmo tanto para las imágenes del set de pruebas como para las imágenes del set de entrenamiento, comenzando con el set de entrenamiento de las habichuelas Jacomelo en la Fig. 22.

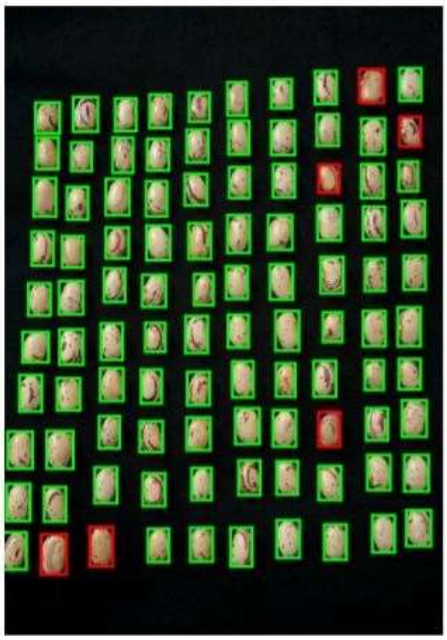

Fig. 22 Clasificación set de entrenamiento clase Jacomelo, dos

La tasa de aciertos obtenida con el set de entrenamiento de las habichuelas Jacomelo tal como se observa en la Fig. 22 fue de 94%, una mejoría considerable observando que utilizando solo la varianza de los pixeles en la práctica I se obtuvo una tasa de aciertos de 81%, por lo que tal y como se explicó al momento de analizar la Fig. 22 la media de los pixeles de cada clase es un buen feature para entrenar el algoritmo y realizar las estimaciones entre cada tipo de habichuela.

Para el caso del set de entrenamiento para las habichuelas Pinta se obtuvieron los siguientes resultados presentados en la Fig. 23:

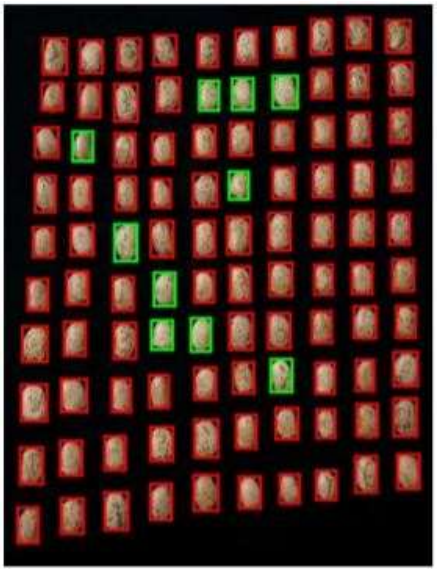

Fig. 23 Clasificación set de entrenamiento clase Pinta, dos features.

La tasa de aciertos de la imagen presentada en la Fig. 23 obtenida a partir de clasificar el set de entrenamiento de las habichuelas Pinta a partir de dos features fue de 90%, observando una mejoría al igual que con el set de entrenamiento de las Jacomelo. La tasa de aciertos obtenida utilizando únicamente el feature de varianza para de los pixeles fue de 86%, mostrando que de igual forma la tasa de aciertos aumentó.

El resultado para el set de pruebas se observa en la Fig. 24, estos fueron:

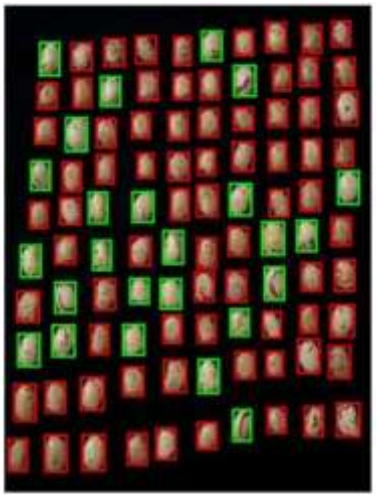

Fig. 24 Clasificación set de pruebas, dos features

La Fig. 24 muestra la estimación de la clasificación lograda a partir del teorema de Bayes utilizando los dos features, explicados anteriormente, la varianza de los pixeles y la media de estos. Al igual que como paso con los sets de entrenamiento, en el set de prueba se obtuvo una mejoría de la estimación para clasificar cada muestra alcanzando una tasa de aciertos de 84%, un 3% mayor que la obtenida a partir de utilizar la varianza como único feature en la práctica I.

De esta manera se verifica que la media de los pixeles si fue un buen feature a implementar ya que mejoró la tasa de aciertos que originalmente se obtenía con un solo feature, los porcentajes de mejoría no son tan elevados por lo que dependiendo de la aplicación que se le dé al algoritmo, con que método se realiza la clasificación y que tanto se puede permitir este extra de procesamiento dependiendo del equipo en donde se vaya a implementar, al igual que la precisión que se esté buscando, agregar este feature o no puede ser de gran ayuda.

Se presenta la Tabla 2 que consiste en la siguiente matriz de confusión:

Tabla II

MATRIZ DE CONFUSIÓN

Matriz de Confusión			
		Estimación	
		Positivo	Negativo
Realidad	Positivo	58	9
	Negativo	7	26
Tasa de aciertos		84%	

Como se puede observar en la matriz de confusión de la Tabla 2 al utilizar dos features se mejoró la cantidad de casos negativos que realmente son negativos, para entender un poco la data que arroja esta matriz es como si las estimaciones positivas fueran Jacomelo y las negativas fueran Pintas, al observar las estimaciones positivas y que realmente eran positivas es como observar en el algoritmo las estimaciones de las muestras que pertenece a la clase de las Jacomelo y que verdaderamente pertenecen a esta clase, por lo que si se estima como Pinta y realmente era Jacomelo o viceversa se presentará como error y disminuirá la tasa de aciertos, a esto se debe que los positivos negativos y los negativos positivos sean la sumatoria de las estimaciones erróneas del algoritmo, mientras que los positivos ciertos y los falsos negativos suman las estimaciones correctas del algoritmo. Para observar algo parecido a lo que sucede con las Fig. 18 y Fig. 19 se grafican las distribuciones de cada clase en 3D para observar cómo estos valores varían en el espacio, esto se observa en las Fig. 25 y Fig. 26 respectivamente.

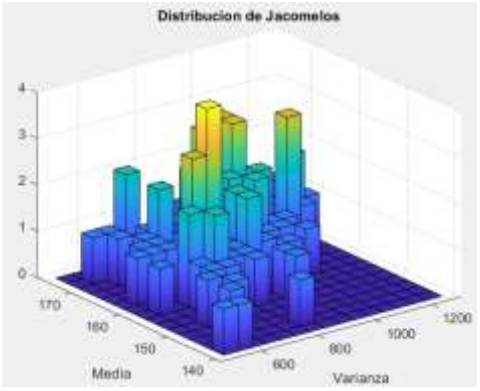

Fig. 25 Distribución en 3D de la clase jacomelo

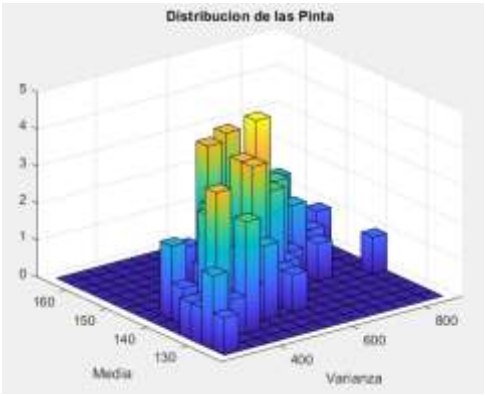

Fig. 26 Distribución en 3D de la clase pinta

IV. NEAREST NEIGHBOR CON REDUCCIÓN DE LA BASE

A. Problema:

Se trabaja con el set de datos de mpg el cual contiene ocho features entre ellos continuos, discretos y uno que corresponde al nombre del carro (Este no fue utilizado debido a que es un valor único para cada una de las muestras del set) que contienen la siguiente información presentada en la Fig. 27:

Attribute Information:

1. mpg: continuous
2. cylinders: multi-valued discrete
3. displacement: continuous
4. horsepower: continuous
5. weight: continuous
6. acceleration: continuous
7. model year: multi-valued discrete
8. origin: multi-valued discrete
9. car name: string (unique for each instance)

Fig. 27 Features de la data set de mpg, donde mpg es la variable dependiente a clasifica [3]

La práctica requería hacer una separación de la columna mpg del set de datos en diez clases (Considerando que se debían tener diez rangos igualmente espaciados) esto, para poder realizar en otros pasos la clasificación de estas clases en la que se separó la columna mpg de la data set original a partir de los siete features restantes.

Estos features tienen distintos tipos de variables algunos de estos son discretos y otros continuos.

Se requiere reducir la cantidad de datos del training set para luego compararse con otro set de datos que no se haya utilizado para entrenar el algoritmo. Esto debe realizarse varias veces organizando la data aleatoriamente en distintas ocasiones (Probar con distintas permutaciones de la data) al igual que probar que sucede cuando se reducir la cantidad de datos del set de entrenamiento.

B. Método

El algoritmo utilizado para estudiar la data fue K-NN (K-Nearest Neighbor) este es un algoritmo del tipo supervisado, donde se utiliza algún tipo de medición, en este caso la distancia euclideana, está distancia se debe obtener de cada punto del set de pruebas con todo el set de entrenamiento para poder observar la cantidad de K puntos más cercanos, con esta información de la cantidad de puntos más cercanos se hace una votación para poder clasificar entre las distintas clases. Tal y como se representa en la Fig. 26:

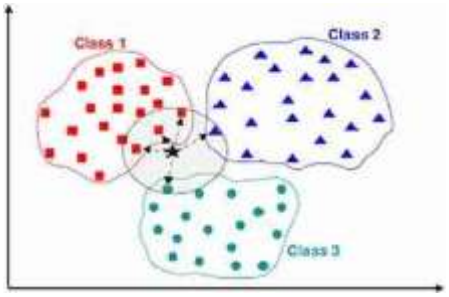

Fig. 28 Muestra del funcionamiento y de cómo funciona la votación para la clasificación [5]

Se aplicó Nearest Neighbor sobre la base, utilizando como medida de distancia la distancia euclideana y aplicando una reducción de la base usando la metodología de Condensed Nearest Neighbor, esta consiste en buscar un subset de datos de la data perteneciente al set de training, el principio de esta metodología se presenta en la Fig. 29 obtenida de Alpaydin (2004) es la siguiente:

$Z \leftarrow \emptyset$
Repeat
  For all $x \in X$ (in random order)
    Find $x' \in Z$ such that $\|x - x'\| = \min_{x^j \in Z} \|x - x^j\|$
    If class(x)≠class(x') add x to Z
Until Z does not change

Fig. 29 Pseudocódigo método de CNN para reducir la base [6]

Se debe considerar el objetivo de esta metodología y es la de encontrar un subset de datos del train set por lo que el set de train al momento de aplicar K-NN (K-Nearest Neighbor) se utilizará como el set de pruebas y el set reducido será el train set, esto se debe a que el set que quiere utilizarse para obtener la información y poder clasificar la data de train set. El principio de esta metodología es tal y como se muestra en la Fig. 29, se crea una matriz Z (Este será el subset que se conseguirá del train set), en distintas fuentes sobre Condensed Nearest Neighbor muestran opiniones contrarias, algunas de estas indican que los datos pueden tomarse de manera aleatoria uno por uno. Después de este paso se realiza K-NN si el valor del train set se clasifica mal a partir de la data en el CNN set (variable Z), esta muestra se pasa CNN, se pasan los que se clasifican mal bajo el principio de que si se clasificó bien es porque el set de pruebas contiene data suficiente para poder clasificar estas muestras y las que se clasificaron mal es porque el set de CNN no contiene la data suficiente para clasificarlos. Este proceso se repite hasta que el subset de CNN deje de cambiar, es decir hasta que este logre clasificar completamente bien todas las muestras en el set de entrenamiento, bajo esta teoría de que si es capaz de clasificar bien todo el set de entrenamiento original pues este subset podrá dar una buena aproximación y como posee menor cantidad de muestras los requerimientos de procesamiento para hacer la clasificación son menor que si se utiliza todo el set de entrenamientos.

C. Análisis:

Lo primero fue extraer la data del data set de mpg proporcionado, a partir de aquí se asignó la columna de mpg en una variable distinta ya que este es el dato que se desea clasificar, esta variable es del tipo continuo por lo que se realizó una agrupación de diez distancias iguales para poder tener diez clases de esta variable después se realizó la permutación de los datos, que consiste en mezclarlos de manera aleatoria para eliminar cualquier tipo de sesgo que pueda existir entre las muestras al igual que dividir los sets de entrenamiento y prueba donde se designó que el set de entrenamiento poseería un 75% de la permutación y el set de pruebas un 25%.

Con esta información se siguió con la metodología de CNN mostrada en la Fig. 29, se creó una matriz vacía Z tanto para las muestras, como para la variable dependiente MPG, se le copio a esta matriz Z el primer dato del set de entrenamiento tanto los siete features como su clase de MPG asociada.

Continuando con el pseudocódigo de la Fig. 29 se creó un ciclo while que dependía de una bandera (anteriormente inicializada como “False”) el primer paso dentro del ciclo while fue realizar K-NN (Para estas pruebas que K-NN siempre debe aplicarse con K = 1 debido a que se desea buscar el valor más cercano al que se esté comparando y tratar de clasificar en base a este, el objetivo es no tener data redundante para poder clasificar las clases), después de aplicar el K-NN en una matriz que compara directamente arrojando “True” si la clasificación fue correcta y “False” si el algoritmo fallo al clasificar esta muestra, iterando cada una de las muestras y evaluando desde el inicio si el algoritmo había clasificado mal otras que ya había clasificado correctamente (Considerando que se incluya data cercana que pueda influir en la decisión para otras clases).

Hasta obtener una tasa de aciertos con respecto al set de entrenamiento de un 100%, en este momento se le da valor "True" al flag si terminar el ciclo while.

La métrica de error a utilizar para medir la exactitud del algoritmo será la tasa de aciertos, esta es la mejor que se puede utilizar para este tipo de aplicaciones como la clasificación debido a que se conoce el valor.

Con todos los features y utilizando K = 11 (Debido que fue la que presentó mejores resultados después de varias iteraciones) la tasa de aciertos obtenida fue de un 43%, lo que se considera una tasa de aciertos baja pero tomando en cuenta la cantidad de clases, un 43% es una tasa de aciertos elevadas ya que la probabilidad de que se seleccione de manera correcta la clase a partir de seleccionarla de manera aleatoria es un 10%, utilizando los subset obtenidos. La Fig. 28 muestra de manera resumida los resultados.

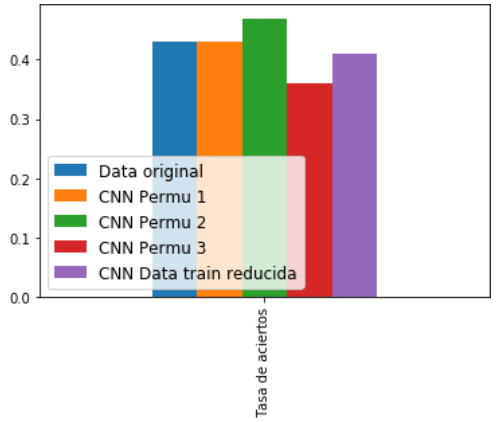

Fig. 30 Gráfica que presenta la tasa de aciertos obtenida

Según lo observado con la segunda permutación de CNN se consiguió la mayor tasa de aciertos que alcanza un 47% lo cual supera el 43% obtenido con todos los features, esto se debe a que CNN busca mantener la información mínima requerida para clasificar cada una de las clases eliminando redundancias en la data e información que este causando sobreajuste en el algoritmo así como eliminar cualquier sesgo en la data, por lo que el modelo se podría ajustar mejor a otros sets que se le coloquen para que los evalúe.

Las cantidades de muestras que conformaban los distintos subsets obtenidos a partir de la metodología de CNN son observados en la Tabla 3:

Tabla III

CANTIDADES DE LOS SUBSETS REDUCIDOS

	Cantidad de datos resultantes CNN
CNN permutación 1	228 muestras
CNN permutación 2	241 muestras
CNN permitación 3	217 muestras
Set de entrenamiento reducido de 75% a 50%	157 muestras

De acuerdo con las tasas de aciertos obtenidas la cantidad de muestras de los subset se observa una proporcionalidad con su tasa de aciertos, donde la permutación dos es la que presentó la mayor tasa de aciertos y mayor cantidad de datos en el subset mediante CNN, pero esto no es una verdadera relación debido a que si se saca una conclusión de esta teoría el set de train completo debería tener la mayor tasa de aciertos. Esta tasa de aciertos del subset de la permutación dos se debe a que CNN logró recopilar la información necesaria y descartar información que solo causaba sobreajuste en el modelo.

Tabla IV

TIEMPOS NECESARIOS PARA REALIZAR LA REDUCCIÓN DE LA BASE

	Tiempo de implementar la metodología CNN
CNN permutación 1	2.5 Segundos
CNN permutación 2	3.1 Segundos
CNN permitación 3	2.2 segundos
Set de entrenamiento reducido de 75% a 50%	1.65 Segundos

En la Tabla 4 se observan los tiempos de clasificación si representan una relación directa con la cantidad de muestras, esto es sencillo de analizar ya que para esas permutaciones que requirieron más muestras también requirieron más iteraciones y esto se ve reflejado de manera directa en el tiempo de ejecución del algoritmo.

Tabla V

TIEMPOS REQUERIDOS PARA CLASIFICAR UTILIZANDO K-NN

	Tiempo de implementar la clasificar
CNN permutación 1	300 ms
CNN permutación 2	375 ms
CNN permitación 3	286 ms
Set de entrenamiento reducido de 75% a 50%	176 ms

En la Tabla 5 se observan los tiempos de clasificar utilizando el algoritmo de K-NN. Para la aplicación que se le da en esta práctica estos tiempos son bastantes bajos, pero pueden llegar a ser significativos en situaciones donde esto deba repetirse 300 veces o 10,000 veces, por lo que esto puede ser un parámetro importante para tomar en cuenta para seleccionar la permutación del CNN porque se puede necesitar llegar a un balance entre la tasa de aciertos, tiempo de clasificación y tiempo de reducción mediante CNN.

Tabla VI

ESPACIO DE MEMORIA REQUERIDO PARA REALIZAR LA REDUCCIÓN DE LA BASE

	Espacio de memoria requerido
CNN permutación 1	16,336 Bytes
CNN permutación 2	17,232 Bytes
CNN permitación 3	14,992 Bytes
Set de entrenamiento reducido de 75% a 50%	10,960 Bytes
Set de entrenamiento completo	22,584 Bytes

Mientras que en la Tabla 6 se presentan los espacios que los subsets ocupan en la memoria son directamente proporcional a la cantidad de muestras que estos tienen, demostrando que para este y otros casos se puede aplicar la metodología de CNN para reducir la base de datos invirtiendo el tiempo que este necesita para reducir la cantidad de datos y ahorrar ese espacio de memoria y ligeramente en otros momentos ahorrar el tiempo de clasificación con el set de pruebas.

Por otro lado, habrán implementaciones en las que se pueden permitir el espacio extra que conlleva usar el dataset completo con todas las muestras y otras en las que se tendrán espacios de memoria reducido y habrá que apoyarse de metodologías como la CNN para poder desarrollar distintos algoritmos en espacios reducidos.

V. SUBSET SELECTION

A. PROBLEMA:

Es el mismo problema que se presenta en la práctica III, Se trabaja con el dataset de MPG el cual contiene ocho features entre ellos continuos, discretos y uno que corresponde al nombre del carro (Este no fue utilizado debido a que es un valor único para cada una de las muestras del set) que contienen la información de la Fig. 27

La práctica requería hacer una separación de la columna mpg en diez clases (Considerando que se debían tener diez rangos igualmente espaciados) esto, para poder realizar en otros pasos la clasificación de estas clases en la que se separó la columna MPG de la data set original a partir de los siete features restantes.

Estos features tienen distintos tipos de variables algunos de estos son discretos y otros continuos.

Para está práctica IV se requiere aplicar subset selection, tanto el algoritmo de forward como el backward al dataset mencionado anteriormente (MPG) para clasificación.

A partir de los subsets obtenidos mediante ambos algoritmos observar el error de realizar la estimación utilizando el algoritmo de K- Nearest Neighbor para poder observar mediante la métrica de error seleccionada que tan eficiente es el implementar estos algoritmos para reducir la cantidad de features de las muestras contra el tiempo y espacio de memoria que conlleva la aplicación de estos algoritmos para reducir el subset.

El objetivo de la práctica es la de encontrar sets con cantidad de features reducidos con respecto al set de datos original.

B. MÉTODO

El algoritmo utilizado para realizar las estimaciones de la clase fue K-NN (K-Nearest Neighbor), el concepto de funcionamiento de este se puede observar en la Fig. 26. Donde se busca encontrar mediante un tipo de medición (Distancia euclideana fue la utilizada para todas las aplicaciones del algoritmo, esto es equivalente a la norma 2 de las matrices) las distancias más cercanas para mediante votación determinar a qué clase pertenece la muestra que se está examinando.

La idea general tras buscar un subset selección es encontrar mediante un algoritmo un set de datos reducido para entrenar los algoritmos de aprendizaje, sea cual sea el método que se utilice el principio de funcionamiento presentado en la Fig. 31 es el siguiente:

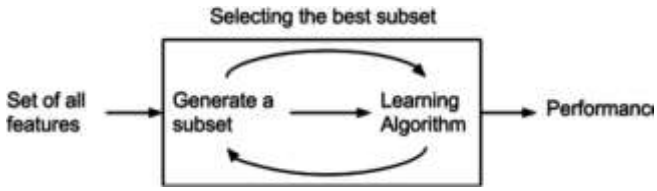

Fig. 31 Principio de funcionamiento de encontrar un subset de datos [4]

Tal como se puede entender a simple inspección de la Fig. 31, encontrar un subset de datos se basa en establecer todos los features, es decir todos los features que tenga el dataset original y a partir de este ir seleccionando el subset que del mejor rendimiento se reduce la cantidad de features o datos a utilizar en el algoritmo de aprendizaje manteniendo un error o tasa de aciertos dentro una tolerancia razonable (Por lo general se toma un 10% del error obtenido con todos los datos o features.).

Se continúa utilizando la misma métrica de error que la implementada para la práctica III utilizando el mismo dataset, esta es la tasa de aciertos donde se compara directamente cuales muestras se clasificaron correctamente y a partir de aquí se hace una sumatoria y se divide entre el total de muestras para encontrar un porcentaje de cuál fue la efectividad de la clasificación.

Por otro lado, los algoritmos implementados para la selección del subset de datos fueron:

- Forward Selection:

Que consiste en probar de manera individual cada uno de los features y seguido de seleccionar el feature resulte con la mayor tasa de aciertos con respecto al set de pruebas, se realizan todas las combinaciones posibles de los features restantes con este feature, es decir que se irá sumando uno por uno los features restantes y realizando todas las combinaciones posibles (sin repetición de estas) para ir determinando el subset con la mejor tasa de aciertos con respecto al set de pruebas. La idea como la seleccionar cualquier subset es la de encontrar una cantidad de features menores a la original que puedan representar dentro de la tolerancia el modelo que se desea implementar [7]. La idea de funcionamiento del forward selection se resume en la Fig. 32:

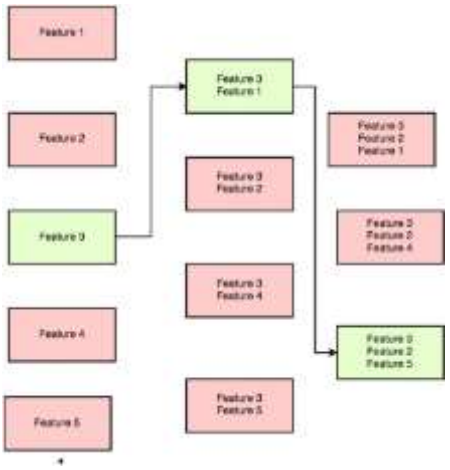

Fig. 32 Principio de funcionamiento algoritmo de forward selection [8]

- Backward Selection:

El backward selection funciona con una lógica similar o forward selection pero una metodología de aplicación distinta a este, este algoritmo implica comenzar a analizar la tasa de aciertos con todos los features que contengan el set de datos original, probar eliminando cada uno de los features y medir como al eliminar cada feature resulta la tasa de aciertos, el feature que al ser eliminado la tasa de aciertos mejore o en otros casos que se mantenga igual será el feature que se eliminara y esto se realizará tantas veces como cantidad de características se tenga en el set de datos o hasta que la tasa de aciertos baje de la tolerancia establecida [8]. Esta idea de backward selection se resume en la Fig. 33

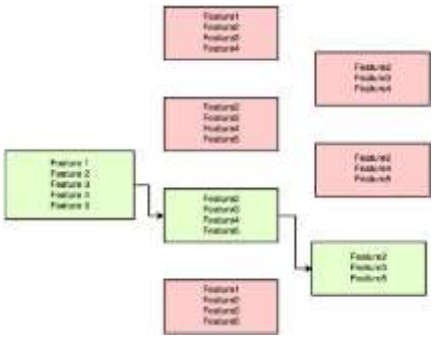

Fig. 33 Principio de funcionamiento algoritmo de backward selection [8]

C. ANÁLISIS:

Después de realizar los mismos pasos que en la práctica III para extraer la data del dataset original al igual que preparar esta para utilizar los algoritmos de subset selection, tanto forward como backward selection explicados en el apartado de método de la esta práctica.

Al iniciar el algoritmo de forward selection el feature con la mayor tasa de aciertos de manera individual resultó ser el feature 3 con una tasa de aciertos 25.74% , este que corresponde al feature de Weight (Peso) tal y como se puede observar en la Fig. 25, después al continuar con las combinaciones el feature combinado con el 3 del cual se obtuvo el mayor resultado fue el 1 que corresponde Displacement (Desplazamiento) con una tasa de aciertos de 50.49%, continuando con las combinaciones a partir de los features 3 y 1, la mayor tasa de aciertos obtenida resultó ser con el feature 5 que corresponde a el modelo del vehículo la tasa de aciertos obtenida con estos features resultó ser del 51.48%, esta fue la mayor tasa de aciertos obtenida con la menor cantidad de features al agregar los demás el algoritmo no era capaz de clasificar correctamente con una tasa de aciertos tan elevada, rondando el 39%.

Al momento de aplicar backward selection tal y como se observa en la Fig. 31, se comenzó a analizar a partir de todos los features que contenía el dataset original, realizando las distintas combinaciones eliminando uno por uno los features para analizar la tasa de aciertos, resultó que la mejor combinación de features fue la misma que utilizando forward selection, tanto los features como la tasa de aciertos obtenida. Estos fueron los features 3, 1 y 5 ya mencionados y una tasa de aciertos de 51.48%. En la siguiente gráfica de la Fig. 34 de histogramas de facilitará la visualización de los resultados.

Se obtuvieron las siguientes tasas de aciertos:

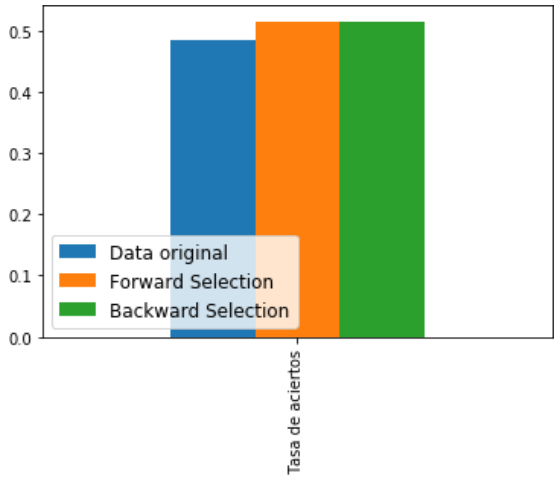

Fig. 34 Comparación tasa de aciertos data original contra fordward y backward selection

Mediante simple inspección se puede observar que la tasa de aciertos de ambos subsets coincide y a la vez esta es superior a la tasa de aciertos que se obtiene con la data original, esto se debe que al igual que el comportamiento observando en la práctica de CNN, con la diferencia de que en esa se eliminaron muestras que representaban data redundante y en esta se eliminaban los features que directamente ocasionaban un peor comportamiento del algoritmo de clasificación, pero estos algoritmos ayudan a reducir el overfitting eliminando estos features que tienen relación inversa a la de los features que si logran clasificar correctamente cada una de las clases.

Tabla VII

CANTIDADES DE LOS FEATURES CON LOS DISTINTOS SUBSETS Y EL SET ORIGINAL

	Cantidad de features
Data original	7
Forward selection	3
Backward selection	3

Tal y como lo plantea el problema y el objetivo de la práctica se obtuvieron dos subsets de menor cantidad de features como se puede apreciar en la Tabla 7 y que a su vez lograron una tasa de aciertos ligeramente mayor al dataset original utilizado.

Tabla VIII

CANTIDADES DE LOS FEATURES CON LOS DISTINTOS SUBSETS Y EL SET ORIGINAL

	Tiempo de implementar el algoritmo
Forward selection	5.918 Segundos
Backward selection	6.118 Segundos

Tabla 1 Tiempo de ejecución de los algoritmos de subset selection

En cuanto al tiempo de ejecución de cada método para encontrar el subset selection, en la Tabla 8 se observa que el forward selection tomo exactamente 200ms menos que implementar el backward selection, siendo un factor por considerar dependiendo de las aplicaciones para cada algoritmo, ya que para esta aplicación los resultados fueron iguales, pero de igual modo la ejecución del backward selection se tomó más tiempo.

Tabla IX

TIEMPO PARA REALIZAR LA ESTIMACIÓN DE CADA SET

	Tiempo para clasificar
Data original	267 ms
Forward selection	215 ms
Backward selection	215 ms

En la Tabla 9 se observan los tiempos invertidos para realizar la clasificación de cada set, los sets obtenidos a partir de forward y backward selection dieron el mismo resultado debido a que su cantidad de muestras y features resultantes es la misma mientras que la data original presenta mayor tiempo y es lógico considerando que esta tiene 4 features más. Para la Tabla 10 se presentan los espacios de memoria.

Tabla X

ESPACIO DE MEMORIA REQUERIDO PARA REALIZAR SUBSET SELECTION

	Espacio de memoria requerido
Data original	22,584 Bytes
Forward selection	11,358 Bytes
Backward selection	14,618 Bytes

En la Tabla 10 se puede apreciar algo parecido a lo que sucedió en la Tabla 9 con los tiempos de ejecución, la idea que da en primer momento es que el backward selection necesitó más pasos y/o variables para implementarse por este motivo este requiere más tiempo al igual que más espacio en la memoria.

VI. FEATURE EXTRACTION: LDA, NEURALNET

PARTEI: USANDO PCA Y LDA

A. Problema

Aplicar las técnicas de feature transformation de PCA (Principal Component Analysis) y LDA (Linear Discriminant Analysis) para el set de datos de MPG clasificación, al igual que aprovechar las propiedades de estas técnicas para reducir la cantidad de feature original.

El objetivo de está práctica es aplicar ambas técnicas de feature transformation para observar tanto las ventajas como algunas desventajas que estas puedan presentar

Con estas se busca observar cómo son capaces de a partir de una transformación lineal conseguir una combinación de features con información equivalente a la data original, de esta misma forma aprovechando las propiedades de la algebra lineal se podría reconstruir la matriz original a partir de esa data reducida, en esta práctica se observará que tal fue el resultado de esto, por otro utilizar los sets reducidos que se obtengan para estimar las distintas clases de MPG y observar el desempeño de estos sets.

Es importante destacar que para la implementación de estas técnicas se solicita hacerlo sobre el set de MPG clasificación por lo que nuevamente se deberá separar la variable objetivo MPG en diez distintas clases recordando la importancia de comparar las implementaciones de PCA y LDA.

B. Método

Después de realizar los mismos pasos que los ejercicios en el inicio de la práctica III y IV para adaptar la data, extraerla, al igual que mantener la misma permutación de estas prácticas para que los resultados obtenidos puedan ser comparables entre una práctica y la otra.

Se aplicaron PCA y LDA y sus propiedades para transformar y reducir los features de la data original, pero ¿En qué consisten estas técnicas de PCA y LDA?

- PCA:

Considerado un aprendizaje no supervisado debido a que este ignora o toma en cuenta las distintas clases que puede tener el set sobre el que se implementa y su objetivo es directamente encontrar los componentes principales o también llamados direcciones que son capaces de maximizar la varianza entre todo el set de datos, haciendo esto se facilita el comprender a que clase pertenece cierto rango de valores de los distintos features [9].

Esta sección de maximizar la varianza entre los datos suele ser un punto que no se logra visualizar muy bien, por lo que en la Fig. 35 se aprecia mejor:

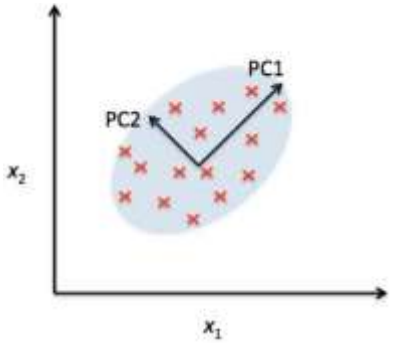

Fig. 35 Representación de maximizar la varianza de los datos para dos features [9]

Tal y como se muestra en la Fig. 35 lo que se busca es dispersar los datos hasta la varianza máxima que exista entre estos, tal y como se mencionó con el objetivo.

- LDA:

A diferencia de PCA, la técnica de LDA busca un subespacio de features que logre maximizar la separación de todas las clases [9]. Esta separación de las clases se puede observar en la Fig. 36 como:

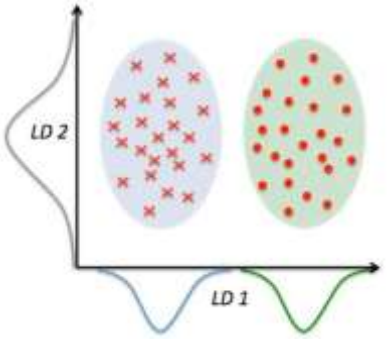

Fig. 36 Representación de los subespacios encontrados para lograr maximizar la separación de features [9].

La Fig. 36 ayuda bastante a poder observar la diferencia básica entre PCA y LDA, donde PCA solo se enfoca en dispersar los datos en base a la máxima varianza de estos, LDA busca subespacios que pueda maximizar la dispersión de las distintas clases. Cabe indicar que LD 2 en la Fig. 34 sería un mal feature a utilizar para tratar de clasificar estas dos clases.

Aunque de manera intuitiva se puede indicar que LDA es superior a PCA para una tarea de clasificación de varias clases donde se conocen las etiquetas de clase, este no siempre es el caso.

Hay ciertos casos y aplicaciones como, por ejemplo: Las comparaciones entre las precisiones de clasificación para para el reconocimiento de imágenes después de usar PCA o LDA muestran que PCA tiende a superar a LDA si el número de muestras por clase es relativamente pequeño, aunque en la práctica no extraño encontrar ambas técnicas implementadas juntas, PCA para la reducción de dimensionalidad seguido de un LDA para por lo general una mejor transformación de los features [9].

Los pasos para implementar el algoritmo de PCA comienzan indicando la estandarización de los datos, después de distintas pruebas para la permutación utilizada el dataset de MPG no se ve afectado en gran medida si se estandarizan o no los datos (alrededor de un 5% de la variación de los resultados) por lo que no se estandarizaron los datos para agilizar los fines prácticos y más importantes de la implementación de esta técnica. Pero de manera general estandarizar los datos con media = 0 y varianza =1, para que los datos queden en un rango de 0 – 1 para impedir que el algoritmo de PCA se vea afectado por las escalas de los distintos features, los otros pasos son, obtenido de Raschka (2014):

- Calcular la matriz de covarianza de dimensiones.
- Obtener los vectores propios y los valores propios de la matriz de covarianza
- Ordenar los valores propios obtenidos en orden descendente y elegir los k vectores propios más grandes (k es el número de features que tendrá el nuevo subset).
- Construir la matriz de proyección W a partir de los k vectores propios seleccionados y transformar el conjunto de datos original a través de W para tener el nuevo subset reducido y con los features transformados.

Los pasos para implementar el algoritmo de LDA presentan cierta similitud, se deben estandarizar los datos de igual manera que si de PCA se tratará, pero esto ya se explicó en el párrafo anterior. Los pasos para aplicar LDA obtenidos de Raschka (2014) son:

- Determinar los d-dimensión vectores medios:

Se debe determinar el vector media correspondiente a cada una de las clases y con tantas columnas como features iniciales tenga el set de datos.

- Calcular la matriz de dispersión de las clases:

Se calcula siguiente la ecuación de la Fig. 37:

$$S_W = \sum_{i=1}^{c} S_i$$

Fig. 37 Formula para determinar la sumatoria de la dispersión de las clases [10]

En la Fig. 35 se presenta el término a S_i la manera de calcular este se observa en la Fig. 38:

$$S_i = \sum_{\boldsymbol{x} \in D_i}^{n} (\boldsymbol{x} - \boldsymbol{m}_i)\,(\boldsymbol{x} - \boldsymbol{m}_i)^T$$

Fig. 38 Formula para determinar la dispersión de las distintas clases [10].

Donde: m_i son cada uno de los vectores media y x les la matriz de datos originales. Esto es equivalente a multiplicar la matriz de covarianza por la cantidad de datos de la clase menos 1, esta fórmula se puede observar en la Fig. 39:

$$S_W = \sum_{i=1}^{c} (N_i - 1)\Sigma_i$$

Fig. 39 Formula equivalente utilizando la covarianza y cantidad de datos por clase [9].

- Calcular la matriz de dispersión entre las clases llamada S_B tal como se observa en la Fig. 40:

$$S_B = \sum_{i-1}^{c} N_i (\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^T$$

Fig. 40 Formula para determinar la dispersión entre las clases [9].

De donde en la Fig. 40 se presenta a: N_i es la cantidad de muestras, m_i los vectores media de cada clase y m es la media de todos los datos del dataset.

- Se obtienen los discriminantes lineales resolviendo la ecuación presentada en la Fig. 41:

$$S_W^{-1} S_B$$

Fig. 41 Formula para obtener los valores y vectores propios de los distintos features [9].

Después de obtener los valores y vectores propios resultante de computar la ecuación presentada en la Fig. 41, como paso extra se realizó una proporción de los valores dividiendo cada valor propio sobre la sumatoria de esto para poder encontrar la proporción en que estos afectan el modelo a partir de aquí ya se pueden descartar los features con valores de proporción bastante bajos indicando que no afectan el modelo estimador con gran mejoría.

Tanto para PCA como para LDA se multiplicará la matriz W con los vectores propios de los features considerados los más relevantes con la data original, esta multiplicación en ambos métodos resultará con una matriz z de features reducidos y la misma cantidad de muestras del set de datos introducido.

Tal y como se mencionó en puntos anteriores tanto PCA como LDA hacen una transformación lineal de los features originales, aún después de la reducción de los features que se consideren menos importantes se puede hacer una reconstrucción de la data original a partir de esta data transformada, para esto se multiplica la matriz W que tendrá los vectores propios de los features considerados más relevantes para el algoritmo con la matriz z que contiene los features reducidos. Esto dará como resultado una matriz con la cantidad de features originales con cierta semejanza a la original, pero en esta transformación lineal se pierde bastante información y se verá reflejado en que algunos features ni siquiera quedarán dentro del rango de escala original.

C. Análisis:

El algoritmo utilizado para clasificar en ambos puntos fue K-NN con K =11, para resumir los resultados obtenidos tanto con el set original de datos, como para los sets reducidos con PCA y LDA está la siguiente gráfica de histograma que se presenta en la Fig. 42:

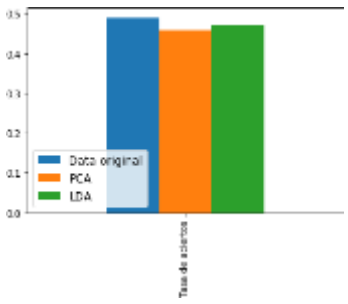

Fig. 42 Tasa de aciertos, data original y los subsets PCA y LDA

En ambas aplicaciones observadas en la Fig. 42, tanto en PCA como en LDA los subsets solo estaban compuestos por tres features, no se le puede indicar un nombre a estos debido a que esta es la mayor desventaja de aplicar alguna de estas técnicas, la pérdida de información sobre los features del dataset, donde ya no se le puede indicar un nombre especifico y no se puede tener algún tipo de medición de lo que estos features reflejan en la realidad, más que una combinación lineal de los demás features.

Las tasas de aciertos presentadas en la Fig. 20, sus valores obtenidas fueron, 48.97% para el set de datos original, 45.91% para el set de datos obtenido a partir de PCA con los tres features seleccionados se obtuvo 45.91% y para el set de datos obtenido con la técnica de LDA la tasa de aciertos obtenida fue de 46.94%.

A simple inspección en la Fig. 42se observa que la mejor tasa de aciertos se obtuvo con el dataset original, a pesar de reducir los features ¿Por qué la tasa de aciertos no mejoró como en otras aplicaciones? Esto se debe a que estas transformaciones lineales tal y como se ha mencionado crean nuevos features con combinación de todos los originales dentro de estos, es decir que si el dataset original contenía algún feature que podría está causando sobre ajuste en el modelo clasificador o simplemente presenta una relación inversamente proporcional con otro feature que si es capaz de clasificar de manera correcta los datos, este/estos siguen presentes en alguna proporción dentro de cada feature creado por las técnicas de PCA y LDA.

Se realizó la reconstrucción de los datos originales a partir de las transformaciones y reducciones obtenidas con las técnicas de PCA presentada en la Fig. 43 y LDA presentado en la Fig. 44. Esto fue como fines únicamente demostrativos debido a que en LDA se pierde más información y por lo general la matriz resultante de la transformación no es invertible y hay que recurrir a otros métodos como pseudo inversa de la matriz que no se tocarán para este apartado.

La reconstrucción a partir de PCA arrojó valores que se pueden visualizar en la Fig. 43.

```
([[ 4.89948846e+00,  3.06168274e+02,  1.25240657e+02, ...,
   -2.77801705e+00, -3.17091915e+00, -1.40904135e+00],
  [ 5.48024085e+00,  3.49120134e+02,  1.59964745e+02, ...,
   -5.54600540e+00, -4.95036851e+00, -1.36172229e+00],
  [ 4.99429838e+00,  3.17146607e+02,  1.45098722e+02, ...,
   -4.76966440e+00, -4.35052226e+00, -1.25479758e+00],
  ...,
  [ 2.56668800e+00,  1.50035813e+02,  8.46095822e+01, ...,
    4.77849775e-01, -6.09917723e-01, -6.70623541e-01],
  [ 2.38267593e+00,  1.39055423e+02,  8.07479224e+01, ...,
    4.18615897e-01, -5.84100204e-01, -6.06683416e-01],
  [ 2.07162610e+00,  1.19036736e+02,  7.35311039e+01, ...,
    7.15847575e-01, -3.16264023e-01, -5.16105867e-01]])
```

Fig. 43 Reconstrucción de la data original a partir del subset de PCA

La reconstrucción a partir de LDA arrojó valores que se pueden visualizar en la Fig. 44.

```
[[  4.68915018 361.09332743  85.65196783 ...  -1.38913462  70.01297797
   -1.33466981]
 [  5.67266465 401.74133851 115.96263363 ...  -3.87009579  65.25975483
    9.23188055]
 [  5.11959832 369.18950224 102.32716109 ...  -3.16870961  65.95658445
    7.95692942]
 ...
 [  2.30361202 256.11482868  14.37433895 ...   3.91385025  91.99974809
  -19.04069929]
 [  2.14367071 241.20466782  12.36417789 ...   3.72262304  90.85120808
  -16.74952426]
 [  1.82828078 221.45281117   5.00732479 ...   4.03044689  90.8766879
  -16.94152332]]
```

Fig. 44 Reconstrucción de la data original a partir del subset de LDA

Ambas reconstrucciones observadas en la Fig. 43 y Fig. 44 correspondientes a PCA y LDA dieron resultados simulares para la reconstrucción, para algunos de los features se perdió bastante información, pero por una simple inspección se puede notar que en el caso de LDA se perdió más información al intentar reconstruir.

PARTE II: USANDO NEURAL NETS

A. Método

Después de aplicar los métodos de PCA y LDA tanto para reconstruir la matriz original y ver que tanta información se perdió en este proceso, como para a partir de los features reducidos realizar la clasificación y observar el resultado, utilizaron redes neuronales para hacer la comparación de la reconstrucción realizada utilizando PCA y comparar la estimación realizada a partir de la matriz reducida de LDA, la práctica también requiere que la red neuronal cuente con tres neuronas en su capa oculta.

El objetivo de esta práctica es comparar la reconstrucción de los features originales a partir de PCA y una red neuronal, así como comparar la clasificación en las distintas clases entre LDA, apoyándose de K-NN y una red neuronal.

Para construir las redes neuronales requeridas se utilizó la aplicación de MATLAB “nntool” [10] que de una manera bastante dinámica permite configurar redes neuronales y la data para esta facilitando de igual forma el importar, usar y exportar la información. Manteniendo la permutación utilizada en las otras aplicaciones para poder comparar los resultados.

B. Análisis:

Para el primer caso que consiste en obtener los features originales a partir de usar una red neuronal que los reduzca a 3 y luego intente transformarlos en la cantidad original (7).

Para esto se configuro tanto la entrada como el objetivo en la interfaz de nntool con las 392 muestras y sus correspondientes 7 features, al igual que se configuró que en la etapa media (capa oculta) tuviera una cantidad de únicamente 3 neuronas para que la red hiciera los cálculos necesarios para transformar esos 7 features originales en solo 3 pero como se le indicó para este caso el objetivo de obtener la matriz original la capa de salida tiene 7 neuronas encargadas de presentar el resultado obtenido mediante este proceso, se puede observar en la Fig. 45.

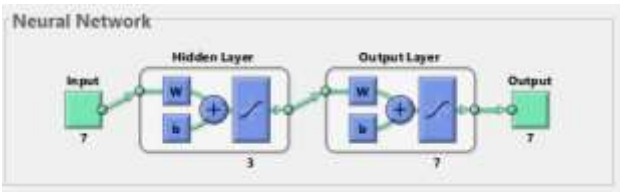

Fig. 45 Red neuronal utilizando nntool

A partir de simple inspección y como se especifica en la Fig. 45, la capa de entrada debe contar con un tamaño de 7 debido a los 7 features utilizadas, la capa oculta o como se muestra en la Fig. 43 Hidden Layer cuenta con 3 neuronas como se había especificado anteriormente, la capa de salida cuenta de igual manera con 7 debido a que se está intentando recuperar los datos original por lo que debe dar un resultado de 7 columnas que serían una transformación aproximada de los features originales.

De igual forma la herramienta de nntool determina la gráfica del error mínimo cuadrado (Que no es más que el promedio elevado al cuadrado de la resta entre el resultado original y el estimado, dando una idea de que tan cercanas o alejadas están las estimaciones) e indica cual fue el epoch de menor error (el término epoch indica un ciclo completo en el que se entrena la red, es decir realizar el fordward y backward a lo largo de todas las muestras de una red equivale a un epoch). Esto se resultado se pueden observar en la Fig. 46.

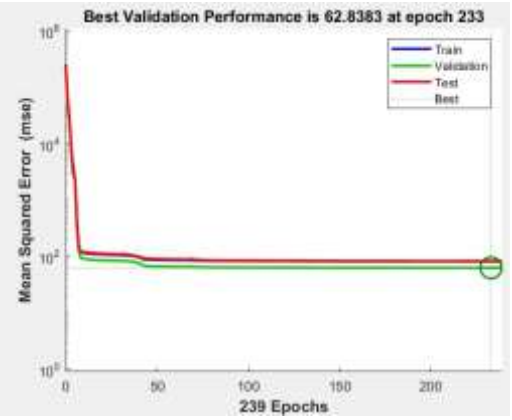

Fig. 46 Error mínimo cuadrado de la aproximación de la red utilizando nntool

En la Fig. 46 se tiene una idea de que tanta información se perdió al intentar reconstruir la matriz original utilizando redes neuronales en el apartado de comparación se observarán las diferencias de la reconstrucción utilizando PCA, al igual de observar cual fue el mejor Epochs.

Por otro lado, la otra aplicación que se le dio a la herramienta de MATLAB nntool fue la de crear una red neuronal que clasifique las distintas etiquetas a partir de la data original, para estos se le asignó como entrada lo mismo que para la red anterior presentada en la Fig. 45, los 7 features

originales con las 392 muestras, una capa oculta de 3 neuronas para simular el procedimiento lo más semejante posible al realizado con LDA y que se comparará posteriormente y una capa final de 1 neurona que será la encarga de procesar la información calculado en la capa oculta para obtener la información de la estimación final, la red quedaría como la Fig. 47.

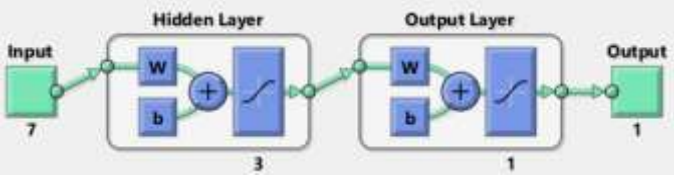

Fig. 47 Red neuronal para realizar la estimación de las etiquetas

Después de crear y entrenar la red presentada en la Fig. 47 se observó el error cuadrado promedio tal y como en la Fig. 46 para determinar qué tan cercano o alejado de las etiquetas reales están las estimaciones calculadas, esto se presenta en la Fig. 48.

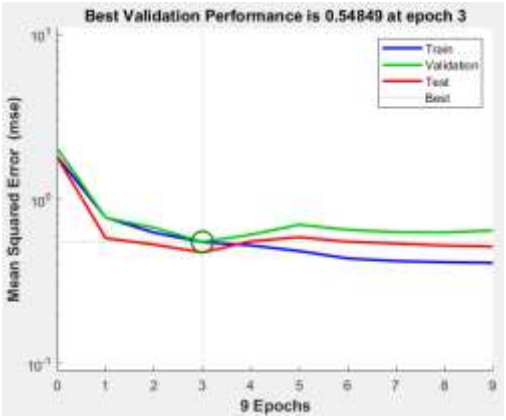

Fig. 48 Error mínimo cuadrado de las estimaciones de las etiquetas

En la Fig. 48 se puede observar el error mínimo cuadrado tanto para el train como para el test al intentar realizar la estimación de las clases, en un círculo indica cual fue el mejor epoch y cuál fue el valor, para este caso dio un error mínimo cuadrado de 0.54849 aproximadamente, esto es un error bastante bajo ya en la tasa de aciertos se sabrá porque, sin embargo, esta métrica de error mide la distancia entre el valor real y el estimado porque al aún existir una diferencia al utilizar otra métrica de error como la tasa de aciertos los resultados serán distintos.

La tasa de aciertos obtenida a partir de la red neuronal mostrada en la Fig. 47 fue de 57.3980%. Este es un resultado bastante bueno si se compara a simple vista con el presentado en la Fig. 42.

C. Comparación entre los métodos

La primera comparación que realizar es la de la reconstrucción de los features originales, donde para esta operación se compararán la reconstrucción utilizando el método de PCA en la Fig. 43 y la reconstrucción a partir de la red neuronal en la fig. 49.

1	2	3	4	5	6	7
7.4552	315.6184	136.2646	3.4969e+03	13.3202	74.3635	1.5442
7.7862	365.5513	155.6754	3.6903e+03	12.2200	73.5334	1.5094
7.6470	335.2907	142.7386	3.4360e+03	12.6908	73.9401	1.5262
7.5423	322.1840	138.1192	3.4296e+03	13.0329	74.1835	1.5365
7.4658	314.5631	135.6065	3.4420e+03	13.2577	74.3371	1.5429
7.9359	413.6415	185.6612	4.3479e+03	11.3134	72.6446	1.4721
7.9673	425.6317	193.1775	4.3564e+03	10.7254	72.2275	1.4497
7.9560	420.3732	189.2329	4.3136e+03	10.9507	72.4052	1.4591
7.9675	426.4749	194.5885	4.4261e+03	10.7578	72.2211	1.4501
7.9060	396.4708	170.9557	3.8490e+03	11.4208	72.9091	1.4812
7.9076	390.9001	166.5315	3.5755e+03	11.2738	72.8717	1.4783
7.7382	354.7868	150.7471	3.6027e+03	12.4132	73.6981	1.5162
7.9038	394.0926	169.2838	3.7707e+03	11.4041	72.9177	1.4813
7.9920	428.9978	191.4345	3.0942e+03	9.4212	71.4256	1.3950
4.2502	115.5152	77.6406	2.3691e+03	16.9212	76.9239	1.6537

Fig. 49 Muestra de la reconstrucción de la data utilizando una red neuronal

Haciendo un análisis a partir de la Fig. 43 y la Fig. 49, la reconstrucción de la red neuronal para la data original es mucho más acertada a esta, los datos quedaron dentro de su rango original y con valores similares mientras que para el caso observado en la Fig. 43 la reconstrucción presenta datos que están fuera del rango de valores de la data original presentando que con este método se perdió aún más información, esto se puede deber a que la red se entrena varias veces al igual que evalúa su error en distintas operaciones hasta seleccionar el entrenamiento de la red que resulte con la mejor efectividad, con este análisis se comprende que el error mínimo cuadrado obtenido anteriormente a partir de la reconstrucción de la red neuronal mostrado en la Fig. 46 si era un buen resultado.

Observando los resultados para la clasificación utilizando la reducción y transformación mediante LDA contra la clasificación realizada utilizando la red neuronal, directamente la métrica de error fue la tasa de aciertos, LDA obtuvo una tasa de aciertos de 46.94%, mientras que utilizando la red neuronal la tasa de aciertos obtenida fue de 57.3980%, a simple vista se puede observar que la transformación y estimación obtenida a partir de la red neural presenta mejores resultados que los métodos de feature transformation de PCA y LDA por lo que comparando los 4 métodos se puede notar que para ambas aplicaciones los algoritmos de redes neuronales presentan mejores resultados, cabe destacar que tanto para la reducción como la clasificación la capa oculta solo poseía 3 neuronas para hacer un proceso semejante al seguido con PCA y LDA en los que se redujo la cantidad de features transformados para realizar la estimación a 3 y a partir de estos se realizó la reconstrucción de la data y la estimación de las clases.

VII. RED NEURONAL

A. *Método*

Se aplicó el algoritmo de backprogation sin el uso de librerías para estimar la regresión del mismo data set de las prácticas anteriores, el estudio de los distintos vehículos donde se especifican 7 features y tiene una cantidad de muestras de 392 sin ningún tipo de valor NaN, esto significa not a number, en español, no es un número (Es decir, 392 muestras de las 398 que contiene el dataset tienen toda la información de los features).

El objetivo de esta práctica es el de implementar, observar y analizar el algoritmo de backpropagation.

El algoritmo ya mencionado consiste en hacer que una red neuronal aprenda a partir de observar sus resultados, los cálculos y los valores iniciales, una idea muy simplificada de como esto funciona es que existe una primera capa donde se introducen los 7 features, por lo que habrán 7 neuronas, una capa oculta que para fines de esta aplicación se decidió utilizar de 3 neuronas, la comunicación entre la capa de entradas y la capa oculta es mediante los pesos (la analogía más certera sería la línea que une cada una de las neuronas se pueden visualizar como pesos) cabe destacar que mediante este método es que algunos tipos de redes neuronales aprenden de manera supervisada ya que al observar sus resultados y en base a este ajustar los pesos de la misma se considera que la red esta literalmente aprendiendo de sus errores El pseudocódigo correspondiente a este algoritmo fue obtenido de Alpaydin (2014), se presenta en la Fig. 50.

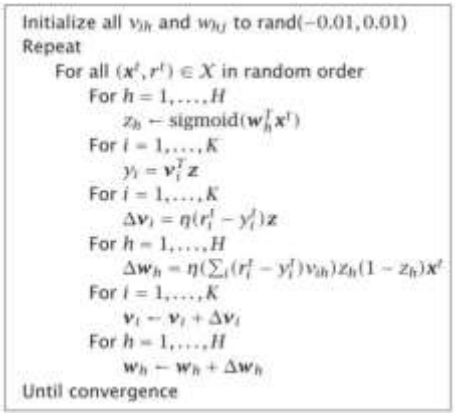

Fig. 50 Pseudocódigo algoritmo de backpropagation [6]

A partir de observar el seudocódigo de la Fig. 50 se requirió comprender como funciona la función de activación sigmoid que será la utilizada para este caso, pero para poder comprender porque se utiliza, se debe entender para que se utilizan estas funciones de activación, estas son utilizadas para definir una salida dentro de unos parámetros dada las entradas que se le introduzca, por ejemplo un circuito eléctrico sufre de activación mediante una función step, cuando recibe corriente eléctrica los circuitos reciben un impulso relativamente instantáneo.

La función de activación utilizada fue la sigmoid, su principió de funcionamiento es la de acotar los valores de entrada entre un rango de 0 y 1, la función sigmoid se presenta en la Fig. que fue la misma fórmula aplicada en el código, de igual manera en el código se requirió aplicar la derivación de esta función.

$$f(x) = \frac{1}{1 - e^{-x}}$$

Fig. 51 Función sigmoid [11]

Utilizando la función presentada en la Fig. 51, se hacen los primeros cálculos pertinentes de la red neuronal de un proceso llamado feedforward, este consiste en el proceso por el cual la red neuronal es capaz de realizar la estimaciones, se multiplica la matriz original o la que se utilizará para entrenar el algoritmo por la primer matriz de pesos que indica la importancia de alguna manera de cada una de las neuronas y su combinación para la capa oculta, después se multiplica por la función sigmoid este resultado para llevar esas entrada a un rango de valores entre 0 y 1, de igual manera se multiplican los pesos de la capa oculta hacia la capa de salida y el resultado se multiplica por la función sigmoid, y este resultado es el que se considera la salida de la red neuronal, para este caso la estimación de la regresiones para la variable dependiente mpg.

Para realizar el backpropagation tal y como se presenta en la Fig. 50. Lo primero es restar los valores de entrenamiento de mpg y la estimación realizada a partir del paso anterior, el resultado de esta resta se multiplica por la derivación de la función sigmoid de la Fig. 51 el resultado de esta multiplicación refleja el gradiente para los pesos, es decir una diferencia que indica cuales pesos deben ajustarse, estos se hace tanto para los pesos que van de la capa oculta a la capa de salida como los pesos que van desde la capa de entrada a la capa oculta, después de estos cálculos de los gradientes se ajustan los pesos en base a esto para eso se suma cada matriz de pesos con el nuevo valor que se determinará, este se determina para la primera matriz de pesos que es la que va desde la capa de entrada como la entrada de la data de entrenamiento transpuesta multiplicada por el gradiente calculado para esta al igual que para la segunda matriz de pesos se multiplica el cálculo realizado para la estimación en la capa oculta transpuesto multiplicado por su respectivo gradiente, con estas operaciones ya se explica el fundamento de cálculo y funcionamiento del algoritmo implementando.

La manera en que este algoritmo aprende es mediante el método de gradiente descendiente que es como se ajustan los pesos tantos de la capa de entrada a la capa oculta como de la capa oculta a la salida, este gradiente, este es calculado para determinar el mayor decremento del error y de esta manera minimizarlos.

B. *Análisis:*

Los resultados obtenidos del algoritmo implementado tal y como se explicó en el apartado anterior fue que a medida que más se entrenaba el algoritmo el error iba disminuyendo hasta un punto en el que se creaba una asíntota en el eje x y el error tendría a este valor, la métrica de error utilizada fue el error

mínimo cuadrado, ya que la implementación del algoritmo fue para estimar los valores de la variable dependiente mpg mediante regresión ya esta cuenta con valores continuos, la curva de aprendizaje del algoritmo que muestra los errores obtenidos se muestra en la Fig. 52.

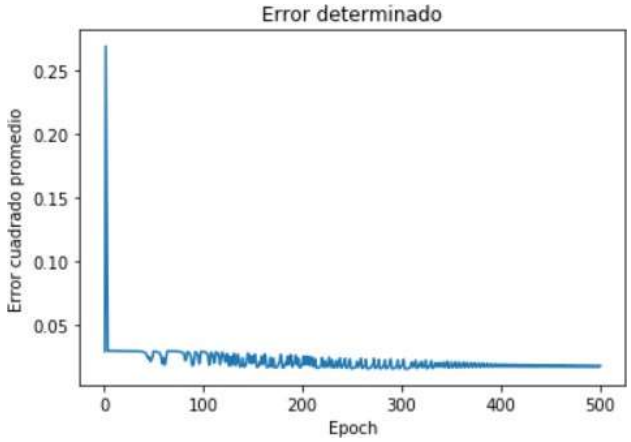

Fig. 52 Gráfica del error cuadrado promedio de la red neuronal a medida que se entrenaba

A medida que más se iba entrenando la red neuronal la tendencia del error cuadrado promedio iba disminuyendo, la tendencia que este presenta es la ir acercándose a 5, a pesar de que en la gráfica se vea un 0.05 esto es un tema de las escalas de la gráfica pero sigue siendo proporcional, en base a la gráfica de esta Fig. 52 se comprende que el backpropagtion fue una muy buen algoritmo para esta aplicación de regresión dio resultados bastantes bajos y el entrenamiento de la red se puede observar en el descenso de la gráfica presentada en la Fig. 52.

VIII. INVESTIGACIONES

A. *Deep Learning*

Deep Learning o en español conocido como aprendizaje profundo, la siguiente definición fue obtenida de Rouse (2017). *"Mientras que los algoritmos tradicionales de aprendizaje automático son lineales, los algoritmos de aprendizaje profundo se apilan en una jerarquía de creciente complejidad y abstracción"* [10]. En otras palabras, mediante esta metodología de las redes neuronales se plantea como meta que las computadoras aprendan mediante distintas pruebas con la información de entrada que al combinarse resulten la salida especificada, a partir de aquí las computadoras pueden aprender, a partir de aquí después de determinar la estructura de la de la red neuronal que funciona para cumplir con la función que se le especifique cuando se vaya a hacer alguna actividad con esta solo utilizará esa estructura, este es un razonamiento simplificado de las redes neuronales del cerebro humano, actividades recurrentes como hablar o caminar todo el tiempo son ejecutadas a partir de una estructura que se creó con aprendimos a realizar estas actividades para cuando necesitemos repetirla solo deba ejecutarse esta estructura y no tener que aprender de nuevo, como se mencionó anteriormente el aprendizaje profundo es una idea simplificada de cómo funciona una red neuronal del cerebro, pero, qué diferencia tiene esto contra la idea de machine learning que ya conocemos, en la Fig. 53 se muestra de manera muy pero muy reducida una comparación entre ambos métodos.

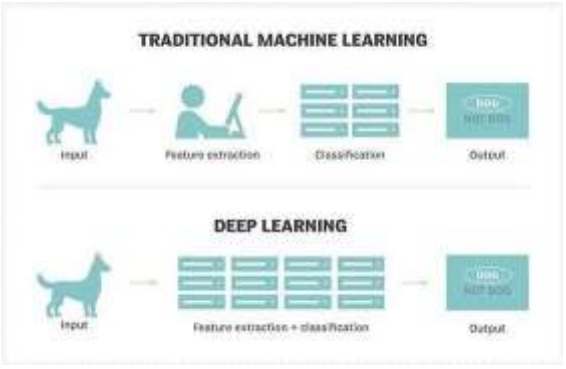

Fig. 53 Comparación de machine learning tradicional contra deep learning [12]

A simple inspección se puede comprender de la Fig. 53 que se está haciendo una comparación de clasificar si es o no un perro también que el método tradicional de machine learning conlleva la extracción manual de las características de los datos para este ejemplo de las imágenes y después se debe implementar un algoritmo para que en base a las características extraídas realice la clasificación, mientras que una red neuronal programada para esta aplicación es capaz de determinar cuáles son las características de las imágenes introducidas así como clasificarlas.

En la Fig. 54. Se muestra una simplificación de cómo se suelen visualizar las redes neuronales para poder describir sus tres capas generales.

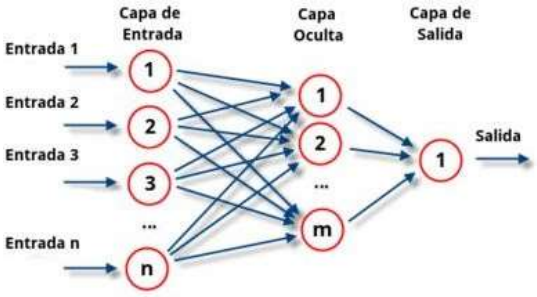

Fig. 54 Capas principales de una red neuronal [11]

En la Fig. 54 se pueden observar las principales capas de una red neuronal, donde la capa de entrada son neuronas dedicadas a asimilar los datos de entrada así sean imágenes, directamente tabla de datos o cualquier otro formato para representar la información. La capa oculta, es la que esta encarga de realizar el procesamiento de la información y hacer los cálculos intermedios, mientras más neuronas se utilicen más complejos serán los cálculos porque se probaran más posibles

escenarios y la capa de salida, consiste en la toma de decisión o tomar alguna decisión en base a los cálculos realizados.

Un ejemplo un poco más enfocado a la realidad utilizando la metodología de deep learning es el que se observa en la Fig. 55, obtenido de Cepelewicz (2019):

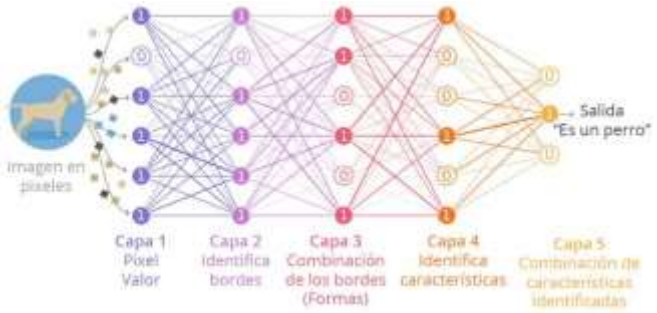

Fig. 55 Identificación de función de cada tapa para esta aplicación [12].

De manera resumida lo que se observa en la Fig. 55 es una estructura más compleja de lo observado en la Fig. 54, debido a que mantiene las mismas tres capas pero presenta más capas ocultas, explicando lo que sucede en la Fig. 55, existe una primera capa que es la que recibe e identifica los datos para este ejemplo son pixeles, una segunda capa de neuronas donde comienza la capa oculta, a partir de esta empieza la extracción de las características de las imágenes para este ejemplo la detección de los bordes, se pasa a una tercera capa que consiste en la combinación de los bordes para identificar formas dentro de la imagen, a partir de esos cálculos se identifican característica más profundas de cada una de las imágenes que el algoritmo pueda reconocer y en la última capa toma una decisión a partir de referencias que se le introducen antes de comenzar todas las etapas.

Entre las aplicaciones del deep learning muchas conocidos y que solemos convivir con ellas día a día son:

- **Traductor de Google y asistentes virtuales:**

Este es un ejemplo que todos hemos experimentado, ya los asistentes virtuales utilizan algoritmos de aprendizaje profundo para hacer reconocimiento de voz al igual que el traductor de Google que es capaz de reconocer la voz y el idioma.

- **Diagnósticos médicos:**

Mediante una base de datos y algoritmos de deep learning se han logrado aplicaciones que mediante imágenes de tomográficas y radiográficas se pueden identificar masas anómalas que se encuentren en el cuerpo, así como en base a los síntomas de un paciente puede hacer una predicción de la que este posee.

- **Conducción autómata:**

Con los algoritmos de deep learning y ayuda de otros tipos de algoritmos se logra que mediante un conjunto de sensores y cámaras un vehículo sea capaz de desplazarse de manera autómata detectando, bordes, vehículos, semáforos al igual que personas y cualquier otro objeto.

B. Reinforcement Learning

El Reinorcement Learning o también conocido en español como aprendizaje por refuerzo, la siguiente definición fue obtenida de Silva (2019) *"Es un área de la inteligencia artificial que está centrada en descubrir que acciones se debe tomar para maximizar la señal de recompensa, en otras palabras, se centra en como mapear situaciones a acciones que se centren en encontrar dicha recompensa. Al agente no se le dice que acciones tomar, sino al contrario él debe experimentar para encontrar que acciones lo llevan a una mayor recompensa, los casos más desafiantes son los que no llevan a una recompensa inmediata si no en las siguientes situaciones"* [13]. Es decir, es un área de centrada en descubrir cómo resolver los problemas que se les plantea observando como varían los aciertos o cualquier métrica de error utilizada para conocer qué tipo operaciones realizar, tal como indica la definición las implementaciones más desafiantes son en las que después de la iteración no se recibe inmediatamente una medida de error por lo que el programa realiza varias iteraciones y después descubre que realizo mal.

Este cuenta con tres características que lo diferencian en gran manera los otros métodos, es considerado un sistema de aprendizaje de lazo cerrado debido a que se sigue un ciclo donde el algoritmo observa el estado de los resultados, realizar los cálculos correspondientes y determina el error que para este método se suele llamar recompensa debido a que tiene una lógica un poco distinta ya que de esta manera reconoce que operaciones le han funcionado, a este tipo de algoritmo no se le indica que salidas o resultados son los correctos, sino que debe reconocerlo mediante la recompensa utilizando el método de prueba y error [13].

Este tipo de algoritmos son bastantes buenos para tareas que se consideren continuas, que evolucionen constantemente o tareas por segmentos donde los estados iniciales y finales están en constante cambio, convirtiéndolo en un buen candidato para la toma decisiones en tiempo real. Algunas aplicaciones son:

- Economía:

Para realizar predicciones en el mercado bursátil se suelen aplicar algoritmos de reinforcement learning capaces de adaptarse a esos cambios continuos.

- Videojuegos:

Debido a esta capacidad de cambiar los cálculos, estos algoritmos son bastante utilizados para la inteligencia de los videojuegos, donde los enemigos y el entorno se deben ir acomodando a las acciones que toma el usuario.

- Navegación con robots:

Es una de las grandes aplicaciones, implementado en el tipo de robots que simulan la caminata, saltos e incluso en cuadrúpedos debidos a que estos deben tomar decisiones rápidas según lo que van examinando del ambiente, estas decisiones que toman son reajustes de dirección y velocidad por lo que son considerados vitales.

IX. Conclusiones

En todo este trabajo se implementaron distintos métodos algunos para las mismas aplicaciones y otros no, los primeros algoritmos consistieron en clasificar dos tipos de habichuelas, más específico clasificar entre las habichuelas Jacomelo y las pinta.

Para el primer algoritmo se implementó Bayes utilizando una sola características de las imágenes de las habichuelas, esta característica seleccionada fue la varianza de los pixeles en la escala de grises de las imágenes, al obtener esta característica para cada una de las habichuelas se utilizó la fórmula de la Fig. 5 para determinar las densidades gaussianas debido a que los valores de la característica obtenida son del tipo continuos, previó se observó en los histogramas si esta característica verdaderamente presentaba un comportamiento gaussiano en su distribución al comprobar que si o al menos que fuera muy semejante se aplicó la formula mencionada anteriormente, no se tomaron en cuenta las probabilidades a prior debido a que se indica que estas deben ser iguales por lo que sería un factor común para ambas clases y no aportaría a la fórmula de Bayes , la tasa de aciertos obtenida para el set de pruebas de las habichuelas fue de un 81% tal como se muestra en la Tabla 1 utilizando esta única característica de la varianza.

Para el algoritmo que se implementó de igual manera se utilizó Bayes y este consiste en realizar la misma clasificación del primer algoritmo, pero utilizando una segunda característica, para esta aplicación se utilizó la media de los pixeles en la escala de grises, de igual manera se comprobó su distribución gaussiana mediante los histogramas de las Fig. 17 y Fig. 18 y a partir de esto se aplicó la fórmula de las densidades gaussianas multivariable presentada en la Fig. 15, para llevar estas densidades a valores de probabilidad se dividió la densidad gaussiana de cada muestra de pertenecer a cada clase entre la sumatoria de la densidad de pertenecer a cada clase, el resultado de esto sin son valores de probabilidad que suman 1.0 entre ellos, para ambos algoritmos se comparó cual probabilidad era mayor y en base a esto se indicó a que clase pertenecía cada muestra, la tasa de aciertos para el set de pruebas con estas dos características fue de 84% tal como se observa en la Tabla 2.

El tercer algoritmo realizado consistía en nearest neighbor con reducción de base también conocido como condensed nearest neighbor, este algoritmo consistió en encontrar un set de datos para el entrenamiento totalmente compatible con el original que pueda tener menos muestras y mantener cierta similitud dentro de la tasa de aciertos, para este caso una de las permutaciones para condensed nearest neighbor logró incluso superar la tasa de aciertos obtenida con la data original alcanzando el 47% de la tasa de aciertos como se muestra en la Fig. 30 y tal y como se puede observar en las Tabla 4, Tabla 5 y Tabla 6 esta permutación que obtuvo la mejor tasa de aciertos tarda una pequeña fracción más de tiempo en crear este set reducido, tarda unos ms más en clasificarse y ocupa alrededor de 1,000 bytes más en memoria, así que dependiendo de la aplicación si se puede permitir este tiempo extra y espacio adicional puede ser un buen algoritmo a implementar si se busca reducir la complejidad de los datos (reduciendo la cantidad de muestras con este método),

El cuarto algoritmo implementado es el de subset selection, utilizando los métodos tanto fordward como backward selection al mismo dataset utilizado para el algoritmo anterior, la data de mpg clasificada en 10 clases igualmente espaciadas. Para ambos métodos aplicados se redujeron la cantidad de features a 3, es decir solo se seleccionaron los tres mejores features según lo determinado en cada método, por otro lado la tasa de aciertos obtenida para cada uno fue igual, superando ligeramente a la tasa de aciertos obtenida con la data original, la tasa de aciertos obtenida con la reducción de los sets fue de 51.48% tal como se presenta en la Fig. 34, comparado con la tasa de aciertos obtenida con la data original que resultó de un 47%, esto se debe al sesgo que causan algunos features con la data por lo que al intentar clasificar el algoritmo se equivoca por pequeños puntos, esto podría visualizarse mejor planteando una métrica de error con el error mínimo cuadrado, sin embargo, se utilizó directamente la tasa de aciertos para facilitar la compresión de esta información.

Para el quinto algoritmo solo se solicita aplicar los métodos de PCA y LDA a la data original, PCA para reducir la cantidad de features y reconstruir la data original para observa como debido a la transformación de la data se pierde información al tratar de recuperar la original, para fines demostrativos se realizó tanto con PCA como LDA y se obtuvieron resultados muy similares tal como se puede observar en las Fig. 43 y Fig. 44. Por otro lado, se aplicó el método de LDA para transformar la data y a partir de aquí observar el resultado de la clasificación apoyado de KNN para esta última parte la tasa de aciertos obtenida fue de 46.94% tal como se presenta en la Fig. 44

Para el sexto algoritmo se utilizó la herramienta nntool de Matlab para crear las redes neuronales necesarias para comparar, con los métodos anteriores de PCA y LDA, directamente los resultados obtenidos con la red neuronal fueron mejores que los obtenidos al transformar la información con los métodos mencionados anteriormente, a simple vista comparando la Fig. 49 con la Fig. 43, de igual manera para la clasificación de los datos la red neuronal obtuvo una mejor tasa de aciertos alcanzando un 57.3980% como se presentó en la práctica correspondiente superando el 46.94% de la tasa

obtenida utilizando el método de LDA para transformar los features.

Para el séptimo algoritmo se implementó en base al pseudocódigo de la Fig. 50, una red neuronal que aprende a partir de backpropagation, este se implementó para probar que tan bien podría estimar la regresión de la data de mpg utilizada para los otros algoritmos, este presentó un muy buen comportamiento tal como se puede observar en la Fig. 52, donde el error mínimo cuadrado de la regresión tiende a ser menor o igual que 5 con valores muy cercanos a los 5.

Acerca de la investigación acerca del aprendizaje profundo o deep learning, donde se crean estructuras con una analogía simplificada de cómo funcionan las redes neuronales del cerebro y a partir de esta se realiza el aprendizaje de la máquina la mejorar y sustituir en muchas ocasiones los algoritmos convencionales, la consecuencia de este método es la capacidad computacional elevada que requiere, sin embargo, tienen bastante poder debido a que redes ya entrenadas se pueden aplicar mediante servidores en dispositivos sencillos con poco poder de procesamiento, como es el caso de los celulares al utilizar un traductor de idiomas o detector de voz. Estos tienen muchas aplicaciones como la compresión de imágenes, detección de rostros de las personas, entre otras aplicaciones.

En cuanto al aprendizaje por refuerzo o reinforcement learning se presenta otro tipo de algoritmo donde se le indica lo que debe hacer y el resultado y en base a esto mediante distintos cálculos de ensayo y error determina la manera de realizar la operación solicitada, tiene muchas aplicaciones, aunque la más llamativa suele ser la de utilizarse para inteligencia artificial en los videojuegos.

X. Referencias

[1] s. b. data, «sitiobigdata.com,» 27 agosto 2018. [En línea]. Available: https://sitiobigdata.com/2018/08/27/clasificador-de-bayes-estimacion-probabilidad-maxima/#.

[2] Communications, «bbva,» 11 Nov 2019. [En línea]. Available: https://www.bbva.com/es/machine-learning-que-es-y-como-funciona/.

[3] D. a. G. C. Dua, «{UCI} Machine Learning Repository,» 2017. [En línea]. Available: https://archive.ics.uci.edu/ml/citation_policy.html. [Último acceso: 28 06 2020].

[4] MC.AI, «MC.AI,» 19 septiembre 2018. [En línea]. Available: https://mc.ai/chapter-1-k-nearest-neighbours-classifier/.

[5] N. Verma, «web.archieve.org,» 01 11 2019. [En línea]. Available: https://web.archive.org/web/20191101073052/http://www.cs.columbia.edu:80/~verma/classes/ml/lec/lec1_intro_mle_bayes_naive_evaluation.pdf.

[6] E. Alpaydin, «Nonparametric Methods,» de *Introduction to Machine Learning*, Mit Press, 2004, p. 174.

[7] M. Mayo, «Step Forward Feature Selection,» junio 2018. [En línea]. Available: https://www.kdnuggets.com/2018/06/step-forward-feature-selection-python.html. [Último acceso: 01 07 2020].

[8] S. Rawale, «Feature Selection Methods in Machine Learning.,» 01 agosto 2018. [En línea]. Available: https://medium.com/@sagar.rawale3/feature-selection-methods-in-machine-learning-eaeef12019cc. [Último acceso: 01 07 2020].

[9] S. Raschka, «What is the difference between LDA and PCA for dimensionality reduction?,» 2001. [En línea]. Available: https://sebastianraschka.com/faq/docs/lda-vs-pca.html. [Último acceso: 02 07 2020].

[10] Mathworks, «Mathworks.com,» mathworks, 2006. [En línea]. Available: https://www.mathworks.com/help/deeplearning/ref/nntool.html. [Último acceso: 2020].

[11] M. Rouse, «Aprendizaje profundo (Deep Learning),» searchdatacenter.techtarget.com, Abril 2017. [En línea]. Available: https://searchdatacenter.techtarget.com/es/definicion/Aprendizaje-profundo-deep-learning#:~:text=El%20aprendizaje%20profundo%2C%20tambi%C3%A9n%20conocido,obtener%20ciertos%20tipos%20de%20conocimiento.. [Último acceso: 04 Julio 2020].

[12] SmartPanel, «¿Qué es el Deep Learning?,» 10 Abril 2018. [En línea]. Available: https://www.smartpanel.com/que-es-deep-learning/#:~:text=El%20Deep%20Learning%20o%20aprendizaje,fin%20de%20obtener%20ciertos%20conocimientos.&text=Los%20algoritmos%20que%20componen%20un,compuestas%20por%20pesos%20(n%C3%BAmeros).. [Último acceso: 05 Julio 2020].

[13] J. Cepelewicz, «To decode the Brain Scientists Automate the Study of Behavior,» quantamagazine.org, 10 Diciembre 2019. [En línea]. Available: https://www.quantamagazine.org/to-decode-the-brain-scientists-automate-the-study-of-behavior-20191210/. [Último acceso: 02 Julio 2020].

[14] M. Silva, «Aprendizaje por Refuerzo: Introducción al mundo del RL,» medium.com, 2017 Abril 2019. [En línea]. Available: https://medium.com/aprendizaje-por-refuerzo-introducci%C3%B3n-al-mundo-del/aprendizaje-por-refuerzo-introducci%C3%B3n-al-mundo-del-rl-1fcfbaa1c87. [Último acceso: 02 Julio 2020].

[15] S. Raschka, «Linear Discriminant Analysis,» 3 Agosto 2014. [En línea]. Available: https://sebastianraschka.com/Articles/2014_python_lda.html#principal-component-analysis-vs-linear-discriminant-analysis. [Último acceso: 03 Julio 2020].