Aprendizaje Automático

Proyecto final

## Autores:

## Daniel Vidal Soroa

## Juan Diego Peña

# Introducción:

El objetivo de este trabajo será obtener un mecanismo que permita realizar la clasificación de mangos en tres distintas categorías según su calidad. Para ello, se buscará complementar el trabajo anterior utilizando técnicas de agrupamiento, reducción de dimensionalidad y detección de anomalías.

Los datos se seleccionaron de la plataforma kaggle [3] y cuentan con tres clases de mango y 200 imágenes por clase:

* ***Clase extra:*** Usados generalmente para la exportación, no contienen defectos o son lo suficientemente leves para no afectar su aspecto, calidad o conservación en general.
* ***Clase I:*** Destinados al consumo local, de buena calidad con algunos defectos como quemaduras por el sol, rozaduras, etc.
* ***Clase II:*** Reservados normalmente para el procesamiento industrial, cuentan con los mismos defectos de la Clase I a una escala mayor.

A picture containing fruit

Description automatically generated

Figura 1. Clases de mango disponible

## Clasificación no supervisada

El desempeño de un clasificador depende principalmente de la calidad del set de datos usado durante el entrenamiento. Para ello, el algoritmo requiere que las imágenes utilizadas se encuentren previamente etiquetadas. Sin embargo, esto no es siempre posible y muchas veces se requiere realizar el etiquetado de forma manual, lo cual puede ser engorroso.

Una de las mejoras propuestas será contar con una técnica de clasificación no supervisada. Esto, permitiría etiquetar un conjunto de imágenes vírgenes para su posterior utilización para el entrenamiento de un clasificador. En este caso, se compararán distintos algoritmos de agrupamiento con el objetivo de separar las imágenes en tres clases distintas, y comparando este resultado con la etiqueta real.

Para el agrupamiento, en principio, se utilizarán las 600 imágenes aplanadas. Sin embargo, debido a que estas imágenes son tan similares, se utilizarán además el extractor de características HOG y una red neuronal pre-entrenada. En ambos casos, lo que se busca es que estos algoritmos sean capaces transformar las imágenes originales de manera que pudiera ser más fácil de agrupar. Los resultados obtenidos pueden apreciarse en la Tabla 1.

Tabla : Resultados obtenidos durante la clasificación no supervisada

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Objetivo | Extracción de características | Algoritmo de agrupamiento | Exactitud | Tiempo | multiprocesos |
| Comparar redes pre-entrenadas | inception | KMeans | 0,38 | 145,44 | FALSE |
| vgg | KMeans | 0,6 | 234,82 | FALSE |
| xception | KMeans | 0,29 | 188,29 | FALSE |
| resnet | KMeans | 0,18 | 196,06 | FALSE |
| Comparar VGG con HOG y RGB | vgg | KMeans | 0,6 | 270,4 | FALSE |
| hog | KMeans | 0,36 | 12,33 | FALSE |
| rgb | KMeans | 0,27 | 16,22 | FALSE |
| Comparar distintos extractores con distintos algoritmos | hog | decision\_tree | 0,4 | 54,37 | FALSE |
| rgb | decision\_tree | 0,41 | 45,07 | FALSE |
| vgg | decision\_tree | 0,57 | 273,32 | FALSE |
| vgg | spectral | 0,3 | 202,47 | FALSE |
| hog | spectral | 0,33 | 56,63 | FALSE |
| rgb | spectral | 0,3 | 6,53 | FALSE |
|  | hog | spectral | 0,33 | 41,8 | TRUE |

Como se puede apreciar, en primer lugar, se utilizaron cuatro redes pre-entrenadas disponibles en TensorFlow en combinación con el algoritmo de agrupamiento K-means. En este caso, el mejor resultado se obtuvo con VGG para un 60% de acierto en la clasificación. Sabiendo que esta red era la de mejor desempeño, se pasó a comparar su resultado con el extractor de características de HOG y el conjunto de imágenes vírgenes en formato RGB.

Luego, se compararon otros algoritmos de agrupamiento, combinándolos con los extractores antes utilizados con el objetivo de encontrar la mejor combinación. Finalmente, el mejor desempeño apreciado fue utilizando VGG como extractor y K-means como clasificador (Figura 2)

Chart, treemap chart

Description automatically generated

Figura : Matriz de confusión VGG + K-means (random\_state = 42)

Sin embargo, los resultados obtenidos no son suficientemente buenos como para considerarlos satisfactorios. En este caso las clases son demasiado parecidas, y ninguno de los algoritmos de agrupamientos utilizados logra separarlas adecuadamente. El hecho de que K-means logre distinguir la clase I aparentemente bien, parece ser más una consecuencia casual que causal. De hecho, si se cambia el parámetro random\_state del algoritmo, el resultado obtenido es totalmente distinto (Figura 3).

Chart, treemap chart

Description automatically generated

Figura : Matriz de confusión VGG + K-means (random\_state = 0)

## Detección de anomalías

Para la detección de anomalías se ha decidido orientar el problema en el sentido de una planta de empaquetado de mangos de clase extra. En este caso, se asume que la mayoría de las frutas será de la mejor categoría y solo tendrán que discriminarse algunas que hayan escapado a la clasificación previa. Es decir, los mangos de clase extra se considerarán como la clase normal, y cualquier mango con golpes o manchas serán considerados anomalías.

Para la detección de anomalías se utilizará una red convolucional autoencóder. El objetivo, será lograr una red que sea capaz de replicar a la salida la imagen de entrada. Luego, si esta red es entrenada usando solo datos de la clase extra (normal), se podrá extraer un umbral de error para datos de esta clase. Entonces, si se intenta utilizar esta red con un mango de alguna otra clase, el error de predicción será mayor que el umbral generado y podrá decirse que la fruta presenta una anomalía.

La arquitectura propuesta es como se muestra en la Figura 4. La red consta de 6 capas convolucionales y tres capas densas. Además, se han utilizado capas max\_pooling entre capas de convolución consecutivas (Tabla 2). La Figura 5 muestra los resultados obtenidos a la salida de la red para la predicción de una de las imágenes de muestra.

Chart

Description automatically generated with low confidence

Figura : Arquitectura propuesta para la red autoencóder

Tabla : Estructura de la red

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Capa | Filtros | Kernel | Salida | Activación |
| Conv2D (entrada) | 64 | 3x3 | 64x64x64 | Relu |
| Max\_pooling2D |  | 2x2 | 32x32x64 |  |
| Conv2D | 32 | 3x3 | 32x32x32 | Relu |
| Max\_pooling2D |  | 2x2 | 16x16x32 |  |
| Conv2D | 16 | 3x3 | 16x16x16 | Relu |
| Max\_pooling2D |  | 2x2 | 8x8x16 |  |
| Flatten |  |  | 1024 |  |
| Dense |  |  | 32 | Relu |
| Dense |  |  | 1024 | Relu |
| Reshape |  |  | 8x8x16 |  |
| UpSampling2D |  | 2x2 | 16x16x16 |  |
| Conv2D | 32 | 3x3 | 16x16x32 | Relu |
| UpSampling2D |  | 2x2 | 32x32x32 |  |
| Conv2D | 64 | 3x3 | 32x32x64 | Relu |
| UpSampling2D |  | 2x2 | 64x64x64 |  |
| Conv2D | 3 | 3x3 | 64x64x3 | Linear |

A picture containing graphical user interface

Description automatically generated

Figura : *Imagen de entrada y salida predicha por la red neuronal*

Los resultados obtenidos fueron bastante cercanos a la lógica. En este caso existe mucha similitud entre la clase 1 y la clase extra, por lo que es entendible que algunos datos de la clase extra sean detectados como anomalías y varios datos de la clase I como frutas sin defectos. Por otra parte, los datos de la clase II son algo más distintos a la clase normal y por este sentido son más fácil de discriminar.

La Figura 6 muestra tres resultados distintos de la red para distintos episodios de entrenamiento (50, 100 y 150 de arriba a abajo). En primer caso se aprecia como la red detecta aproximadamente un 9% de los datos normales de entrenamiento como anomalías, para una sensibilidad (recall) del 91%. Mientras, confunde un 38% de los datos de la clase I como normales y solo uno de la clase II para una especificidad del 80%. La métrica f1-score con esta prueba fue del 92%.

Text

Description automatically generated

Figura : Resultados obtenidos para la detección de anomalías con 50, 100 y 150 episodios

La segunda y tercera gráficas muestran los resultados obtenidos para 100 y 150 epochs. El objetivo del aumento de episodios es tratar de disminuir, mediante un mayor entrenamiento, el error cometido para detectar los datos de la clase normal. Esto debería hacer que sea más fácil que datos con anomalías superen este umbral. Sin embargo, pese a que en las gráficas dos y tres se aprecia como el umbral se va ajustando hasta lograr aislar por completo los datos anómalos, también comienza a ocurrir un sobre entrenamiento (Figura 7).

Esto significa que la red se sobre ajusta para recrear sólo los datos de entrenamiento y luego, cuando debe predecir otros datos normales que no ha visto aún, muchas veces comete errores que la hacen detectarlos como anomalías. En la segunda gráfica se aprecia como, al utilizar 100 episodios se logra aislar por completo los datos de la clase II. Mientras, con 150 episodios ya se obtiene una precisión del 100 %, es decir, todos los datos anómalos son detectados. Sin embargo, como resultado del sobre entrenamiento muchos datos normales se detectan como fallos para un incremento del 9 al 27% y 38 % respectivamente (Tabla 3).

Tabla : Resultados obtenidos para distintos episodios de entrenamiento

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Epochs | Lote | Falsas alarmas | Clase 1 < Umb. | Clase 2 < Umb. | Sens. | Spec. | Prec. | f1-Score |
| 50 | 32 | 0,09 | 0,38 | 0,02 | 0,91 | 0,80 | 0,92 | 0,92 |
| 100 | 40 | 0,27 | 0,08 | 0 | 0,73 | 0,96 | 0,98 | 0,84 |
| 150 | 50 | 0,38 | 0 | 0 | 0,62 | 1,00 | 1,00 | 0,77 |

Como se puede apreciar en los resultados obtenidos, generar un umbral para a detección de anomalías en este caso es muy difícil debido a la similitud entre las clases. En este sentido, debe sacrificarse o la precisión o la sensibilidad. Es decir, si se fuerza el umbral para detectar todas las frutas con golpes como tal, se detectarán y discriminarán muchas frutas intactas también. Por el contrario, si se utiliza un umbral más relajado, se discriminarán menos frutas buenas, pero se tomarán como premium hasta un 38% de frutas pertenecientes a la clase I y alguna que otra de la clase II.

Sin embargo, si se tuviese que escoger una de estas soluciones se tomaría la primera. En primer lugar, esta aplicación debe priorizar disminuir el descarte de frutas premium ya que estas son dedicadas a la exportación. Si existe alguna fruta de clase I con pocos golpes que puede ser tomada como clase extra esto no representaría un gran inconveniente. Además, se entiende que esta aplicación procesará un gran volumen de frutas premium y solo algunas vendrán dañadas. Es decir, el parámetro de la sensibilidad en este caso es más relevante, ya que se prefiere cometer la menor cantidad de errores posibles durante la clasificación de los datos verdaderos. El parámetro f1-score, reflejado en la Tabla 3 también respalda esta decisión.

A picture containing graphical user interface

Description automatically generated

Figura 7: Curva de entrenamiento vs prueba para 50 y 150 epochs

## Reducción de dimensión

En el trabajo anterior se redujo el tamaño de las imágenes de la resolución original a 32x32x3 para reducir el costo computacional. Sin embargo, siempre que se reduce la dimensión de una imagen se pierde información al promediar los píxeles, esta información, pudiera ser importante luego para mejorar el desempeño del clasificador.

Por esta razón, para la reducción de la dimensión en este trabajo se intentará utilizar una red autoencóder. La lógica que sustenta esta decisión es que las redes autoencóder aprenden a extraer la información más relevante de una imagen para ser capaces de reproducirlas a la salida. De este modo, si en el cuello de botella de la red se obtiene un tensor de dimensión 32x32x3, entonces este tendría la misma dimensión que si se redujera el tamaño de la imagen. Sin embargo, esta contará con las características más importante de la imagen de entrada.

La estructura de la red diseñada aparece representada en la Figura 8. La imagen de entrada tendrá una dimensión de 128x128x3. La red contará con una capa convolucional de entrada con 64 filtros. Luego, le seguirán 5 capas ocultas como se muestra en la Figura 8 y una capa extra para obtener a la salida una imagen de 128x128x3.

A picture containing graphical user interface

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Capa | Filtros | Kernel | Salida | Activación |
| Conv2D (entrada) | 64 | 3x3 | 128x128x64 | Relu |
| Max\_pooling2D |  | 2x2 | 64x64x64 |  |
| Conv2D | 32 | 3x3 | 64x64x32 | Relu |
| Max\_pooling2D |  | 2x2 | 32x32x32 |  |
| Conv2D | 3 | 3x3 | 32x32x3 | Relu |
| UpSampling2D |  | 2x2 | 64x64x3 |  |
| Conv2D | 32 | 3x3 | 64x64x32 |  |
| UpSampling2D |  | 2x2 | 128x128x32 |  |
| Conv2D | 64 | 3x3 | 128x128x64 | Relu |
| Conv2D | 3 | 3x3 | 128x128x3 | Linear |

Figura : Estructura de la red autoencóder

Los resultados obtenidos no son tan buenos como se esperaba. La Tabla 4 muestra que para todos los clasificadores probados (Random Forest, LDA o CNN), la exactitud es siempre mejor reduciendo la dimensión de la imagen de forma tradicional. El desempeño al utilizar el autoencóder no dista mucho del resultado obtenido al redimensionar, pero dado que este proceso requiere el entrenamiento de una red convolucional, no merece la pena agregar esa complejidad al proceso.

Tabla : Resultados obtenidos con distintas técnicas de reducción

|  |  |  |
| --- | --- | --- |
| Clasificador | Técnica de reducción | Exactitud |
| LDA | **Autoencóder** | **0,84** |
| LDA | **Redimensión** | **0,84** |
| LDA | PCA | 0,33 |
| RForest | Autoencóder | 0,86 |
| RForest | **Redimensión** | **0,88** |
| RForest | PCA | 0,71 |
| CNN | Autoencóder | 0,64 |
| CNN | **Redimensión** | **0,77** |

Graphical user interface

Description automatically generated with medium confidence

Figura : Comparación de la imagen obtenida al reducir la dimensión de la imagen con el autoencóder y de forma tradicional.

## Clustering

Clustering es una técnica de aprendizaje no supervisada que se encarga de dividir todo el conjunto de datos en *clusters* o grupos a partir sus características. Se decidió implementar técnicas de aprendizaje no supervisado con el fin de tener una referencia del desempeño cuando se tienen datos etiquetados y cuando no. Se usarán las mismas métricas empleadas para los anteriores clasificadores supervisados.

Además, el clustering se combinó junto con técnicas de reducción de dimensión, para este caso PCA. Posteriormente se compararon tres algoritmos de clustering, K-means, arboles de decisión y *spectral clustering.*

Las técnicas de reducción de dimensión se usaron en todo caso como pre procesamiento de los datos para extraer sus características y posteriormente usar los métodos de clustering como clasificadores no supervisados.

A continuación, se muestran los resultados de cada modelo de agrupamiento y su análisis de hiperparametros.

#### Spectral clustering

El espacio de hiperparametros optimizados es el siguiente:

|  |  |
| --- | --- |
| Parámetros | Parámetros propuestos |
| eigen\_solver | arpack, lobpcg, amg |
| assign\_labels | kmeans, discretize |

Tabla 5. Espacio de hiperparametros.

A continuación, se muestran las métricas usando diferentes combinaciones de hiperparametros:

Chart, box and whisker chart

Description automatically generated

Figura 10. Comparación de hiperparametros

Chart, box and whisker chart

Description automatically generated

Figura 11. Comparación de hiperparametros.

Chart, box and whisker chart

Description automatically generated

Figura 12. Comparación de hiperparametros.

Chart, box and whisker chart

Description automatically generated

Figura 13. Comparación de hiperparametros.

Finalmente, haciendo un análisis de hipótesis se llega a la conclusión que lo modelos son iguales, por lo que se decide usar la siguiente combinación par el entrenamiento final.

|  |  |
| --- | --- |
| eigen\_solver | assign\_labels |
| Lobpcg | discretize |

Tabla 6. Mejor combinación de hiperparametros.

##### Resultados:

La prueba final del modelo se realiza con los datos de validación y la mejor combinación de hiper parámetros, a continuación, se muestran los resultados:

Chart

Description automatically generated

Figura 14. Matriz de confusión

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Sensibilidad* | Precisión | F1 | Exactitud | Especificidad |
| 0.33 | 0.35 | 0.33 | 0.33 | 0.67 |

Tabla 7. Métricas finales en datos de validación.

#### K-means

El espacio de hiperparametros consta de dos, init, el primero se usa para establecer la forma de inicializar la búsqueda. Algorithm establece que algoritmo se va a usar en el método.

|  |  |
| --- | --- |
| Parámetros | Parámetros propuestos |
| Init | [K-means++,random] |
| algorithm | [auto, full, elkan] |

Tabla 8. Espacio de hiperparametros.

A continuación, se muestra el resultado de los hiperparametros:

Chart, box and whisker chart

Description automatically generated

Figura 15. Comparación de hiperparametros.

Chart, box and whisker chart

Description automatically generated

Figura 16. Comparación de hiperparametros.

Chart, box and whisker chart

Description automatically generated

Figura 17. Comparación de hiperparametros.

Chart, box and whisker chart

Description automatically generated

Figura 18. Comparación de hiperparametros.

Finalmente, haciendo un análisis de hipótesis se llega a la conclusión que lo modelos son iguales, por lo que se decide usar la siguiente combinación par el entrenamiento final.

|  |  |
| --- | --- |
| algorithm | init |
| full | random |

Tabla 9. Mejor configuración de hiperparametros.

##### Resultados:

Chart, treemap chart

Description automatically generated

Figura 19. Matriz de confusión.

La prueba final del modelo se realiza con los datos de validación, a continuación, se muestran los resultados:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Sensibilidad* | Precisión | F1 | Exactitud | Especificidad |
| 0.57 | 0.48 | 0.50 | 0.57 | 0.79 |

Tabla 10. Métricas finales en datos de validación.

#### Random tree

El espacio de hiperparametros optimizados es el siguiente:

|  |  |
| --- | --- |
| Parámetros | Parámetros propuestos |
| affinity | euclidean, l1, l2, manhattan, cosine |
| linkage | complete,average,single |

Tabla 11. Espacio de hiperparametros.

Chart, box and whisker chart

Description automatically generated

Figura 20. Comparación de hiperparametros.

Chart, box and whisker chart

Description automatically generated

Figura 21. Comparación de hiperparametros.

Chart, box and whisker chart

Description automatically generated

Figura 22. Comparación de hiperparametros.

Chart, box and whisker chart

Description automatically generated

Figura 23. Comparación de hiperparametros.

Finalmente, haciendo un análisis de hipótesis se llega a la conclusión que lo modelos son iguales, por lo que se decide usar la siguiente combinación par el entrenamiento final.

|  |  |
| --- | --- |
| affinity | linkage |
| manhattan | complete |

Tabla 12. Combinación optima de hiperparametros.

##### Resultados:

Chart

Description automatically generated

Figura 24. Matriz de confusión.

La prueba final del modelo se realiza con los datos de validación, a continuación, se muestran los resultados:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Sensibilidad* | Precisión | F1 | Exactitud | Especificidad |
| 0.37 | 0.24 | 0.29 | 0.38 | 0.68 |

Tabla 13. Métricas finales en datos de validación.

## Optimización de tiempo de ejecución

Una de las etapas de más consumo de recursos dentro del proceso de evaluación de modelos, es la optimización de hiperparametros mediante validación cruzada. Es por eso por lo que resulta importante un análisis del impacto que tiene la característica del método GridSearchCV de skitlearn, el cual te permite escoger el número de hilos para ejecutar la optimización de hiperparametros.

Como base se utilizó el clasificador rondón foresta para medir el impacto. A continuación, se muestran los resultados del tiempo de ejecución promedio por cada división de datos. Sin embargo, dicho método también mide los tiempos de ejecución y los almacena en los resultados finales que fueron exportados a Excel.

Chart, box and whisker chart

Description automatically generated

Figura 25. comparación de tiempos de ejecución.

A manera de optimizar el tiempo de ejecución en todo el análisis, se ejecutó el código con la mayor cantidad de hilos posibles.

También, se propone el uso de multiproceso en la fase de preprocesado de datos, que en el caso del presente estudio se basó en aumento de datos. Continuación se muestran los resultados:

Chart, bar chart

Description automatically generated

Figura 26. comparación de tiempos en aumento de datos.

## Comparación de modelos y selección

Luego de optimizar cada clasificador, se debe seleccionar el mejor. La métrica encargada realizar la comparación es la exactitud. Por cada modelo se realiza una prueba estadística para validar si los datos son iguales o no.

Chart, box and whisker chart

Description automatically generated

Figura 27. comparación de modelos no supervisados

El contraste de hipótesis arroja que los modelos son iguales, por lo que se recomienda el uso de algoritmos con menor costo de cómputo.

# Conclusiones

Los métodos de agrupamiento o *Clustering* no mostraron un desempeño comparable con los métodos de aprendizaje supervisado, por lo que se recomienda el uso de este solamente como un preprocesamiento de datos.

Es importante la optimización de los recursos de computación cuando se trata de aprendizaje automático, el manejo de una gran cantidad de datos conlleva a un uso significativos de recursos. El uso de procesos multi hilo puede reducir drásticamente el tiempo total de entrenamiento.