|  |
| --- |
| Título |
| Clasificación de mangos mediante técnicas de aprendizaje |
| Resumen |
| En este trabajo se analiza el desempeño de varias técnicas de clasificación para categorizar mangos en tres clases diferentes de acuerdo con su integridad física. Los métodos seleccionados son Análisis discriminante lineal, K vecinos más cercanos, Bosques aleatorios y Redes neuronales de convolución. Entre las redes neuronales, además, se evalúan tres arquitecturas de entre las más utilizadas en el estado del arte para problemas de clasificación de imágenes: ResNet, MobileNet y Vgg16. |

|  |
| --- |
| Autores |
| Daniel Vidal Soroa |
| Juan Diego Peña N |
| 19/03/2022 |

|  |  |  |  |
| --- | --- | --- | --- |
| Rev. | Descripción | Elaborado | Aprobado |
| 01 | Primera edición |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Índice

[1. Introducción 3](#_Toc99300799)

[1.1. Objeto 3](#_Toc99300800)

[1.2. Alcance 3](#_Toc99300801)

[Normas y referencias 3](#_Toc99300802)

[1.3. Definiciones y abreviaturas 3](#_Toc99300803)

[2. Descripción general 4](#_Toc99300804)

[3. Diseño e implementación 4](#_Toc99300805)

[3.1. Preprocesamiento de datos 5](#_Toc99300806)

[3.2. Evaluación de modelos y optimización de hiperparámetros 5](#_Toc99300807)

[3.3. Comparación de modelos y selección 18](#_Toc99300808)

[4. Conclusiones 18](#_Toc99300809)

[5. ANEXOS: 19](#_Toc99300810)

[5.1. ANEXO I : ANALISIS DE HIPER PARAMETROS K-NN 19](#_Toc99300811)

[5.2. ANEXO I : ANALISIS DE HIPER PARAMETROS RF 23](#_Toc99300812)

# Introducción

## Objeto

El objetivo de este proyecto es implementar técnicas de aprendizaje automático supervisado sobre un conjunto de datos definido. Cada método se evaluará escogiendo las métricas de error más adecuadas, para luego optimizar sus hiperparámetros y determinar el de mayor rendimiento.

## Alcance

El proyecto incluye la aplicabilidad de modelos ya existentes en las librerías scikit-learn [1] y keras [2]. Así como el uso datos de dominio público, el proyecto no contempla el uso de técnicas de aprendizaje no supervisado y se ceñirá únicamente a las técnicas: LDA, KNN, RF y CNN.

## Normas y referencias

|  |  |
| --- | --- |
| [1] | «Scikit learn,» [En línea]. Available: https://scikit-learn.org/stable/. |
| [2] | «TensorFlow,» [En línea]. Available: https://www.tensorflow.org/guide/keras?. |

## Definiciones y abreviaturas

**ML** Machine learning (*Aprendizaje automático)*

**LDA** Linear Discriminant Analysis (Análisis *discriminante lineal)*

***KNN*** K-Nearest Neighbor (K-Vecinos)

***RF*** *Random Forest (Bosques aleatorios)*

***CNN*** Convolutional neural networks (Redes neuronales convolucionales)

# Descripción general

El proyecto se motiva en servir como una herramienta base para países productores de mango al eliminar un control de calidad manual que posibilita el daño de la fruta y abarata el costo de otros métodos de control como el uso resonancias magnéticas. Además, debido a la migración rural negativa y el corto período de conservación de esta fruta, cada año se pierden millones de toneladas debido a falta de mano de obra y errores de clasificación.

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 . Clases de mango disponible

# Diseño e implementación

El proceso de implementación de las técnicas de aprendizaje se basa en el propuesto en la guía de usuario de scikit-learn. Este se basa en un flujo de trabajo típico para este tipo de aplicaciones.

En primer lugar, se dividen los datos en dos grupos, uno de validación y otro de entrenamiento. Posteriormente, se optimizan los hiperparámetros de cada modelo para consecutivamente elegir el modelo más adecuado. Tanto la selección de hiperparámetros como de modelo se hace mediante una prueba estadística. Se muestra un diagrama del flujo en el que se basa el proyecto:

Diagram

Description automatically generated

Figura . Flujo de trabajo típico

## Preprocesamiento de datos

Primeramente, se definió un tamaño de imagen de 32x32 pixeles con el fin de reducir el tiempo de procesamiento y extraer de manera más eficiente características relevantes de las imágenes. Seguidamente, utilizando la librería de tensor Flow, se generaron 1800 imágenes nuevas para cada clase. Estas imágenes generadas se usan como entrenamiento y las 200 imágenes originales se dejan como test de validación. Los datos también fueron reorganizados aleatoriamente para que el orden no sea un factor determinante para los clasificadores.

A picture containing shape

Description automatically generated

Figura . Ejemplo de imágenes generadas y redimensionadas.

## Evaluación de modelos y optimización de hiperparámetros

Las métricas utilizadas para la evaluación de todos los modelos serán, la exactitud, sensibilidad, precisión, f1-score, especificidad y área bajo la curva ROC. Con estas métricas y en conjunción con el método de validación cruzada se hará una evaluación objetiva de cada clasificador. Ya que el conjunto de datos es relativamente mediano (6000 imágenes), los datos de entrenamiento se dividirán en 5 grupos diferentes.

### Análisis discriminante lineal (LDA)

Esta técnica se basa en encontrar el hiperplano que mejor separe los datos proporcionados. Ya que este método cuenta con pocos hiperparámetros, se realizó un análisis de los tipos de *solucionadores (solver):*

|  |  |
| --- | --- |
| Parámetros | Parámetros propuestos |
| Solucionador | [svd, lsqr] |

Tabla . Hiperparámetros analizados.

#### Resultado hiperparámetros

A continuación, se muestran los resultados obtenidos para cada métrica de error propuesta evaluando los dos tipos de solucionadores:

A picture containing graphical user interface

Description automatically generated

Figura . Resultados sensibilidad (recall).

Chart, box and whisker chart

Description automatically generated

Figura . Resultados precisión.

A picture containing graphical user interface

Description automatically generated

Figura . Resultados F1.

A picture containing graphical user interface

Description automatically generated

Figura Resultados exactitud.

Graphical user interface

Description automatically generated with low confidence

Figura . Resultados, área bajo la curva ROC.

Aunque en los resultados se evidencia un claro mejor desempeño por parte del solucionador *svd (descomposición de valores singulares),* se realizó una prueba estadística de Wilcoxon que comprueba que los modelos son diferentes estadísticamente, por lo que el entrenamiento final se realizó con el solucionador *svd.*

#### Evaluación final del modelo

Luego de determinar la mejor combinación de hiperparámetros, mediante validación cruzada, se hace una validación final con los datos dispuestos para este propósito. A continuación, los resultados:

Chart, treemap chart

Description automatically generated

Figura . Matriz de confusión para el clasificador LDA.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Sensibilidad* | Precisión | F1 | Exactitud | AUC | Especificidad |
| 0.945 | 0.945 | 0.945 | 0.945 | 0.972 | 0.972 |

Tabla . Resultados en datos de validación.

### K-Vecinos (KNN)

El clasificador KNN se encarga de almacenar los datos de entrenamiento y realiza la clasificación basándose en una votación de vecinos. Cada vez que se realiza una predicción, se le asigna una clase a la entrada basado en la clase preasignada de los vecinos más cercanos.

Uno de los algoritmos usados por el clasificador para escoger el vecino más cercano es *KD Tree*, que usa como uno de sus parámetros el *Leaf size* y será tomado en cuenta para el análisis.

Otro hiperparámetro a tener en cuenta es el número de vecinos empleado para realizar la clasificación, tomando en cuenta la recomendación de que este número sea menor a la raíz cuadrada del número de datos, se definió un rango de vecinos a evaluar.

Finalmente se evalúa también el modelo con dos definiciones de distancia mínima diferente, la euclidiana y la de Minkowski. A continuación, se muestra u resumen de los parámetros utilizados en la optimización:

|  |  |
| --- | --- |
| Parámetros | Parámetros propuestos |
| Leaf size | [1, 11, 21, 31, 41] |
| K-vecinos | [5, 13, 21, 29, 37] |
| Tipo de distancia[[1]](#footnote-2) | [Manhattan, Euclidiana] |

Tabla . Hiperparámetros a optimizar.

#### Resultados:

Ya que se tomaron en cuenta 250 combinaciones de hiperparámetros, se mostrará solamente los 5 mejores resultados de cada métrica, en el anexo I se muestran las gráficas completas.

Chart, box and whisker chart

Description automatically generated

Figura . Resultados *sensibilidad*

Chart, box and whisker chart

Description automatically generated

Figura . Resultados precisión.

Chart, box and whisker chart

Description automatically generated

Figura . Resultados f1.

Chart, box and whisker chart

Description automatically generated

Figura . Resultados exactitud.

Chart, box and whisker chart

Description automatically generated

Figura . Resultados, área bajo la curva ROC.

Ya que el comportamiento de todas las métricas de error exhibe un comportamiento similar, se hace uso del *accuracy* o exactitud como métrica para realizar un análisis de hipótesis y decidir el mejor modelo. Cabe destacar por lo que la exactitud es un buen indicador para el análisis de hipótesis.

Usando una prueba de Wilcoxon, se compara la exactitud de cada una de las cinco divisiones de la validación cruzada. Dando como resultado que los modelos son iguales, de manera que se entrena el modelo más eficiente computacionalmente con los siguientes parámetros:

|  |  |  |
| --- | --- | --- |
| *Leaf size* | *# Vecinos* | *Tipo de distancia* |
| 41 | 5 | Manhattan |

Tabla . Parámetros finales del modelo.

#### Evaluación final del modelo

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

Chart

Description automatically generated

Figura . Matriz de confusión, modelo KNN.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Sensibilidad* | Precisión | F1 | Exactitud | AUC | Especificidad |
| 0.892 | 0.895 | 0.891 | 0.892 | 0.980 | 0.946 |

Tabla . Métricas finales en datos de validación.

### Bosque aleatorio (RF)

Random forest o bosque aleatorio, realiza la clasificación a través de una serie de pruebas a cada variable hasta alcanzar su clasificación, en cada prueba el conjunto de datos se divide en un cierto número, la cantidad de divisiones son controladas por el hiperparámetro *min\_samples\_split* el cual se usará para el análisis.

El número máximo de variables que se evalúa en cada prueba es otro hiperparámetro escogido para el análisis, así como el *boostrap* y *mean\_samples\_leaf*

|  |  |
| --- | --- |
| Parámetros | Parámetros propuestos |
| Mínimo de divisiones | [2,5,10] |
| Mínimo de hojas | [1,2,4] |
| Bootstrap | [Falso, Verdadero] |
| Máximo de variables | [Automático, raíz cuadrada] |

Tabla . Combinación de hiperparámetros a evaluar.

#### Resultados

Ya que se entrenaron mas de 180 combinaciones, se muestran solo los mejores dos resultados de cada métrica, los resultados completos se encuentran en el anexo I

A picture containing text

Description automatically generated

Figura . Resultados de sensibilidad.

A picture containing diagram

Description automatically generated

Figura . Resultados precisión.

Text

Description automatically generated with medium confidence

Figura . Resultados f1.

Text

Description automatically generated with medium confidence

Figura . Resultados exactitud.

A picture containing text

Description automatically generated

Figura . Resultados área bajo la curva ROC.

Aplicando de nuevo la prueba estadística de Wilcoxon, se llega al resultado que los modelos son iguales, por lo que se usa los parámetros que agilicen la velocidad de cómputo:

|  |  |  |  |
| --- | --- | --- | --- |
| *Boostrap* | *Máximo de variables* | *Mínimo de hojas* | *Mínimo de divisiones* |
| Falso | Automático | 2 | 5 |

Tabla . Modelo final.

#### Evaluación final del modelo

Luego de escoger el mejor modelo, se evalúa el desempeño con los datos de validación:

Chart

Description automatically generated

Figura . Matriz de confusión con los datos de validación.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Sensibilidad* | Precisión | F1 | Exactitud | AUC | Especificidad |
| 0.982 | 0.982 | 0.982 | 0.982 | 0.999 | 0.991 |

Tabla . Métricas en datos de validación.

### Redes neuronales convencionales (CNN)

Los clasificadores de redes artificiales convolucionales se basan en la aplicación de filtros sucesivos a imágenes, que facilitan la extracción de características que serán usadas luego para realizar la clasificación. Sin embargo, el desempeño de estas técnicas depende en gran medida de la calidad del proceso de entrenamiento. Específicamente, se recomienda contar con un conjunto numeroso de muestras de entrenamiento.

En este caso, incluso luego de haber aumentado el conjunto de datos original, no se disponían de datos suficientes para realizar un entrenamiento de calidad. Por este motivo, se decidió utilizar un concepto denominado transferencia de conocimiento, a través del cual se toma una red neuronal pre-entrenada, y se sustituyen sus capas finales por unas adecuadas al problema de clasificación en cuestión.

Para este trabajo, se tomaron en cuenta diferentes arquitecturas predefinidas en la librería keras. Los criterios seguidos para la selección de estas arquitecturas fueron la velocidad de cómputo, ya que esta aplicación se espera pueda desarrollarse en tiempo real, y la exactitud de la clasificación. La Tabla 9 muestra los modelos seleccionados y sus distintas configuraciones en cuanto a su profundidad y número de parámetros (pesos) totales.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Modelo | Exactitud | Parámetros | Profundidad | Tiempo (ms) por paso |
| Vgg16 | 90.1 | 138.4 M | 16 | 69.5 |
| ResNet50 | 92.1 | 25.6 M | 107 | 58.2 |
| MobileNet | 89.5 | 4.3 M | 55 | 22.6 |

Tabla : Modelos CNN seleccionados

#### Resultados:

Para la optimización de estas arquitecturas, se utilizaron tres combinaciones distintas de capas de salida para analizar cuáles presentan mejores resultados a la hora de transformar el conocimiento provisto por los modelos pre-entrenados en la clasificación final. Las combinaciones propuestas son como se muestran a continuación:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Modelo 1 | Modelo 2 | Modelo 3 |
| Capa 1 | Aplanamiento | Aplanamiento | Aplanamiento |
| Capa 2 | Densa Relu (64 neuronas) | Densa Relu (100 neuronas) | Densa Relu (128 neuronas) |
| Capa 3 | Densa Softmax (3 neuronas) | Dropout (0.2) | Dropout (0.5) |
| Capa 4 |  | Densa Softmax (3 neuronas) | Normalización de batch |
| Capa 5 |  |  | Densa Softmax (3 neuronas) |

Tabla : Arquitecturas propuestas para capas de salida

A continuación, se muestran algunos de los mejores resultados obtenidos combinando los tres modelos pre-entrenados con las configuraciones de salida propuestas. Como es posible apreciar, los mejores resultados se obtuvieron con la red Vgg16. Además, la arquitectura de salida más eficiente fue la más sencilla, compuesta por una capa densa de 64 neuronas y función de activación Relu y capa de salida de tres neuronas con activador Softmax.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Modelo base | Modelo final | Param | Epoch | Batch size | Exact. | Spec. | f1\_scr | AUC |
| vgg16 | 1 | 33027 | 200 | 60 | 0,95 | 0,98 | 0,95 | 0,98 |
| vgg16 | 1 | 33027 | 250 | 60 | 0,94 | 0,97 | 0,94 | 0,99 |
| vgg16 | 2 | 66051 | 300 | 50 | 0,92 | 0,96 | 0,92 | 0,98 |
| vgg16 | 3 | 52003 | 100 | 50 | 0,84 | 0,93 | 0,84 | 0,97 |

Tabla : Mejores resultados obtenidos

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Modelo base | Modelo final | Param | Epoch | Batch size | Exact. | Spec. | f1\_scr | AUC |
| mobilnet | 1 | 65795 | 200 | 60 | 0,69 | 0,89 | 0,68 | 0,87 |
| mobilnet | 2 | 131587 | 250 | 60 | 0,68 | 0,88 | 0,65 | 0,86 |
| resnet | 1 | 131331 | 300 | 50 | 0,66 | 0,9 | 0,63 | 0,85 |
| resnet | 1 | 131331 | 200 | 60 | 0,63 | 0,91 | 0,56 | 0,82 |
| resnet | 1 | 131331 | 200 | 60 | 0,63 | 0,89 | 0,6 | 0,82 |

Tabla : Resultados obtenidos con modelos ResNet y MobileNet

Chart

Description automatically generated

Figura . Arquitectura VGG16.

Chart, line chart

Description automatically generated

Figura . Arquitectura ResNet50

Chart, line chart, scatter chart

Description automatically generated

Figura : Arquitectura MobileNet

#### Evaluación final del modelo

A continuación, se muestran las métricas finales de la arquitectura elegida (Vgg16 + modelo de salida 1) utilizando los datos de prueba:

Chart, treemap chart

Description automatically generated

Figura . Matriz de confusión modelo CNN

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Sensibilidad* | Precisión | F1 | Exactitud | AUC | Especificidad |
| 0.95 | 0.95 | 0.95 | 0.95 | 0.98 | 0.98 |

Tabla . Métricas finales modelo CNN.

### Agrupamiento o *clustering*

Clustering es una técnica de aprendizaje no supervisada que se encarga de dividr todo el conjunto de datos en *clusters* o grupos a partir sus características. Se decidió implementar técnicas de aprendizaje no supervisado, más específicamente de agrupamiento. 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.

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 . Espacio de hiperparametros.

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

Chart, line chart

Description automatically generated

Figura . Rendimiento del modelo.

Finalmente se escoge la mejor combinación:

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

Tabla . 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 . Matriz de confusión

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

Tabla 16. 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 . Espacio de hiperparametros.

A continuación, se muestra el resultado de los hiperparametros:

Chart, line chart

Description automatically generated

Figura . Desempeño del espacio de hiperparametros

Finalmente se decide optar por la siguiente configuración.

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

Tabla . Mejor configuración de hiperparametros.

##### Resultados:

Chart, treemap chart

Description automatically generated

Figura . 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 19. 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 . Espacio de hiperparametros.

Chart

Description automatically generated

Figura . Desempeño de espacio de hiperparametros.

Finalmente se decide optar por la siguiente configuración

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

Tabla . Combinación optima de hiperparametros.

##### Resultados:

Chart

Description automatically generated

Figura . 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 22. Métricas finales en datos de validación.

### 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 32: Arquitectura propuesta para la red autoencóder

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

Tabla 23: Estructura de la red

A picture containing graphical user interface

Description automatically generated

Figura 33: *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 34: 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.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

Tabla 24: Resultados obtenidos para distintos episodios de entrenamiento

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 35: 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 36: 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.

|  |  |  |
| --- | --- | --- |
| 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** |

Tabla 25: Resultados obtenidos con distintas técnicas de reducción

Graphical user interface

Description automatically generated with medium confidence

Figura 37: Comparación de la imagen obtenida al reducir la dimensión de la imagen con el autoencóder y de forma tradicional.

## Optimización de tiempo de ejecución

Una de las etapas de mas 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 numero de hilos para ejecutar la optimización de hiperparametros.

Como base se utilizo el clasificador rondón foresta para medir el impacto. A continuación, se muestran los resultados del tiempo de ejecución promedio por cada división de datos.

Chart, box and whisker chart

Description automatically generated

Figura . 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.

# Conclusiones

El objetivo de este trabajo era seleccionar un algoritmo de clasificación que fuese capaz de discriminar entre tres clases de mango de acuerdo con su presencia física. Para ello, se han evaluado los clasificadores LDA, KNN, RF y CNN. Luego de ajustar los hiperparámetros de cada uno de estos métodos, se llega a la conclusión de que Random Forest es el que mejor resultados ofrece.

Este modelo es el más exacto y, además es un algoritmo relativamente rápido y fácil de parametrizar. Así mismo, la repetitividad con la que este algoritmo brindó buenos resultados también es destacable. Además, analizando la matriz de confusión obtenida durante la etapa de prueba, es posible apreciar que el clasificador es especialmente bueno distinguiendo entre las tres clases y, sobre todo, confunde muy pocas veces la clase 2 y la clase extra.

Esto es importante para la comercialización, ya que significa que muy pocos mangos no adecuados serán dedicados a la exportación, y muy pocas frutas premium serán enviadas a su procesamiento industrial.

Respecto a las redes CNN, ha de decirse que mostraron mucho potencial, especialmente la red creada a partir de la arquitectura Vgg16. Durante su evaluación, las redes mostraron que podían mejorar su desempeño a partir de un aumento de las iteraciones (epochs), pudiendo llegar a cotas comparables incluso con RF. Sin embargo, para esta aplicación se prefiere el método antes mencionado, ya que este requiere menos parametrización y entrenamiento.

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 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.

# ANEXOS:

## ANEXO I : ANALISIS DE HIPER PARAMETROS K-NN

Chart

Description automatically generated with medium confidence

Chart, box and whisker chart

Description automatically generated

Chart

Description automatically generated

Chart

Description automatically generated

Chart, box and whisker chart

Description automatically generated

## ANEXO I : ANALISIS DE HIPER PARAMETROS RF

Chart

Description automatically generated

Chart, box and whisker chart

Description automatically generated

Chart

Description automatically generated

Chart, box and whisker chart

Description automatically generated

Chart

Description automatically generated

1. Representado en scikit-learn como p (1 =Manhattan 2 = Euclidiana) [↑](#footnote-ref-2)