Facultad de Ingeniería - Universidad Nacional de Cuyo

Clasificación de frutas mediante visión artificial usando k-means y k-nn

Estudiante:
Jerónimo Pérez Serpa

Resumen:

El presente trabajo sobre visión artificial y machine learning fue realizado como trabajo final de la cátedra Inteligencia Artificial I, de la Facultad de Ingeniería de la Universidad Nacional de Cuyo.

Se desarrolló un sistema de reconocimiento de frutas mediante el uso de técnicas de procesamiento de imagen e inteligencia artificial. Para ello se utilizó la última versión del dataset "Fruit 360" y se utilizaron fotografías de bananas, naranjas y limones sobre las cuales se realizó un trabajo de pre-procesamiento de las imágenes para disminuir el ruido y acentuar características geométricas para la posterior extracción de las mismas.

Para clasificar las imágenes se utilizan algoritmos de aprendizaje K-means y K-nn, y se hace un estudio de desempeño comparativo entre los dos algoritmos. El lenguaje de programación utilizado a lo largo del proyecto es Python.

Índice

1. Introducción

Con la intención de agilizar el proceso de cobro en una caja de supermercado se propone desarrollar un sistema de reconocimiento de frutas por visión artificial mediante un agente autónomo que pueda reconocer bananas, naranjas y limones. Se diseñó e implementó un agente en el lenguaje de programación Python que realiza esta tarea mediante distintos módulos externos como numpy, matplotlib y OpenCV y módulos desarrollados específicamente para esta ocasión. Mediante técnicas de procesamiento de imágenes se extrajeron características de un dataset de bananas, naranjas y limones para ser utilizadas como datos de entrada para el agente. Se usaron los algoritmos k-means y k-nn de clustering y de clasificación, respectivamente, para poder definir de qué clase es cada fruta. En primer lugar se utilizó el entorno de desarrollo Google Collab para familiarizarse con el pipeline necesario para el tratamiento de imágenes y luego se procedió a una implementación local con Anaconda y el IDE Spyder. Se investigaron distintos métodos de extracción de características y se opto por poner a prueba el de los "Hu Moments"[2][6]. Luego de un análisis de histogramas se eligieron las componentes del vector de momentos de Hu más representativas para aplicar los algoritmos k-means y k-nn. Una vez cargada la base de datos y obtenidas las características para cada instancia se procedió a crear un modulo para la aplicación de k-means y k-nn respectivamente y luego se evaluó la performance de cada uno.

2. Agente

La descripción del agente y su entorno se realiza en base a la clasificación de Peter Norving [7], Capítulo 1.

2.1. REAS: Rendimiento, Entorno, Actuadores y Sensores

El principal descriptor para el entorno de la tarea es la tabla REAS [7] , la cual especifica la medida de Rendimiento, el Entorno, los Actuadores y los Sensores con los que cuenta el agente.

- **Rendimiento:** Cantidad de aciertos del conjunto de test como una medida de performance de clasificación.
- **Entorno:** Área en el que está ubicada la fruta a ser clasificada. La iluminación y el fondo son extremadamente importantes por lo que en una aplicación real estos factores pueden limitar enormemente la practicidad de un sistema autónomo. En nuestro caso y a fines de simplificar el estudio el entorno esta acotado a un fondo blanco con buena iluminación uniforme.
- **Actuadores:** Forma de comunicar o actuar del agente según el resultado de la clasificación. En este caso podría ser la colocación de una etiqueta, una señal luminosa o simplemente la asignación de un precio unitario.
- **Sensores:** En nuestro caso es una cámara con la cual se observan las frutas.

2.2. Propiedades del entorno de la tarea

- **Parcialmente observable:** No podemos observar completamente el entorno ya que estamos tomando una fotografía bidimensional en la cual solo vemos una proyección el volumen real. Un cambio de ángulo alcanza para modificar totalmente la información por lo que decimos que el entorno es parcialmente observable.
- **Agente simple:** Hay un único agente involucrado.
- **Estocástico:** El siguiente estado del entorno, es decir, la fruta siguiente a analizar, no esta determinada por la fruta actual.

- **Estático:** El entorno no cambia mientras el agente está deliberando. Una vez tomada la fotografía, comenzado el procesamiento el agente se aísla en cierto modo del entorno.
- **Discreto:** El agente no interactúa en tiempo real con el entorno, sino solo en el corto lapso en que toma la fotografía.

2.3. Diseño del agente

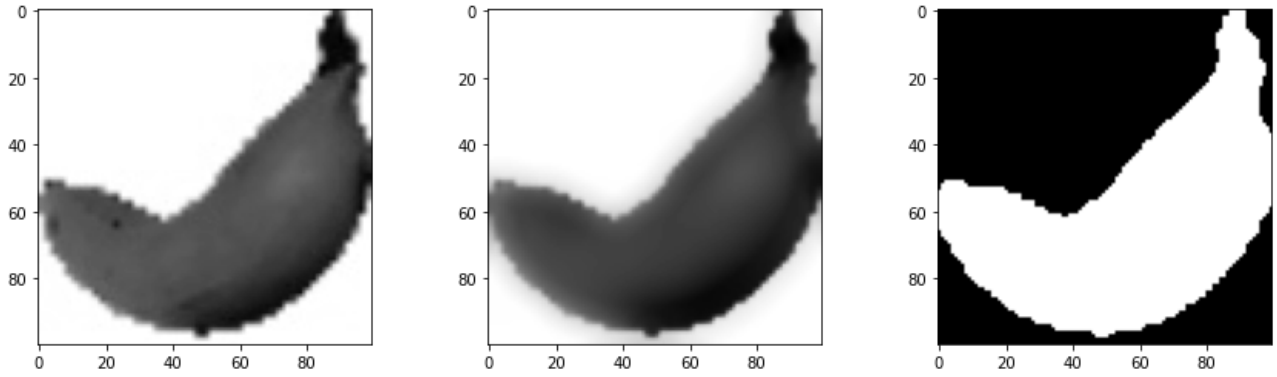
Se utilizaron los algoritmos K-means y K-nn con el objetivo de que el agente aprenda de cada experiencia, y así lograr un cierto nivel de autonomía apoyándose cada vez mas en lo aprendido gracias a sus propias percepciones que en el conocimiento inicial proveído por el diseñador. En este caso trabajamos con un agente racional ya que "En cada posible secuencia de percepciones, un agente racional deberá emprender aquella acción que supuestamente maximice su medida de rendimiento, basándose en las evidencias aportadas por la secuencia de percepciones y en el conocimiento que el agente mantiene almacenado."[7]. Además es no omnisciente, ya que no conoce el resultado de sus acciones, no posee total información de todo su entorno.

3. Tratamiento de imagen

Las imágenes del dataset se cargan mediante OpenCV y se inicializa un objeto "Fruit"para cada imagen. Es en este objeto en donde se realizaran las tareas de filtrado, binarización y extracción de características que nos permitirán luego aplicar los algoritmos de clusterización.

3.1. Filtrado

Las imágenes son cargadas en color, luego se transforman a nivel de grises y se aplica un filtro bilateral para eliminar ruido y mejorar el desempeño del algoritmo. Este filtro es necesario ya que las imágenes de la base de datos están en baja resolución para facilitar el tratamiento y pueden contener ruido. Se eligió este tipo de filtro en lugar de uno gaussiano o de mediana ya que permite conservar mejor los bordes de la imagen facilitando la tarea siguiente.



(a) Escala de grises (b) Filtro bilateral (c) Binarización

Figura 1: Preparación de la imagen

```
#load image
img = cv.cvtColor(cv.imread(self.path),cv.COLOR_BGR2RGB)
```

```
#transform to grayscale
img_g = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
#filter
img_gf = np.zeros((100,100,1),np.uint8)
img_gf = cv.bilateralFilter(img_g,15,80,80)
#binarize
_,img_b = cv.threshold(img_gf, 200, 255, cv.THRESH_BINARY)
img_b = np.invert(img_b)
```

3.2. Extracción de características: *Hu moments*

Para poder aplicar los algoritmos de clasificación necesitamos encontrar un conjunto de características que representen a cada fruta. Existen muchos métodos para obtener valores característicos de una imagen que se adaptan a distintas situaciones y nuestro caso dada la naturaleza de las imágenes una aproximación geométrica es mas conveniente y es por eso que utilizaremos los *Hu moments*.

Un momento de imagen es cierto promedio ponderado particular de las intensidades de los píxeles de una imagen; o también una función de tales momentos. Los *Hu moments* son un grupo de 7 valores propuestos por Hu que son **invariantes a la translación, rotación y escala de la imagen**. Estos se calculan en función de los momentos centroidales del objeto. Los valores y las escalas de los 7 Hu moments son muy distintos entre sí, con diferencias de hasta 3 o 4 órdenes de magnitud entre ellos y escalas en el orden de 10^{12} o 10^{14}. Es por esto que se aplica una transformación logarítmica para que los valores sean comparables entre ellos y que además las escalas de los números sean más apreciables para un humano.

```
#Calculate Moments
moments = cv.moments(img_b)

#Calculate Hu Moments
huMoments = cv.HuMoments(moments)

#Log scale hu moments
hu = -np.sign(huMoments) * np.log10(np.abs(huMoments))
```

3.2.1. Análisis de histogramas

Una vez obtenidos los 7 valores que representan a cada imagen debemos analizar cuales son mejores, es decir los que presentan mas variación entre una fruta y la otra para luego aplicar los algoritmos de clusterizacion. Para esto analizamos todas las imágenes del grupo de Training calculando sus Hu moments y graficando un histograma para cada uno de los 7 valores como podemos ver en la figura 2 :

Aquí podemos ver que la separación es muy buena en las componentes 1 y 3 y que además las componentes 4,5,6 podrían utilizarse en valor absoluto o eligiendo la componente negativa o positiva de cada uno. En nuestro caso utilizamos solo las componentes 1 y 3.

4. Algoritmos

Utilizaremos dos algoritmos distintos para la clusterizacion y posterior clasificación, k-means y k-nn.

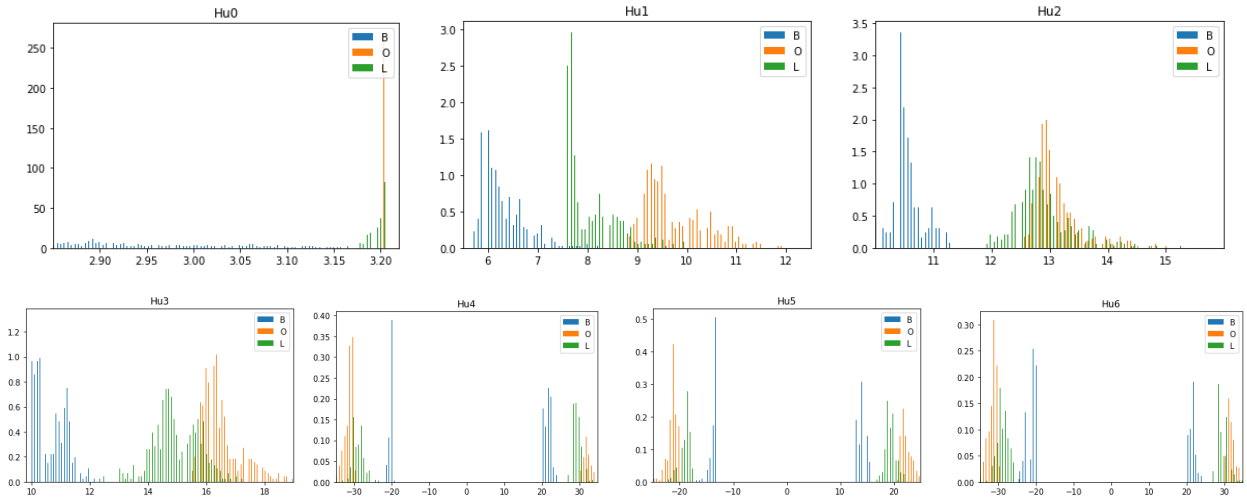


Figura 2: Histogramas de los 7 Hu moments

4.1. k-means

K-means es un algoritmo de clasificación no supervisada (clusterización) que agrupa objetos en k grupos basándose en sus características. El agrupamiento se realiza minimizando la suma de distancias entre cada objeto y el centroide de su grupo o cluster. Se suele usar la distancia cuadrática.

El algoritmo se puede explicar en tres pasos:

1. **Inicialización:** Una vez escogido el número de grupos, k, se establecen k centroides en el espacio de los datos, por ejemplo, escogiéndolos aleatoriamente.

2. **Asignación objetos a los centroides:** Cada objeto de los datos es asignado a su centroide más cercano.

3. **Actualización centroides:** Se actualiza la posición del centroide de cada grupo tomando como nuevo centroide la posición del promedio de los objetos pertenecientes a dicho grupo.

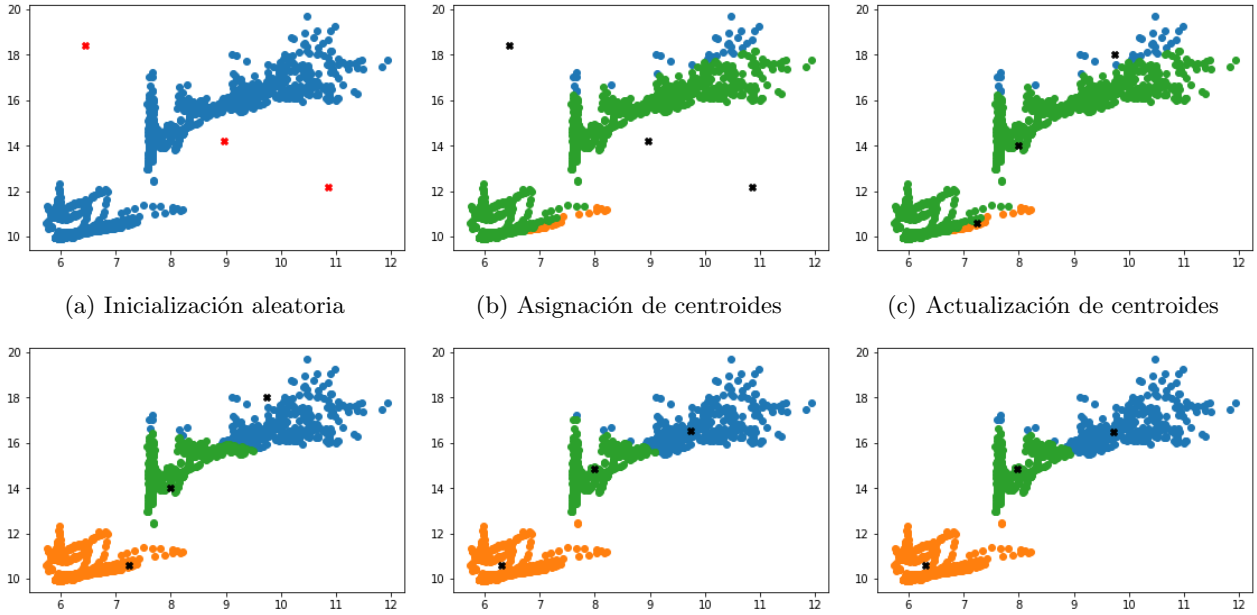
Se repiten los pasos 2 y 3 hasta que los centroides no se mueven, o se mueven por debajo de una distancia umbral en cada paso.

El algoritmo k-means resuelve un problema de optimización, siendo la función a optimizar (minimizar) la suma de las distancias cuadráticas de cada objeto al centroide de su cluster. Las principales ventajas del método k-means son que es un método sencillo y rápido. Pero es necesario decidir el valor de k y el resultado final depende de la inicialización de los centroides. En principio no converge al mínimo global sino a un mínimo local.

4.1.1. Implementación

Se desarrollo una clase *"kmeans"* que trabaja con una lista de objetos de la clase *"Fruit"* que contienen principalmente las características calculadas es decir los Hu moments 1 y 3, la etiqueta real y la etiqueta estimada por el programa. Nuestro objeto kmeans tiene dos funcionalidades, en su inicialización va a calcular los centroides correspondientes a cada cluster y por votación asigna una etiqueta a los mismos. La inicialización

de los centroides puede ser aleatoria o manual. El programa itera los pasos 2 y 3 mencionados en la sección 4.1 hasta que los centroides se mueven menos que una tolerancia determinada arbitrariamente.



(a) Inicialización aleatoria (b) Asignación de centroides (c) Actualización de centroides

(d) Iteración 1 (e) Iteración 3 (f) Iteración 4

Figura 3: Implementación de K-means para la base de datos Fruits 360

Además de encontrar los clusters el programa asigna una etiqueta a cada centroide por el simple método de votación, es decir que al centroide al que se le hayan asignado mas bananas tendrá la etiqueta "Bananas". Esto nos sirve para la segunda funcionalidad que es la clasificación de nuevas imágenes.

Para la función de clasificación lo que hacemos es calcular para cada nueva imagen la distancia a los centroides encontrados anteriormente, y la asignamos al mas cercano dándole así la etiqueta del mismo (Nearest Centroid Classifier).

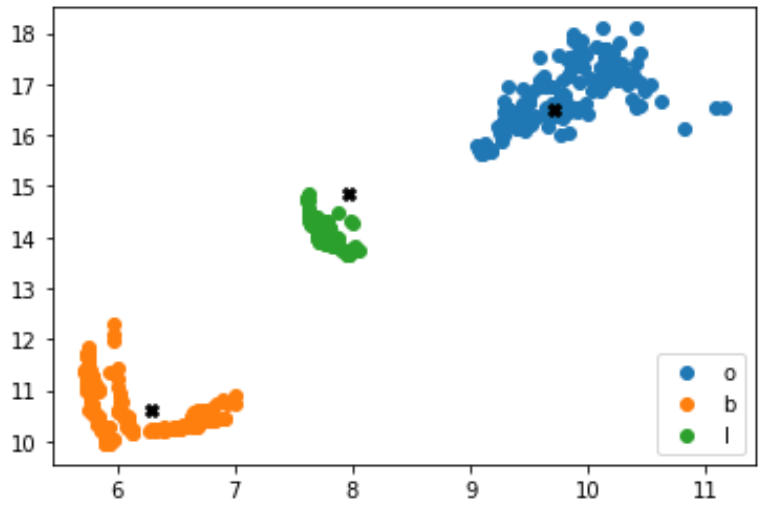


Figura 4: Resultado de clasificación con nuevas imágenes

4.2. k-nn

Es un método que simplemente busca en las observaciones más cercanas a la que se está tratando de predecir y clasifica el punto de interés basado en la mayoría de datos que le rodean. Es supervisado ya que requiere una base de datos previamente etiquetada y basado en instancia ya que no aprende explícitamente un modelo si no que memoriza las instancias de entrenamiento que son usadas como "base de conocimiento" para la fase de predicción.

El algoritmo consta de 3 pasos:

1. **Cargar una base de datos etiquetada:** Se carga cada instancia en su posición para usar como base de conocimiento.
2. **Seleccionar los "k" elementos más cercanos:** Se calcula la distancia entre la nueva instancia a clasificar y los elementos cargados anteriormente para luego elegir los k elementos mas cercanos.
3. **Realizar una "votación" entre los n puntos:** los de una clase/etiqueta que "dominen"(es decir, la moda) decidirán su clasificación final.

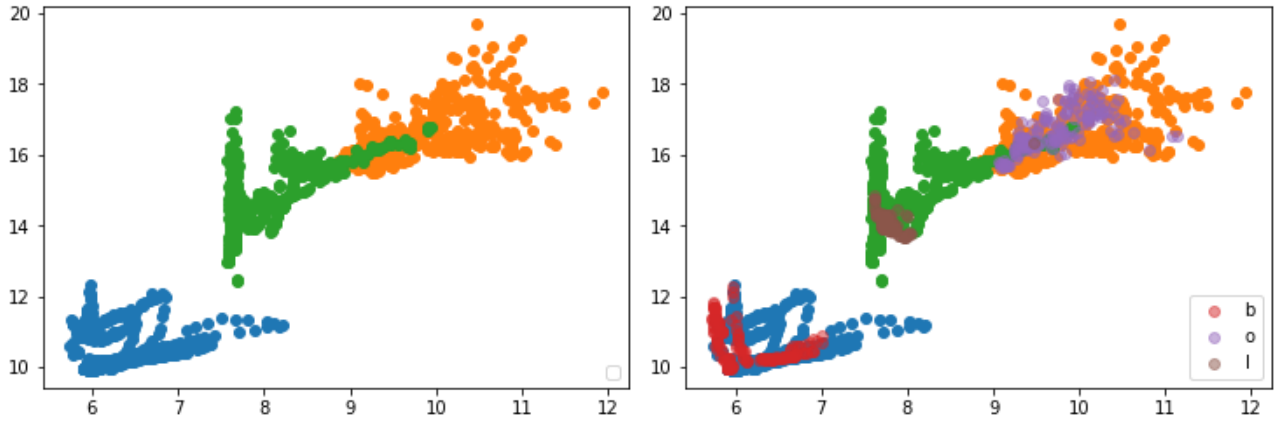
En este algoritmo es muy importante el valor de "k", pues este terminará casi por definir a qué grupo pertenecerán los puntos, sobre todo en las fronteras entre grupos. El numero de vecinos ideal varia según cada aplicación, pero lo que es seguro es que cuantos más "puntos k", más tardará nuestro algoritmo en procesar y darnos respuesta.

Como pros tiene sobre todo que es sencillo de aprender e implementar. Tiene como contras que utiliza todo el dataset para entrenar "cada punto" y por eso requiere de uso de mucha memoria y recursos de procesamiento (CPU). Por estas razones kNN tiende a funcionar mejor en datasets pequeños y sin una cantidad enorme de features.

4.2.1. Implementación

Se desarrollo una clase "knn"que trabaja con 3 listas de objetos "Fruit", una para cada fruta que contienen principalmente las características calculadas es decir los Hu moments 1 y 3, la etiqueta real y la etiqueta estimada

por el programa. En la inicialización del objeto se cargan las listas y se especifica el valor de "k"deseado. Se experimento con distintos valores de "k"pero no se observo una diferencia en la performance del algoritmo por lo que se fijó en k = 3. Una vez que tenemos la base de datos etiquetada podemos proceder a la clasificación siguiendo los pasos 2 y 3 mencionados anteriormente a través de la función çlassify".



(a) Base de conocimiento

(b) Clasificación K = 2

Figura 5: Implementación de K-nn para la base de datos Fruits 360

5. Resultados

5.1. k-means

Para evaluar la performance del algoritmo k-means aprovechamos que las imágenes ya están etiquetadas y analizamos los errores cometidos tanto en la etapa de aprendizaje como en la de clasificación.

En la etapa de clusterizacion encontramos un error del 2,8 % que representa a 41 frutas que fueron mal agrupadas que están representadas en la siguiente tabla:

Total = 1461		Estimated		
		Bananas	Oranges	Lemons
Real	Bananas	490	0	0
	Oranges	0	479	0
	Lemons	1	40	452

Figura 6: Errores en la etapa de clustering

Cabe destacar que se experimentó con diferentes tolerancias para la actualización de los centroides y a partir de 0,1 no se observó una disminución del error.

En cuanto a la etapa de clasificación con la base de datos "test", el rendimiento fue del 100 % probablemente gracias a la gran similitud que hay entre las imágenes de prueba y las de entrenamiento.

5.2. knn

En este caso solo se evalúa la etapa de clasificación ya que es un algoritmo supervisado. Se evaluó la performance con distintos valores de k y se encontró que a partir de k = 4 la performance es del 100 % seguramente debido a la similitud que hay entre las imágenes de prueba y las de entrenamiento.

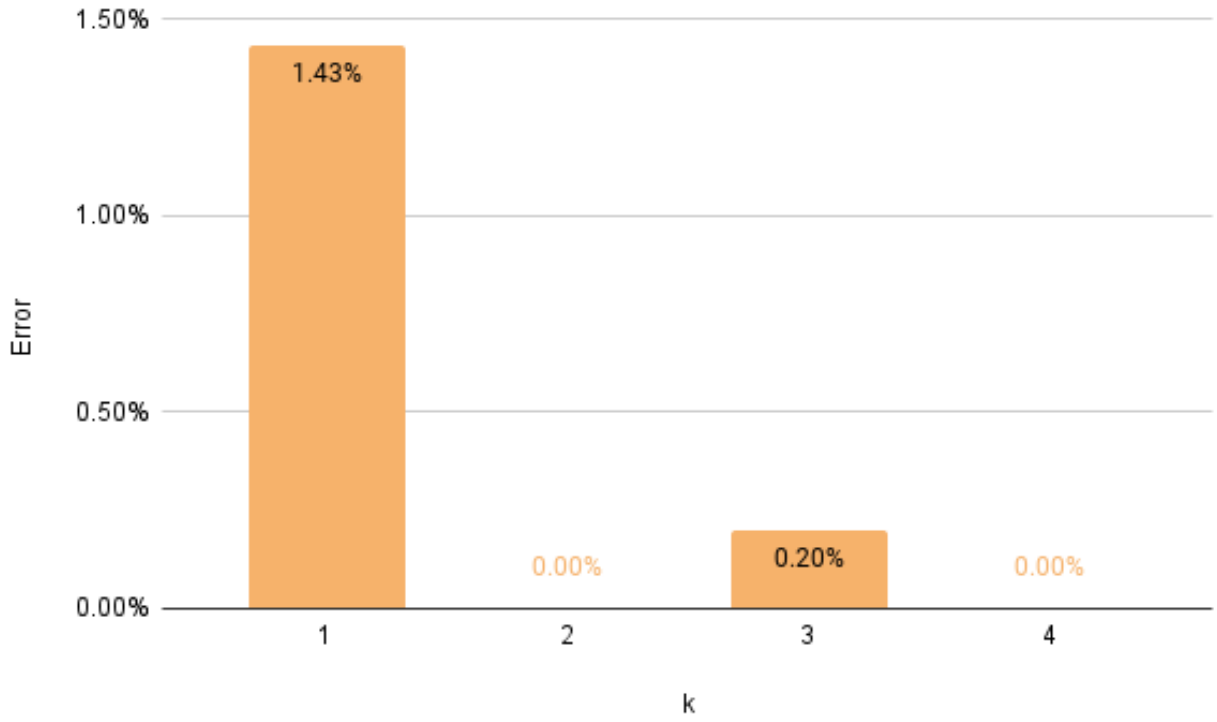


Figura 7: Errores en la etapa de clasificación

6. Conclusión

Con el objetivo de agilizar el proceso de cobro en las cajas de un supermercado se diseñó un agente inteligente capaz de identificar y clasificar frutas entre bananas, naranjas y limones.

Se comenzó por una especificación del agente indicando la tabla REAS y las propiedades del entorno de la tarea. El agente trabaja con un dataset de uso libre llamado Fruits 360. Posteriormente se detallo el procesamiento de las imágenes que consistió en un acondicionamiento mediante un filtro bilateral para disminuir el ruido y una binarización simple mediante thresholding y una extracción de características geométricas mediante el calculo de los "Hu moments". Mediante un análisis de histogramas se encontraron las componentes mas representativas de los Hu moments y fueron utilizadas para representar a cada imagen en los algoritmos estudiados. tanto K-means como Knn dieron muy buenos resultados con el dataset utilizado por lo que elegir entre uno y otro dependerá de factores externos como tamaño de la base de datos o potencia de calculo disponible al momento de la implementación.

6.1. Posibles mejoras

Si bien los algoritmos mostraron ser muy eficaces, el uso de una base de datos con imágenes pre-procesadas facilita mucho la tarea y dada la similitud de características entre frutas como las naranjas y los limones, en algunos casos estas frutas podrían ser confundidas fácilmente. Para solucionar este problema se propone trabajar con imágenes a color para agregar una dimensión mas a los algoritmos k-means y k-nn a través de un análisis de histogramas RGB.

Referencias

[1] Python Software Foundation. *Python Docs*. 2022.

[2] Satya Mallick y Krutika Bapat. *Shape Matching using Hu Moments*. 2018.

[3] Na8. *Clasificar con K-Nearest-Neighbor ejemplo en Python*. 2018. URL: `https://www.aprendemachinelearning.com/clasificar-con-k-nearest-neighbor-ejemplo-en-python/` (visitado 04-04-2022).

[4] *OpenCV Docs*. 2022.

[5] Universidad de Oviedo. *El algoritmo k-means aplicado a clasificación y procesamiento de imágenes*. 2022. URL: `https://www.unioviedo.es/compnum/laboratorios_py/kmeans/kmeans.html` (visitado 04-04-2022).

[6] Adrian Rosebrock. *OpenCV Shape Descriptor: Hu Moments Example*. 2014.

[7] Stuart Rusell y Peter Norving. *Inteligencia Artificial: Un Enfoque Moderno*. Pearson Educación S. A, 2004.