SOLUCION DE SISTEMAS DE ECUACIONES LINEALES

Métodos Directos

Cancel

Start

INTEGRANTES:
Barrera, Paula
Dávila, Juan
Vildoza, Berenice

PROGRAMA

TEMAS ABORDADOS

- INTRODUCCION
- METODOS DIRECTOS
- APLICACIONES E IMPLEMENTACIONES
- CONCLUSION

¿QUE SON LOS SISTEMAS DE ECUACIONES LINEALES?

$$\begin{gathered} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{gathered}$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

¿CUÁL ES SU IMPORTANCIA?

Los sistemas de ecuaciones lineales nos sirven para resolver diversos problemas, desde los que se presentan en nuestra vida diaria hasta problemas que se presentan en ingeniería, física, matemáticas, economía y otras ciencias. El interés en encontrar la solución a estos sistemas es muy antiguo.

PROGRAMA

TEMAS ABORDADOS

- INTRODUCCION
- METODOS DIRECTOS
- APLICACIONES E IMPLEMENTACIONES
- CONCLUSION

ELIMINACIÓN DE GAUSS

Se busca llegar a un sistema equivalente que tenga una matriz triangular superior

ELIMINACIÓN DE GAUSS

Sea el Sistema: Ax=b

Pasamos el sistema a su forma matricial:

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \begin{Bmatrix} x_1 \\ \vdots \\ x_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ \vdots \\ b_n \end{Bmatrix}$$

Armamos la matriz ampliada:

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} & b_1 \\ \vdots & \ddots & \vdots & \vdots \\ a_{n1} & \cdots & a_{nn} & b_n \end{bmatrix}$$

Eliminamos los elementos bajo la diagonal mediante operaciones elementales.

Se resuelve usando Sustitución hacia atrás

DESCOMPOSICON LU

Se busca descomponer la matriz de los coeficientes en el producto de dos matrices.

DESCOMPOSICION LU

¿Cómo resolvemos un sistema de ecuaciones lineales con descomposicion LU?

- **Paso 1:**

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & L_{12} & L_{13} & \dots & L_{1n} \\ 0 & 1 & L_{23} & \dots & L_{2n} \\ 0 & 0 & 1 & \dots & L_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & L_{nn} \end{bmatrix}$$

$$U = \begin{bmatrix} U_{11} & 0 & 0 & \dots & 0_{1n} \\ U_{21} & U_{22} & 0 & \dots & 0_{2n} \\ U_{31} & U_{32} & U_{33} & \dots & 0_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ U_{n1} & U_{n2} & U_{n3} & \dots & U_{nn} \end{bmatrix}$$

A=LU

DESCOMPOSICION LU

- **Paso 2:**

Paso k: suponiendo $a^{k}_{kk} \neq 0$

$$m_{ik} = a^{k}_{ik} / a^{k}_{kk} \qquad i = k+1, \ldots, n$$

$$a_{ij}^{k+1} = a^{k}_{ij} - m_{ik}\, a^{k}_{kj} \qquad i,j = k+1, \ldots, n$$

$$A^{k+1} = (a_{ij}^{k+1})$$

Despues de n-1 pasos:

$$U = \begin{bmatrix} a^1_{11} & a^1_{12} & a^1_{13} & . & a^1_{1n} \\ 0 & a^2_{22} & a^2_{23} & . & a^2_{2n} \\ 0 & 0 & a^3_{33} & . & a^3_{3n} \\ . & . & . & . & . \\ 0 & 0 & 0. & . & a^n_{nn} \end{bmatrix} \qquad L = \begin{bmatrix} 1 & 0 & 0 & . & 0 \\ m_{21} & 1 & 0 & . & 0 \\ m_{31} & m_{32} & 1 & . & . \\ . & . & . & . & . \\ m_{n1} & m_{n2} & . & . & 1 \end{bmatrix}$$

DESCOMPOSICION LU

- **Paso 3:**

$$Ax=b$$

$$LUx=b$$

$$\begin{cases} Ly=b \\ Ux=y \end{cases}$$

$Ly=b$ → **Sustitucion hacia adelante**

$Ux=y$ → **Sustitucion hacia atrás**

COSTO ALGORITMICO: O(N^3)

DESCOMPOSICION LU

¿Cómo podemos usar la descomposición de LU para encontrar la **inversa de la matriz**?

$$[A]\ [A]^{-1} = [I] \Rightarrow [A]\,[B] = [I]$$

$$[A]\begin{bmatrix} b_{11} \\ b_{21} \\ \vdots \\ b_{n1} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad [A]\begin{bmatrix} b_{12} \\ b_{22} \\ \vdots \\ b_{n2} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

Todas las columnas de [B] se pueden encontrar resolviendo n diferentes conjuntos de ecuaciones siendo la columna del lado derecho las n columnas de la matriz de identidad.

DESCOMPOSICION LU

El tiempo computacional para la eliminación gaussiana y la descomposición de LU es idéntico :

$$T\left(\frac{8n^3}{3}+12n^2+\frac{4n}{3}\right)$$

¿POR QUÉ DEBERÍA APRENDER EL MÉTODO DE DESCOMPOSICIÓN LU CUANDO TOMA EL MISMO TIEMPO COMPUTACIONAL QUE LA ELIMINACIÓN GAUSSIANA?

El tiempo computacional total requerido para encontrar la inversa de una matriz usando descomposición de LU:

$$T\left(\frac{32n^3}{3}+12n^2-\frac{20n}{3}\right)$$

El tiempo computacional total requerido para encontrar la inversa de una matriz mediante el uso de la eliminación gaussiana:

$$T\left(\frac{8n^4}{3}+12n^3+\frac{4n^2}{3}\right)$$

PIVOTEO PARCIAL

Técnica para reducir los errores
al momento de resolver
Sistemas de Ecuaciones Lineales

PIVOTEO PARCIAL

Tomamos como pivote el elemento de mayor valor absoluto de una columna:

$$\boxed{c_k = \max_{i=k\ldots n} \left| a_{ik}^{k} \right|}$$

De esta forma los multiplicadores:

$$|m_{ik}| \leq 1 \qquad , i = k+1, \ldots, n$$

Si i>k se intercambian las filas en A y b.

Luego, se trabaja de igual manera que en la descomposicion LU

PIVOTEO PARCIAL

Si representamos con P=(Pn-1...P1) las permutaciones realizadas, entonces:

PA=LU

Dado:

Ax=b

si PA=LU ∴ A=PLU

El sistema a resolver quedaría:

PLUx=b

Ly=Pb

Ux=y

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22}^{1} & a_{23}^{1} & a_{24}^{1} \\ 0 & a_{32}^{1} & a_{33}^{1} & a_{34}^{1} \\ 0 & a_{42}^{1} & a_{43}^{1} & a_{44}^{1} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

Pivóte Máximo

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{42}^{1} & a_{43}^{1} & a_{44}^{1} \\ 0 & a_{32}^{1} & a_{33}^{1} & a_{34}^{1} \\ 0 & a_{22}^{1} & a_{23}^{1} & a_{24}^{1} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_4 \\ b_3 \\ b_2 \end{pmatrix}$$

PIVOTEO PARCIAL ESCALADO

Se utiliza en casos donde algunas de las ecuaciones de un sistema tienen coeficientes mucho más grandes que otros

PIVOTEO PARCIAL ESCALADO

Se toma como elemento pivote, al de mayor valor absoluto de una columna dividido en el elemento en su respectivo elemento perteneciente al vector tamaño:

$$c_k = \max_{i=k\ldots n} \frac{\left|a_{ik}^{k}\right|}{s_i}$$

s (vector tamaño): se inicializa al comenzar el algoritmo:

$$s_i = \max_{j=1\ldots n} \left|a_{ij}\right|$$

Luego, se procede igual que en pivoteo parcial: Ax=b

El sistema a resolver quedaría: Ly=Pb, Ux=y

OTROS MÉTODOS

OTROS MÉTODOS

GAUSS JORDAN

SIMILAR A GAUSS, LA DIFERENCIA ES QUE SE DIAGONALIZA LA MATRIZ. DESVENTAJA: EL COSTO AUMENTA EN UN 50%

Se utiliza en ingeniería mecánica y civil para resolver sistemas de ecuaciones de fuerza.

CHOLESKY

USA UNA MATRIZ TRIANGULAR INFERIOR CON DIAGONAL POSITIVA Y SU TRANSPUESTA. VENTAJA: EL COSTO ES LA MITAD DE LU Y NO NECESITA PIVOTEO

Se usa comúnmente en el método de Montecarlo

THOMAS

VARIANTE DE LU, USA MATRICES TRIDIAGONALES . VENTAJA: EL COSTO ES O(N)

Se usa en el metodo Cubic Spline

QUIZZ

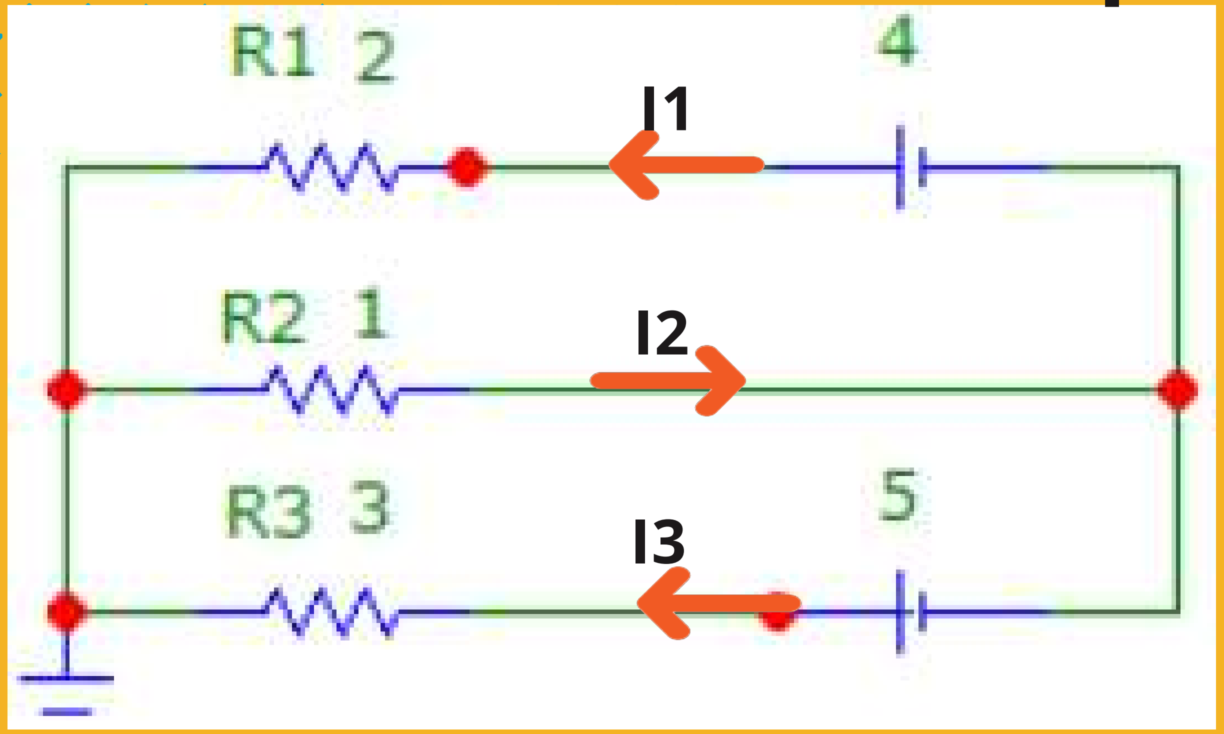
PROGRAMA

TEMAS ABORDADOS

- INTRODUCCION
- METODOS DIRECTOS
- APLICACIONES E IMPLEMENTACIONES
- CONCLUSION

CIRCUITOS ELÉCTRICOS

- Determinar las corrientes electricas para el siguiente circuito:



Utilizando las leyes de Kirchhoff de tensión y corriente se llega al sistema:

$$\begin{cases} I1-I2+I3=0 \\ 2*I1+I2=4 \\ I2+3*I3=5 \end{cases}$$

$$\begin{bmatrix} 1 & -1 & 1 \\ 2 & 1 & 0 \\ 0 & 1 & 3 \end{bmatrix} \begin{bmatrix} I1 \\ I2 \\ I3 \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \\ 5 \end{bmatrix}$$

BALANCE DE ECUACIONES (QUIMICA)

$$\underbrace{Zn}_{a} + \underbrace{HNO_3}_{b} \rightarrow \underbrace{Zn(NO_3)_2}_{c} + \underbrace{NH_4NO_3}_{d} + \underbrace{H_2O}_{e}$$

Zn: $a=c$

N: $b=2c + 2d$

H: $b=4d+2e$

O: $3b=6c+ 3d +e$

$\rightarrow$ $a=c$

$b=1$

$$\begin{cases} 2c + 2d = 1 \\ 4d + 2e = 1 \\ 6c + 3d + e = 3 \end{cases}$$

$$\begin{bmatrix} 2 & 2 & 0 \\ 0 & 4 & 2 \\ 6 & 3 & 1 \end{bmatrix} \begin{bmatrix} c \\ d \\ e \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix}$$

MÉTODO DE MONTE CARLO

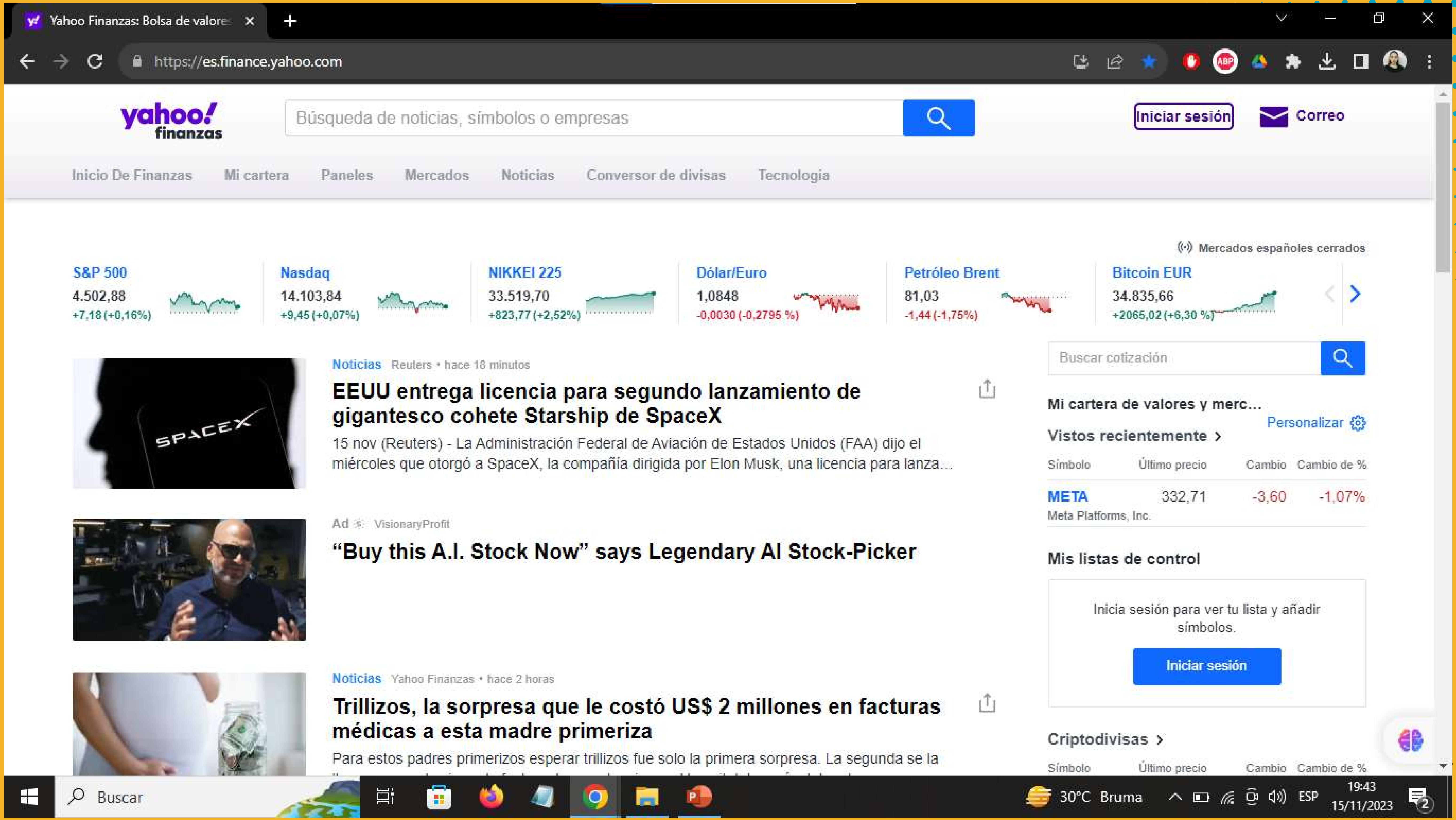
En este caso para una cartera con unos activos y pesos determinados, siguiendo la factorización de Cholesky para darle estabilidad numérica y simular sistemas con variables múltiples correlacionadas. El proceso consiste en descomponer esta matriz de correlación entre los activos para obtener la triangular inferior "L", para a continuación multiplicar esta por un vector de ruidos simulados "u" descorrelacionados. Con esto obtendremos un vector "Lu" que mantiene las propiedades de covarianza del sistema a ser modelado.

$$\textit{Por el metodo de cholesky:} \quad [A] = [\,L] * [U]$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} \end{bmatrix}$$

$$= \begin{bmatrix} l_{1,1} & 0 & 0 & 0 & 0 \\ l_{2,1} & l_{2,2} & 0 & 0 & 0 \\ l_{3,1} & l_{3,2} & l_{3,3} & 0 & 0 \\ l_{4,1} & l_{4,2} & l_{4,3} & l_{4,4} & 0 \\ l_{5,1} & l_{5,2} & l_{5,3} & l_{5,4} & l_{5,5} \end{bmatrix} * \begin{bmatrix} 1 & U_{2,1} & U_{3,1} & U_{4,1} & U_{5,1} \\ 0 & 1 & U_{3,2} & U_{4,2} & U_{5,2} \\ 0 & 0 & 1 & U_{4,3} & U_{5,3} \\ 0 & 0 & 0 & 1 & U_{5,4} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

MÉTODO DE MONTE CARLO



MÉTODO DE MONTE CARLO

Fragmento del código de la implementación en Python

```
L = np.linalg.cholesky(covar)
u = norm.ppf(np.random.rand(num_stocks, num_stocks))
Lu = L.dot(u)
```

CIFRADO DE HILL

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

La palabra que vamos a codificar es: MÉTODOS NUMÉRICOS

M	E	T	O	D	O	S	N	U	M	E	R	I	C	O	S
12	4	19	14	3	14	18	13	20	12	4	17	8	2	14	18

$$\begin{pmatrix}12\\4\end{pmatrix}\begin{pmatrix}19\\14\end{pmatrix}\begin{pmatrix}3\\14\end{pmatrix}\begin{pmatrix}18\\13\end{pmatrix}\begin{pmatrix}20\\12\end{pmatrix}\begin{pmatrix}4\\17\end{pmatrix}\begin{pmatrix}8\\2\end{pmatrix}\begin{pmatrix}14\\18\end{pmatrix}$$

CIFRADO DE HILL

CODIFICACIÓN

Matriz de codificación: $\begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix}$

Multiplicamos la matriz de codificación por cada uno de los vectores obtenidos anteriormente

$$\begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 12 \\ 4 \end{bmatrix} = \begin{bmatrix} 20 \\ 4 \end{bmatrix} \xrightarrow{\text{mod } 26} \begin{bmatrix} 20 \\ 4 \end{bmatrix}$$

CIFRADO DE HILL

DECODIFICACIÓN

Para decodificar la palabra, serealiza el mismo proceso,peroestavezmultiplicando los vectores de la palabra codificada por la inversa de la matriz de codificación

$$\begin{bmatrix} 3 & -2 \\ -1 & 1 \end{bmatrix}$$

Multiplicamos la inversa con cada uno de los vectores formados:

$$\begin{bmatrix} 3 & -2 \\ -1 & 1 \end{bmatrix} * \begin{bmatrix} 20 \\ 24 \end{bmatrix} = \begin{bmatrix} 12 \\ 4 \end{bmatrix} \xrightarrow[\text{mod } 26]{} \begin{bmatrix} 12 \\ 4 \end{bmatrix}$$

Nos devuelve el valor del vector antes de codificar

PROCESAMIENTO DE IMÁGENES

La provincia, Mendoza, representa más del 60% de la producción de vino del país.

Hay una gran variedad de varietales, y para alguien que no es de la región es difícil reconocer cada uno a simple vista.

Se pueden clasificar a partir de distintas características: color, pH, acidez, momentos de Hu, etc

https://github.com/cabustillo13/Charla-procesamiento-de-imagenes

PROCESAMIENTO DE IMÁGENES

CASO PRÁCTICO:

La provincia, Mendoza, representa más del 60% de la producción de vino del país.

Hay una gran variedad de varietales, y para alguien que no es de la región es difícil reconocer cada uno a simple vista.

Se pueden clasificar a partir de distintas características: color, pH, acidez, momentos de Hu, etc

- Che, ¿qué es la visión por computadora?
- ¿ Qué aspectos y etapas debo considerar al momento de definir mi agente?
- ¿Por dónde comienzo con el Procesamiento de Imágenes?

NOS ADENTRAMOS AL PROBLEMA...

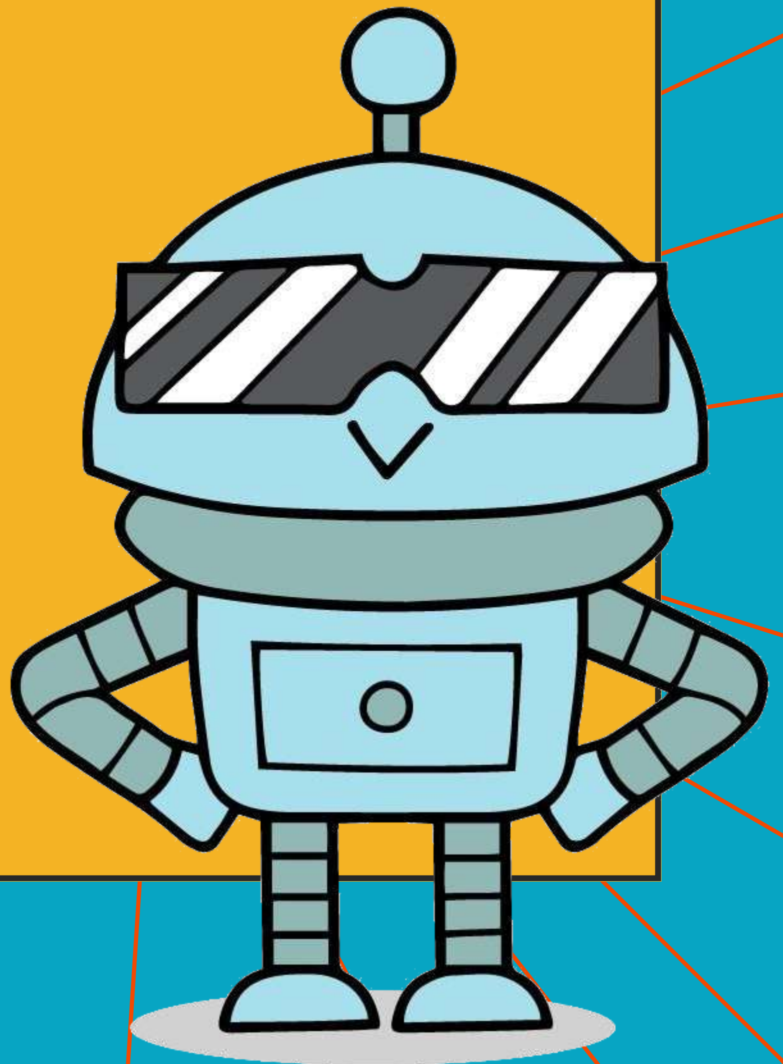
Se propone un modelo para analizar uvas blancas, rosadas y azules. En el cual le presentamos una imagen al Agente.py y las clasifique.

[Input]

[Agente.py]

Es una Grape Blue / Uva azul

[Output]



ETAPAS

01

Adquisición de imágenes

Vamos a usar un dataset de Kaggle

02

Transformación

Modificación de dimensiones de la imagen

03

Preprocesamiento

Conversión a escala de grises

04

Filtración

Suavizar bordes, eliminar ruido, realzar la imagen y detectar bordes

05

Extracción de rasgos

Vamos a usar Momentos de HU

06

Clasificación

Vamos a usar el algoritmo de aprendizaje supervisado KNN

01 ADQUISICIÓN DE IMÁGENES

Vamos a utilizar un dataset de Kaggle: Fruits 360.
Y solo utilizaremos las clases grapeBlue, grapeWhite y grapePink.

Link: https://www.kaggle.com/moltean/fruits

02

TRANSFORMACIÓN

Inicialmente cada imagen es de 100x100 píxeles.

Acá se puede recortar el elemento de la imagen, variación en la escala, etc.

03

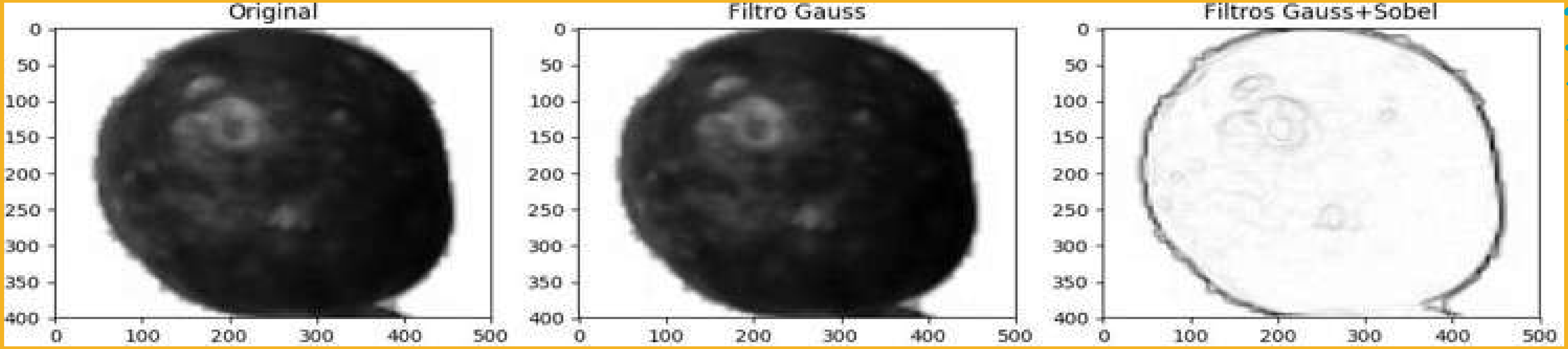
PREPROCESAMIENTO

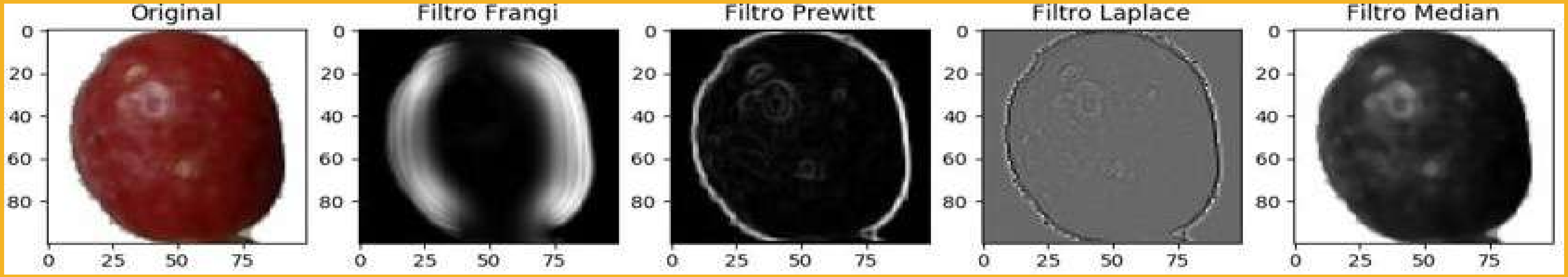
Conversión a escala de grises cada imagen.

04

FILTRACIÓN

Aplicar un filtro para suavizar la imagen, eliminar ruido, realzar la imagen y detectar bordes.





05

EXTRACCIÓN DE RASGOS

A veces previo a esta etapa, se hace una segmentación.

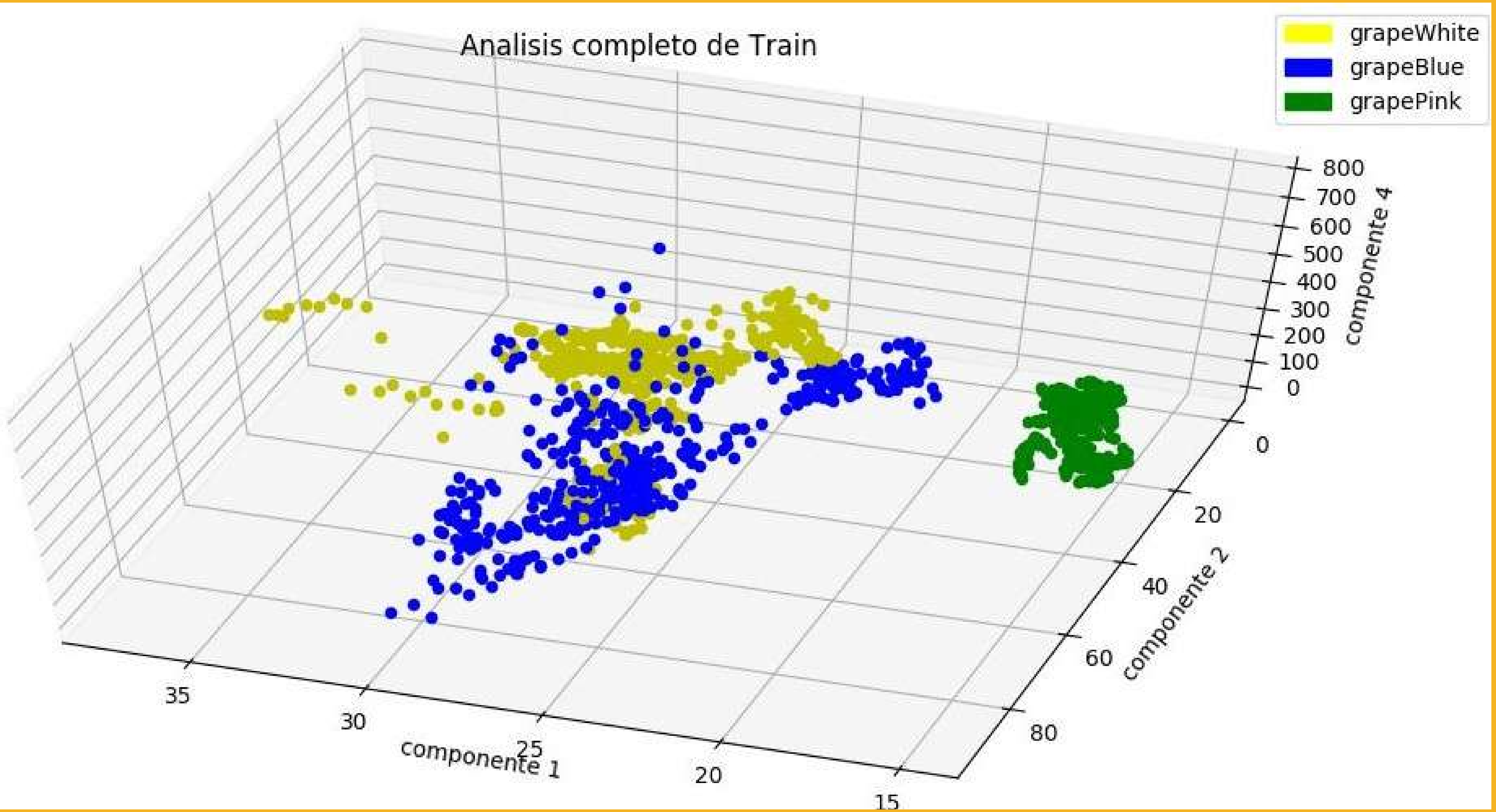
Definir los momentos de Hu 1,2 y 4.

Blue Grape : [7.2662083187, 0.0240486485524, 0.68530837863]
Pink Grape : [4.73860099690, 0.139367923548, 0.64289598354]
White Grape: [8.9039352540, 0.0051721799563, 0.04644156847]

05

EXTRACCIÓN DE RASGOS

Gráfico de Momentos de Hu para 1470 imágenes ¿eso es bueno?

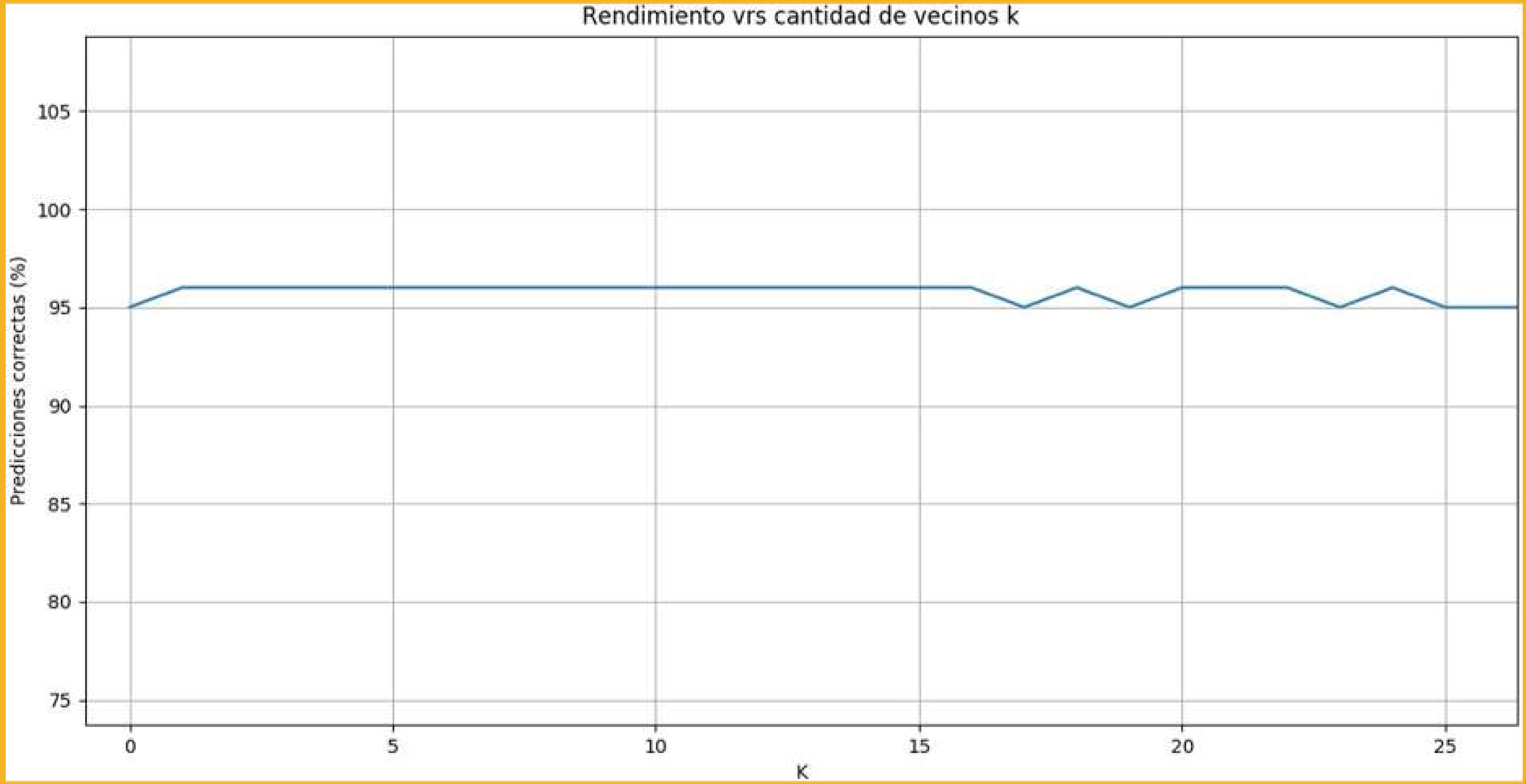


06

CLASİFICACİÓN

Utilizamos el algoritmo KNN (vecinos más cercanos) debido que se obtuvo un buen rendimiento.

Para KNN con k=2 -> Accuracy: **96%**
Para K Means -> Accuracy: **78%**



06

CLASIFICACIÓN

Utilizamos el algoritmo KNN (vecinos más cercanos) debido que se obtuvo un buen rendimiento.

Para KNN con k=2 -> Accuracy: **96%**
Para K Means -> Accuracy: **78%**

Se hace la analogía de este algoritmo con esta frase:

Implementación en Python

TERMINAL PUERTOS CONSOLA DE DEPURACION PROBLEMAS SALIDA

```
PS C:\Users\plbar\Downloads\Procesamiento-de-imagenes> python -u "c:\Users\plbar\Downloads\Procesamiento-de-imagenes\main.py"
grapeWhite OK
grapeBlues OK
grapePink OK
Analisis completo de la base de datos de Train
Cantidad de imagenes analizadas:
1470
Introduce numero de la foto: 0

Inicializacion KNN

Predicciones para KNN con K=2:
grapeBlue
grapeBlue
PS C:\Users\plbar\Downloads\Procesamiento-de-imagenes>
```

photo0.jpg

Implementación en Python

TERMINAL PUERTOS CONSOLA DE DEPURACIÓN PROBLEMAS SALIDA Code

```
PS C:\Users\plbar\Downloads\Procesamiento-de-imagenes> python -u "c:\Users\plbar\Downloads\Proce
samiento-de-imagenes\main.py"
grapeWhite OK
grapeBlues OK
grapePink OK
Analisis completo de la base de datos de Train
Cantidad de imagenes analizadas:
Introduce numero de la foto: 13

Inicializacion KNN

Predicciones para KNN con K=2:
grapeWhite
grapeWhite
```

photo13.jpg

PROGRAMA

TEMAS ABORDADOS

- INTRODUCCION
- METODOS DIRECTOS
- APLICACIONES E IMPLEMENTACIONES
- CONCLUSION

CONCLUSION

"DOMINAR LA RESOLUCIÓN DE SISTEMAS DE ECUACIONES LINEALES TE BRINDA LA CAPACIDAD DE ENCONTRAR SOLUCIONES EN UN MUNDO CADA VEZ MÁS INTERCONECTADO."

Para acceder a toda la información de la presentación

Para acceder a toda la información de la presentación

¡MUCHAS GRACIAS!