DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 2

Reconocimiento de dígitos

17 de octubre de 2021 — Metodos Numericos

Grupo 05

Integrante	LU	Correo electrónico
Casco, Rocío Diana	512/20	rociodcasco@gmail.com
Dallegri, Pablo	445/20	dallegri.p@gmail.com
Totaro, Facundo Ariel	43/20	facutotaro@gmail.com
Vitali, Lucas Marcelo	278/20	lucasvitali001@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
https://exactas.uba.ar

Índice

1. Introducción **2**
1.1. kNN [1] . . . 2
1.2. Método de la potencia [2] . . . 2
1.3. PCA . . . 2

2. Valores de k y α óptimos **3**
2.1. kNN (sin PCA) . . . 3
2.2. kNN con PCA . . . 4
2.3. Conclusión . . . 5

3. Variación de cantidad de imágenes en muestra **6**
3.1. Experimento 1: Tomando porcentajes . . . 6
3.2. Experimento 2: aumentamos de a 500 imágenes . . . 8
3.3. Conclusión . . . 9

4. Variación k y cantidad de imagenes **10**
4.1. Experimento . . . 10
4.2. Métricas: accuracy, precision y recall . . . 10
4.2.1. Fijando la cantidad de vecinos y variando el porcentaje de imágenes . . . 11
4.2.2. Fijando el porcentaje de imágenes y variando la cantidad de vecinos . . . 11
4.3. Tiempos . . . 12
4.3.1. Fijando el porcentaje de imágenes y variando la cantidad de vecinos . . . 12
4.3.2. Fijando la cantidad de vecinos y variando el porcentaje de imágenes . . . 12

5. Validación (K-Fold) **14**

6. Conclusión **15**

Keywords

Optical digit recognition (OCR), kNN, PCA, K-Fold

1. Introducción

En este trabajo buscaremos implementar un programa el cual dada una imagen de 28x28 pixeles que tenga un dígito entre el 0 y el 9, reconozca qué dígito se encuenta en la imagen.

Para ello, se utilizarán los métodos de kNN y PCA. Se experimentará con ciertos parametros (cantidad de vecinos a considerar en kNN, cantidad de componentes principales a considerar en PCA, etc.) para observar como varían los resultados, ya sea precisión, tiempo de computo, etc.

Para experimentar se utilizará un dataset con dígitos manuscritos en imágenes de 28x28 píxeles en escala de grises, donde los pixeles tomarán valores entre 0 y 255 según su intensidad. Las imágenes tienen forma de una matriz perteneciente a $\mathbb{R}^{28\times 28}$ pero para trabajar con estas imágenes se las transformará en un vector perteneciente a $\mathbb{R}^{784}$. Luego, mediante el método de PCA se reducirá la cantidad de dimensiones a una cantidad α. Seguido de haber realizado esta reducción, se calculará la distancia euclídea del dígito que se quiere conocer al resto de los dígitos del dataset para así, con el método de kNN, determinar a cuál dígito corresponde la imágen.

1.1. kNN [1]

Como se mencionó anteriormente, la idea es considerar a las imágenes como un conjunto de puntos en espacio, y luego tomar las distancias entre ellos. Este método, toma como criterio para determinar el resultado, a los k vecinos más cercanos en distancia euclídea a la imagen de interés. Entonces, tomamos la moda de las etiquetas correspondientes a estos números para determinar a qué dígito corresponde la imagen.

Para hacer esto primero se calcula la distancia de todos las imágenes vectorizadas a la imágen del dígito que nos interesa. Luego se busca a los k dígitos más cercanos a este para después buscar qué dígito aparece más entre estos k más cercanos.

1.2. Método de la potencia [2]

Esta técnica es una etapa central para el Análisis de Componentes Principales. Permite, dada una matriz simétrica, aproximar su autovalor de mayor módulo, junto con su autovector asociado.

Su funcionamiento se basa en aplicar repetidas veces (sobre algún vector) la transformación lineal asociada a una matriz simétrica. Esto produce que cada componente del vector inicial en la dirección de cada autovector de dicha matriz (ortogonales entre sí), modifique su longitud según el autovalor correspondiente, y al iterar, el mayor en módulo vuelve despreciables al resto. Esto permite obtener una aproximación del principal autovector asociado.

1.3. PCA

El algoritmo PCA (prinicipal component analisys) se utiliza para eliminar la correlacion entre las muestras de diferentes variables aleatorias. El objetivo de este proceso es modificar los datos para eliminar la redundancia entre ellos. Para ello se realiza una transformación lineal construida en base a las componentes principales de la matriz de covarianza creada a partir de los datos de entrada. Este proceso tambien se puede aprovechar para reducir la dimension de los datos, usando una cantidad de componentes principales α menor a la cantidad total de componentes principales.

En primer lugar, dada una matriz con un dato de entrada en cada fila, se calcula su matriz de covarianza. Luego, de manera iterativa, se calcula por metodo de la potencia el mayor autovalor y su autovector asociado de la matriz de covarianza. Luego dicha matriz se deflaciona para anularle el mayor autovalor y se corre otra iteracion con la matriz resultante. De esta manera, luego de α iteraciones se obtienen α componentes principales. Finalmente se arma la matriz de cambio de base que elimina la covarianza entre los datos a partir de los autovectores obtenidos.

2. Valores de k y α óptimos

En este bloque de experimentación se van a analizar la calidad de los resultados obtenidos de la utilización de los métodos "kNN (sin PCA)" y "kNN con PCA", variando los paramentos k, la cantidad de vecinos a tomar en cuenta en kNN, y α, la cantidad de componentes principales a tomar en cuenta en PCA.

El objetivo final es obtener los valores k y α que maximice la calidad de nuestro modelo, haciéndolo a su vez preciso y robusto, y en un tiempo de entrenamiento aceptable.

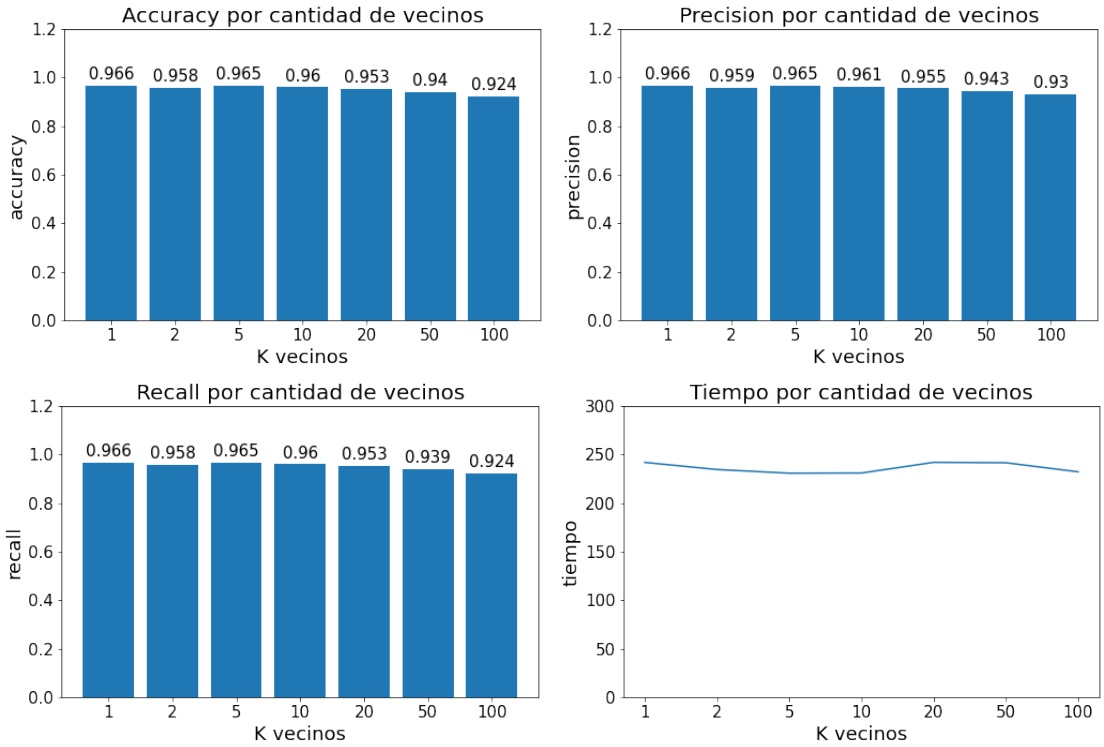
2.1. kNN (sin PCA)

Para este experimento se va a entrenar el modelo con un k igual a 1, 2, 5, 10, 20, 50, 100. Por simplicidad de la experimentación, de los 42000 datos dados, siempre se van a utilizar un 80 % para entrenamiento y un 20 % para validación. Para evaluar los resultados utilizaremos las métricas de accuracy*, precision y recall*, y mediremos los tiempos de ejecución.

Hipótesis:

Creemos que a medida que el parámetro k se incremente, las métricas van a resultar en valores más favorables (cercanos a 1), ya que se cuenta con más información, y los tiempos de ejecución se van a incrementar de manera lineal, ya que se debe aumentar la cantidad de vecinos a evaluar por cada nuevo dato a clasificar.

Resultados:



Como se observa en los gráficos de las métricas según la cantidad de vecinos, y contradiciendo nuestra hipótesis, las mismas llegan a un valor máximo con un k = 1 y decrementan a partir de k = 5. Creemos que esto es a causa de que si uno tomase un k igual a la cantidad de datos en la muestra, la predicción será siempre la misma sin importar la imagen que se quiera predecir, y si uno tomase un k igual a 1 podría resultar ser poca información. Por lo tanto el k que maximiza las métricas debe ser uno intermedio, el cual en este caso en particular basto con k = 1.

Luego en el segundo gráfico podemos observar como los tiempos de ejecución parecerian no variar de forma significativa según el valor de k, contradiciendo nuestra hipótesis inicial. Creemos que esto es a causa de que la cantidad de vecinos mas cercanos que se deben buscar es muy pequeña comparada a la cantidad de datos totales entre los que se los busca. Por lo que suponemos que el tiempo de ejecucion podria llegar a incrementarse con valores de k mas grandes.

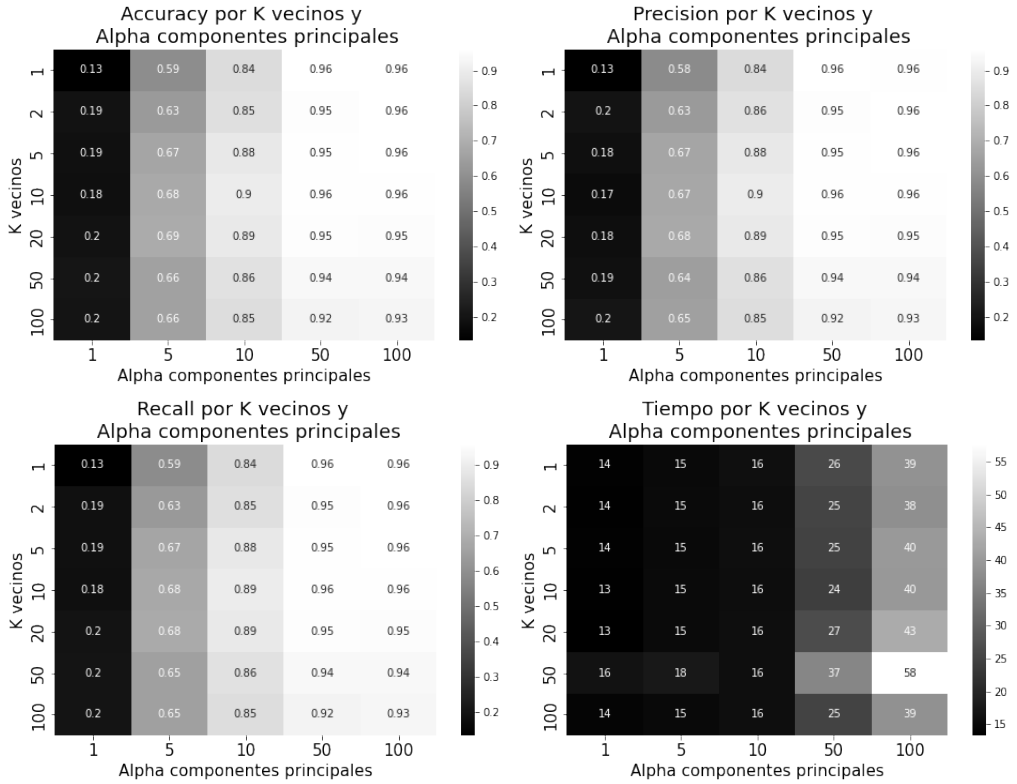
2.2. kNN con PCA

Para este experimento se va a entrenar el modelo con un k igual a 1, 2, 5, 10, 20, 50, 100 y α igual a 1, 5, 10, 50, 100. Por simplicidad de la experimentación, de los 42000 datos dados, siempre se van a utilizar un 80 % para entrenamiento y un 20 % para validación. Para evaluar los resultados utilizaremos las métricas de accuracy, precision y recall, y mediremos los tiempos de ejecución.

Hipótesis:

Creemos que a medida que los parámetros k y α se incrementen, las métricas van a resultar en valores más favorables (cercanos a 1), ya que se cuenta con más información, y los tiempos de ejecución se van a incrementar respecto de k y α, ya que se debe aumentar la cantidad de vecinos a evaluar por cada nuevo dato a clasificar junto con la cantidad de componentes a analizar.

Resultados:



Como se observa en los gráficos de las métricas según la cantidad de vecinos y cantidad de componentes, a excepción del caso de $\alpha = 1$, las métricas según k crecen hasta un máximo y luego decrecen. Creemos que esto ocurre por el mismo motivo que en kNN sin PCA, con la diferencia de que el valor óptimo de k es mayor a 1 para los alphas estudiados. Razonando sobre lo anterior sospechamos que este comportamiento se replica en $\alpha = 1$ pero con una mayor cantidad de vecinos.

Después podemos observar que siempre se da una mejora de las métricas a medida que alpha incrementa, apoyando nuestra hipótesis.

Finalmente en el segundo gráfico podemos observar como los tiempos de ejecución aumentan a medida que aumenta el k y el α, apoyando nuestra hipótesis inicial.

2.3. Conclusión

Los valores óptimos para los parámetros de entrada son un α lo más grande posible (784) con un k que maximiza las métricas. Creemos que dicho k debe ser menor a 5 ya que se observa que el valor de k que maximiza las métricas es cada vez menor a medida que se aumenta el α.

Si además tomamos en cuenta los tiempos de ejecución, uno podría optar por un α mucho más chico que el máximo, ya que la mejora en las métricas es pequeña comparado su costo temporal cuando el valor de α es muy grande. Luego por ejemplo convendría tomar $\alpha = 10$ y el k = 10 que maximiza las métricas y se calcula en un tiempo de 16 segundos (predecir sobre 0.2x42000 = 8400 imágenes) obteniendo un accuracy, precision y recall de alrededor de 0.9, o tambien se podria tomar $\alpha = 50$, k = 10, teniendo un tiempo de procesamiento mayor sobre la misma cantidad de imágenes de 24 segundos pero obteniendo métricas de alrededor de 0.96.

3. Variación de cantidad de imágenes en muestra

Buscamos ver la relación entre la cantidad de imágenes en la muestra y la calidad de los resultados del kNN. Con muy pocas imágenes el kNN quizá no es tan preciso y con gran cantidad de imágenes se estabilizará el resultado, es decir, a partir de cierta cantidad de imágenes, por más que continuemos agregando imágenes, la calidad del resultado no se modificará significativamente

Para ver esto se harán 2 experimentos:

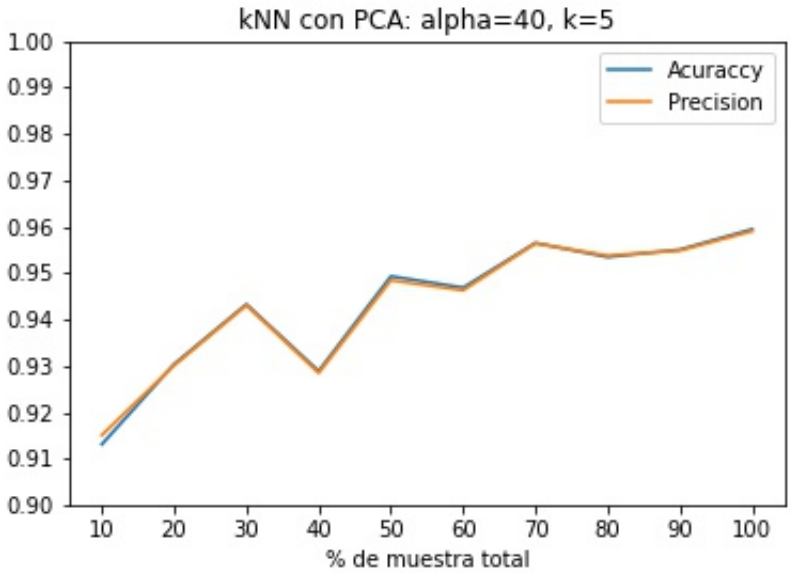
- Tomando porcentajes de la muestra total, empezando con 10 % y terminando con 100 %. Esto nos dará una visión general de los resultados.
- Empezando con 500 imágenes, sumándole 500 en cada vez hasta llegar al total de la muestra. Esto nos dará más detalle de cómo se va comportando el kNN con valores intermedios.

En cada experimento se usará k=5 y α=40

3.1. Experimento 1: Tomando porcentajes

Tomaremos primero el 10 % de la muestra total y luego vamos a ir sumando el 10 % de la muestra cada vez, es decir, vamos a tomar primero 10 %, luego 20 % y así hasta llegar a 100 % de la muestra total.

Calidad de datos:



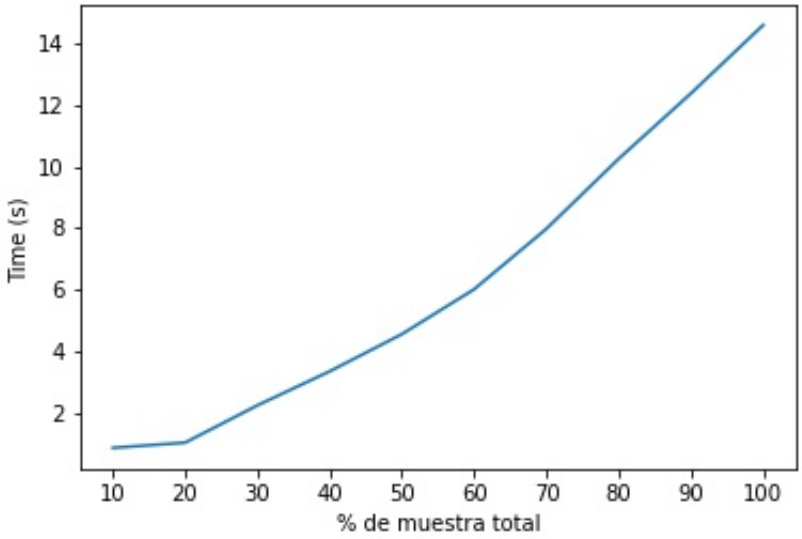
Vemos como del 10 % al 30 % la mejora es muy significativa, luego tiene una caída al 40 %*(*)* y desde ahí mejora hasta llegar al 70 % donde se estabiliza y no varía tanto en el resto de los porcentajes.

Con el 70 % (23519 instancias de entrenamiento y 5880 de validación) obtenemos un accuracy de 0.9565 y con el 100 % obtenemos uno de 0.9594. Tenemos una mejora del Accuracy de solo el 0.002942 y tardamos 6 segundos más en calcularlo, es decir, el costo que pagamos en tiempo no lo ganamos en calidad.

El accuracy y la precision tiene valores muy similares, esto se puede dar debido a que las clases están balanceadas.

Los máximos valores de accuracy y precision se obtienen con el 100 % de la muestra total pero, como dijimos antes, varían muy poco con los resultados obtenidos con el 70 % de la muestra total.

Tiempo de ejecución:



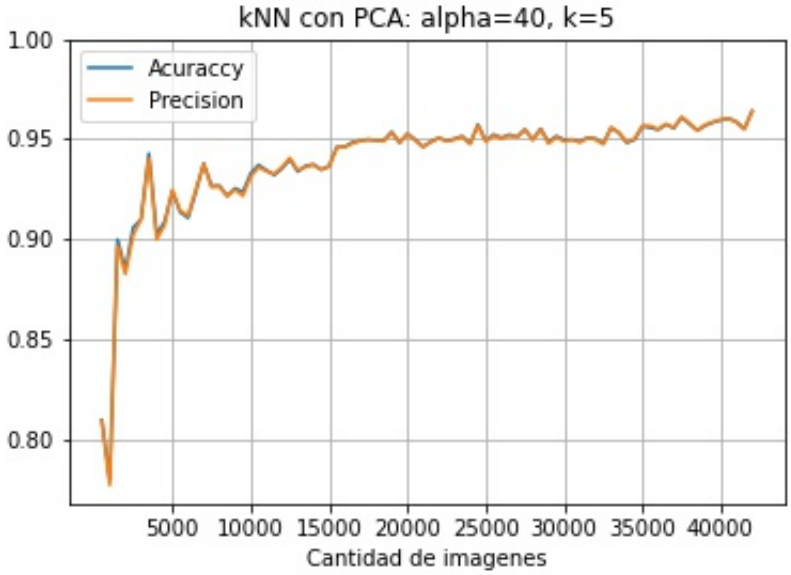
En todos los casos, cuanto más grande es la muestra, más tiempo se tarda en procesarla. En promedio, se tarda entre 1 y 2 segundos con cada 10 % (3360 instancias de entrenamiento y 840 de validación) de la muestra.

(*) La caida del 40 % pensamos que era un resultado atípico propio del experimento, sin embargo, se repitió varias veces dando como resultado siempre una caida cuando se pasaba del 30 % al 40 % de la muestra.

3.2. Experimento 2: aumentamos de a 500 imágenes

Tomaremos 500 imágenes y luego iremos ampliando la muestra sumándole 500 imágenes en cada iteración hasta llegar al total de la muestra con 42000 imágenes.

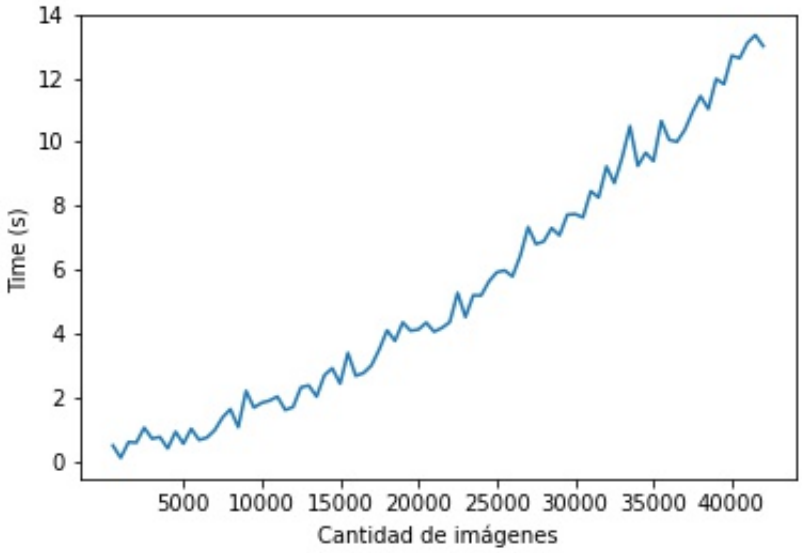
Calidad de datos:



Vemos que tiene una mejora significativa de la calidad entre las 500 y las 10000 fotos, luego tiene variaciones y se estabiliza cuando llega a las 20000 imágenes aproximadamente.

A las 24500 imágenes (19600 instancias de entrenamiento y 4400 de validación) se alcanza un accuracy de 0.9573 que es 0.0067 menor al accuracy máximo (0.9640) que se alcanza con 42000 imágenes (muestra completa). Es decir, se tarda el doble de tiempo pero la mejora que se obtiene es muy leve.

Tiempo de ejecución:



A diferencia del experimento anterior donde el tiempo de cada iteración siempre era mayor al tener mayor cantidad de imágenes, acá podemos ver una gran variación de los tiempos de ejecución pero siempre tendiendo a mayor tiempo cada vez.

3.3. Conclusión

Con ambos experimentos podemos observar que teniendo entre el 60 % (25200 imágenes) y el 70 % (29400 imágenes) se alcanza un accuracy y una precision muy similar a los valores máximos pero tardando la mitad del tiempo.

4. Variación k y cantidad de imagenes

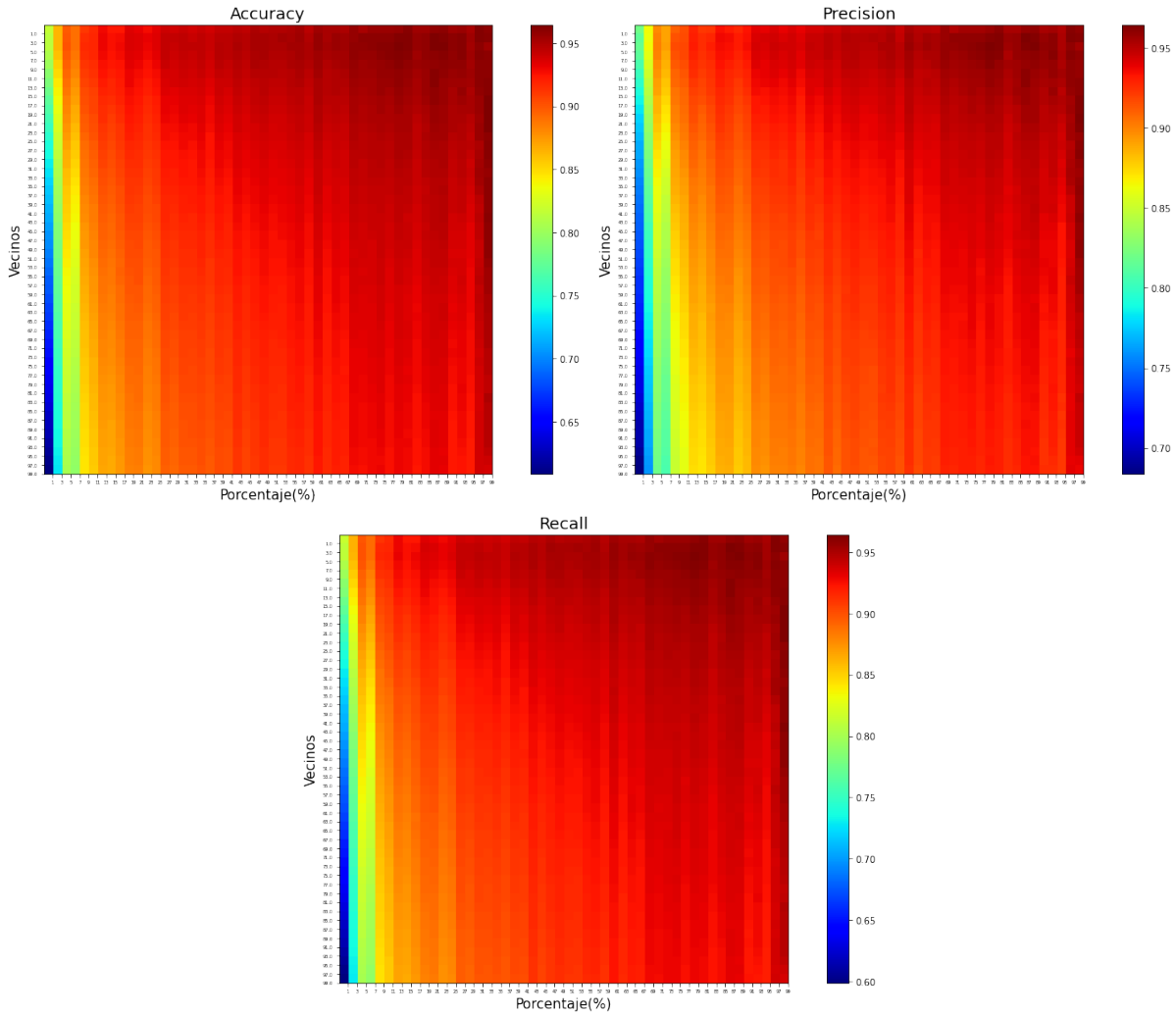
4.1. Experimento

Para realizar este experimento se fue variando el porcentaje de la muestra como conjunto de entrenamiento, el resto de la muestra se utiliza para evaluación, y con los distintos porcentajes ir variando la cantidad de vecinos.

Entonces los porcentajes de la muestra tomados se encuentran en el rango entre 1 % y 99 % y la cantidad de vecinos variaba entre 1 y 99. Nos limitamos a este número de vecinos ya que, aunque se había observado que la cantidad óptima de vecinos era alrededor de 5, se quería observar si con algún porcentaje de imágenes se observaba algo muy distinto. También, por una cuestión de velocidad, se usó un α igual a 40 para realizar el PCA.

4.2. Métricas: accuracy, precision y recall

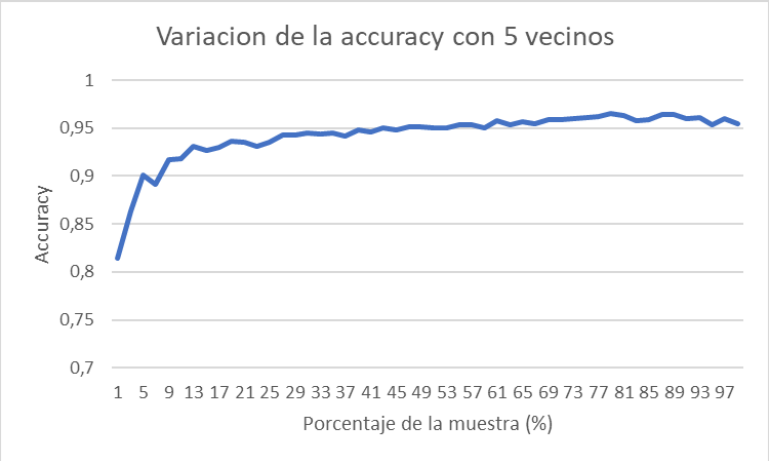
Para una mejor visualización, como eran dos parámetros los que variaban, se optó por realizar un mapa de calor para así observar cómo se comportan las métricas según las variaciones de los parámetros. Cabe aclarar, que con las tres métricas los resultados obtenidos eran prácticamente iguales por lo que, por cuestiones prácticas, de aquí en más cuando se nombren valores, nos vamos a referir a los de accuracy.



4.2.1. Fijando la cantidad de vecinos y variando el porcentaje de imágenes

Con esto ya se experimentó anteriormente y se llegó a la conclusión de que a mayor cantidad de imágenes mayores serán los valores de las métricas. Hay unos casos en los que con menos imágenes se obtiene un valor mayor de la métrica. Esto parece ocurrir debido a la distribución de las imágenes y que además, a mayor tamaño de la muestra del conjunto de entrenamiento, menor es la muestra de calificación, lo que implica que el cálculo de las métricas se ve afectado por esto. Aunque la tendencia es clara, y se llega a la misma conclusión que antes.

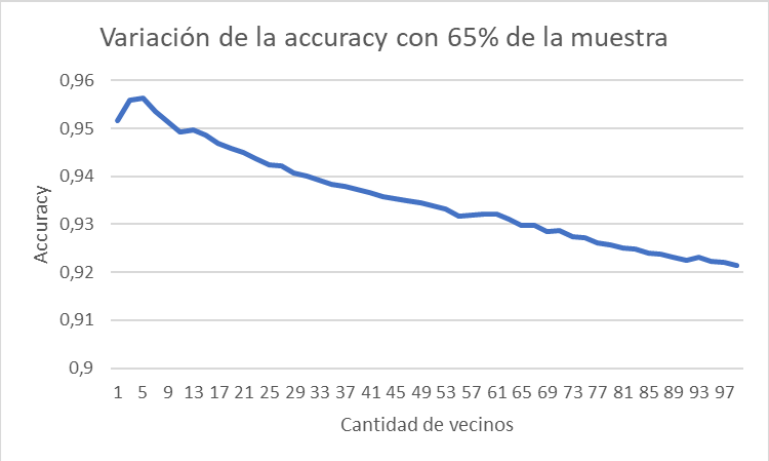
A continuación, se puede ver como varia el accuracy fijando k = 5



4.2.2. Fijando el porcentaje de imágenes y variando la cantidad de vecinos

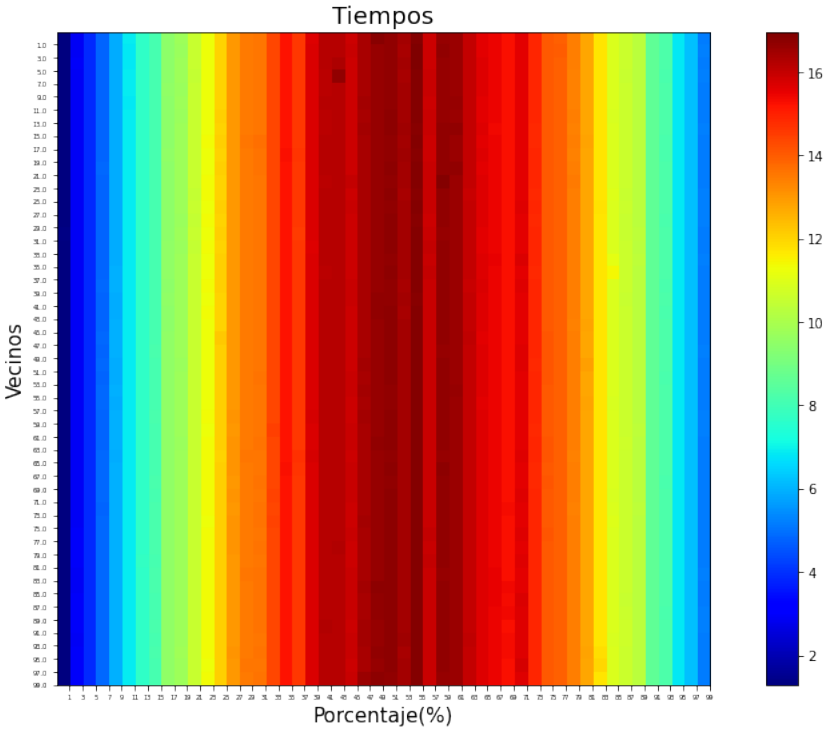
Al tener un porcentaje de imágenes fijo, se llega a una cantidad de vecinos donde las métricas se maximizan y luego comienzan a bajar. En 29 de los 50 porcentajes tomados, vemos que el accuracy se maximiza en k=5. En 17 casos lo hace en k=3. En 29 de los 50 porcentajes tomados, vemos que el accuracy se maximiza en k=5. En 17 casos lo hace en k=3. Lo más interesante es que cuando tomamos un 1 % de la muestra, el accuracy se maximiza con 1 vecino en 0,817, este valor resulta muy bajo respecto a otros valores con más porcentaje de la muestra, por ejemplo, para 9 % de la muestra, el valor más bajo que se obtiene es 0,846. También, con el 99 % de la muestra, el máximo se alcanza con K=17,19,21 y 23. Esto puede deberse a que solo se está calificando con **420** imágenes y la cual es relativamente una cantidad baja para reflejar la verdadera accuracy del experimento.

A continuación, se puede ver como varia el accuracy tomando un 70 % de las imagenes



4.3. Tiempos

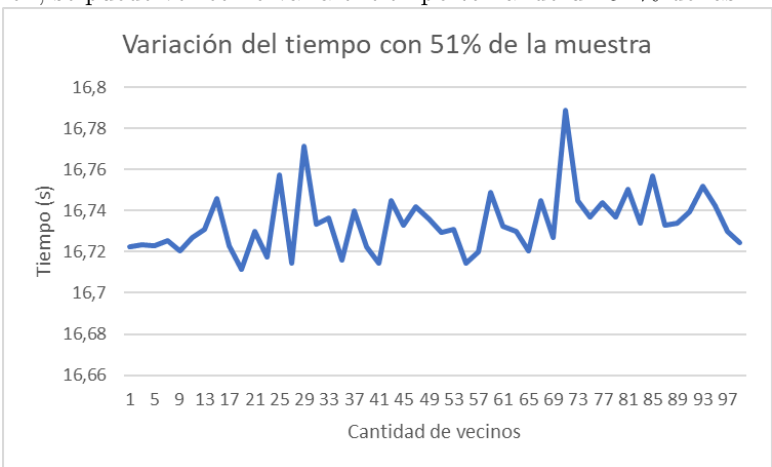
Los tiempos que se observan tienen en cuenta el proceso del PCA.



4.3.1. Fijando el porcentaje de imágenes y variando la cantidad de vecinos

Los tiempos que se manejan son todos prácticamente los mismos, con un ligero incremento a mayor cantidad de vecinos. Esto se debe a que en nuestro algoritmo de kNN, solo tomamos las k distancias necesarias.

A continuación, se puede ver como varia el tiempo tomando un 51 % de las imagenes



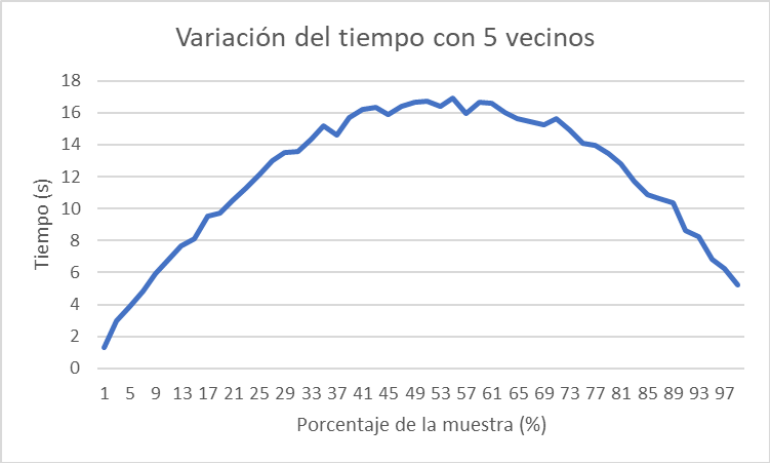
Se puede observar que el tiempo varía menos de una décima de segundo entre el mayor y el menor.

4.3.2. Fijando la cantidad de vecinos y variando el porcentaje de imágenes

A diferencia del experimento que se realizó anteriormente, lo que sucede es que experimentamos siempre con todas las imágenes y variamos la cantidad de las imágenes de los conjuntos de

entrenamiento y calificación. Entonces lo que se observa es que los tiempos varían en forma de parabola dado que el cálculo de distancias que se realiza entre n y (total-n) elementos es n*(total-n). Debido a cómo funciona el transform del PCA, que por ejemplo pasar 10 % de imágenes de entrenamiento y 90 % de calificación, el procesamiento del transform es el mismo que si tenemos 10 % de imágenes de calificación y 90 % de entrenamiento. Por esto y por como es el algoritmo, el transform tarda aproximadamente una cantidad constante de tiempo, en nuestro caso de un segundo.

A continuación, se puede ver como varia el tiempo fijando k = 5



5. Validación (K-Fold)

En esta sección vamos a corroborar que los valores señalados para cada parámetro del modelo de estudio (que fueron estudiados por separado en los experimentos previos) son valores útiles para la tarea de clasificación dada, al ser utilizados de forma conjunta.

La estrategia de validación cruzada, nos permitirá obtener más certeza de que el clasificador es capaz de sintetizar la información relevante que se le presenta en la etapa de entrenamiento, pudiendo generalizar, y evitando sucumbir ante casos de overfitting ("estudiar para sacar buena nota en el examen, pero no aprender realmente los contenido").

Para corroborar que el modelo "aprende" independientemente de cómo se realice la división entre datos de entrenamiento y datos de calificación, emplearemos una validación cruzada de K iteraciones ("K-Fold").[3]

La idea principal detras de este método es particionar el conjunto de muestras etiquetadas en K grupos de igual tamaño (de forma aleatoria, para evitar sesgos), y luego calcular varias versiones de la métrica de interés, entrenando al modelo con ciertos grupos y evaluando con los restantes.

Esto resulta en K valores resultantes para la métrica en cuestión que luego se pueden promediar para buscar una medida resumen representativa, o bien se puede tomar el mínimo para obtener una medida más conservadora. En la práctica esta última elección dependerá de la naturaleza específica de la aplicación. En este trabajo mostraremos ambos resultados, los cuales son similares debido a la uniformidad que existe entre las clases.

Para elegir el parámetro K a utilizar, buscaremos superar la cantidad ideal (estudiada en un experimento previo) de datos de entrenamiento, manteniendo así representatividad de los resultados, pero tomando el compromiso de utilizar valores relativamente bajos para mantener los tiempos de cómputo en valores asequibles.

El máximo valor teórico alcanzable para K sería el número de muestras etiquetadas, las cuales se deberían ir agrupando al momento de calificar, para poder tener una cantidad significativa de muestras, mitigando la volatividad de la calificación.

En nuestro caso particular, contamos 42.000 imágenes etiquetedas y previamente mostramos que la calidad del modelo tanto en accuracy como en precision desacelera considerablemente su creciemiento pasadas las 25 mil muestras de entrenamiento, sin detener su crecimento en tiempo de cómputo. Esto nos lleva a concluir que valores para K muy mayores a 3 no aportarían información nueva. Optamos por experimentar con K=3, K=4 y K=5.

Es decir, los experimentos que realizaremos sobre nuestro modelo son validaciones cruzadas a 3, 4 y 5 vías (K), con accuracy como métrica de interés, preprocesando los datos mediante un análisis de 40 (α) componentes principales y clasificándolos en base a sus 5 (k) vecinos más cercanos.

Resultados

K	Promedio	Mínimo
3	0.955190	0.952928
4	0.956952	0.952857
5	0.957333	0.955119

Como esperábamos observar, aunque valores de K mayores implican entrenar con más imágenes, no incrementan significativamente los resultados.

6. Conclusión

Luego de haber realizado todos los experimentos podemos llegar a la conclusión que utilizando en kNN un k entre 5 y 10, en PCA un α entre 40 y 50, y entre el 60 % y el 70 % de la muestra total, se llega a una calidad de datos muy parecida que utilizando valores mucho más altos de α o imágenes pero tardando mucho menos tiempo.