UNIVERSIDAD
NACIONAL
DE COLOMBIA

Técnicas en Aprendizaje Estadístico

# Trabajo 4

*Cleyver Ramírez*
*Alberto Ceballos*
*Mauricio Soto*

Docente
Juan David Ospina.

23 de marzo de 2019

# Índice

# Índice de figuras

# Índice de cuadros

# 1. Introducción

Los clasificadores multiclase son conjuntos de distintos clasificadores binarios que los cuales al combinarse permiten realizar predicciones sobre múltiples clases. En este trabajo se tiene el objetivo de entrenar modelos para la clasificación de dígitos (del 0 al 9) con base en el conjunto de datos MNIST. Para lo anterior, se propone explorar versiones multiclase de los siguientes algoritmos con sus respectivas siglas:

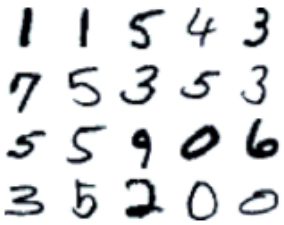
- Regresión Logística (GLM).
- Árbol de Clasificación (CART).
- Bosque Aleatorio (RF).
- Máquina de Soporte Vectorial (SVM).
- Redes Neuronales (NN).

Se pretende en últimas generar la base para un sistema que permita leer los códigos postales en cartas a fin de automatizar el procesamiento de correspondencia. El trabajo está organizado de la siguiente forma: en la sección 2 se hace una breve introducción al problema de estudio. En la sección 3 se detalla la estrategia metodológica escogida, mientras que en la sección 4 se ahonda en la implementación de la misma. En la sección 5 se muestran los resultados, en la sección 6 se discuten los resultados y en la sección 7 se muestra la plataforma web de clasificación de dígitos. Luego, en la sección 8, se explica la propuesta de sistema de automatización del procesamiento de correspondencia, y, finalmente, en la sección 9 se muestran algunas conclusiones.

## 2. Planteamiento del problema

El conjunto de datos MNIST (base de datos modificada del Instituto Nacional de Estándares y Tecnología) es una gran base de datos de dígitos escritos a mano que se usa comúnmente para capacitar a varios sistemas de procesamiento de imágenes [LeC+98]. Fue creada mezclando de nuevo las muestras de los conjuntos de datos originales de MNIST. Los datos de capacitación se tomó de los empleados de la oficina del censo de los estados unidos, mientras que el conjunto de datos de las pruebas se tomó de los estudiantes de secundaria estadounidenses. Además las imágenes se normalizaron en un cuadro delimitador de 28x28 píxeles y con anti-aliasing, lo que introdujo niveles de escala de grises.

Figura 1: Imágenes de muestra del conjunto de datos MNIST



Con estos datos se pretende:

- Construir, validar y comparar los cinco métodos de clasificación mencionados en un problema multiclase bajo varias configuraciones. El problema a resolver será la clasificación de dígitos.

- Proponer una arquitectura básica para explotar el mejor modelo en la práctica de leer códigos postales y clasificar cartas. Esta propuesta se verá como un esquema de componentes conectados con su debida explicación de cada componente y cada conexión.

# 3. Estrategia Metodológica escogida

La estrategia metodológica se basa en los siguientes 4 pasos:

1. Separación Entrenamiento-Prueba
2. Validación Cruzada
3. Entrenamiento Final
4. Comparación

Por familia de modelos entenderemos las 5 clases de modelos a comprar: GLM, SVM, CART, RF, NN. La familia se forma al considerar todas las posibles combinaciones de los parámetros que tiene cada familia.

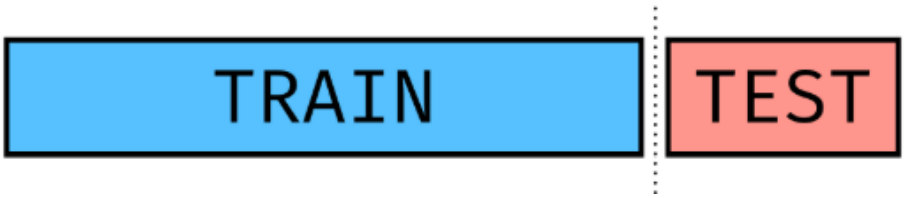



Figura 2: Separación de datos en Entrenamiento (TRAIN) y Prueba (TEST)

## 3.1. Separación Entrenamiento-Prueba

Los datos del conjunto MNIST vienen ya con una separación de los datos en conjuntos de entrenamiento y prueba como se esquematiza en la figura 2. En particular, el conjunto de entrenamiento está compuesto por 60 mil imágenes con sus respectivas etiquetas que relacionan la imagen con el número del 0 al 9 que esta escrito en ella. El conjunto de prueba, en cambio, contiene 10 mil imágenes también etiquetadas.

Esta división permite una justa evaluación entre modelos de diferentes familias, osea, modelos entrenados y evaluados sobre los mismo datos.

## 3.2. Validación Cruzada

El objetivo de la validación cruzada es el de tener una medida de comparación del rendimiento entre diferentes modelos de la misma familia (utilizando diferentes parámetros del mismo modelo) donde el rendimiento se mide sobre datos no incluidos en el entrenamiento.
Por ejemplo, la figura 3 muestra la separación del conjunto de entrenamiento en 3 pliegues. El procedimiento de validación cruzada comprende la iteración en cada uno de estos pliegues, entrenar un modelo en los datos TRAIN y medir el rendimiento de éste modelo sobre los datos VALIDATION.

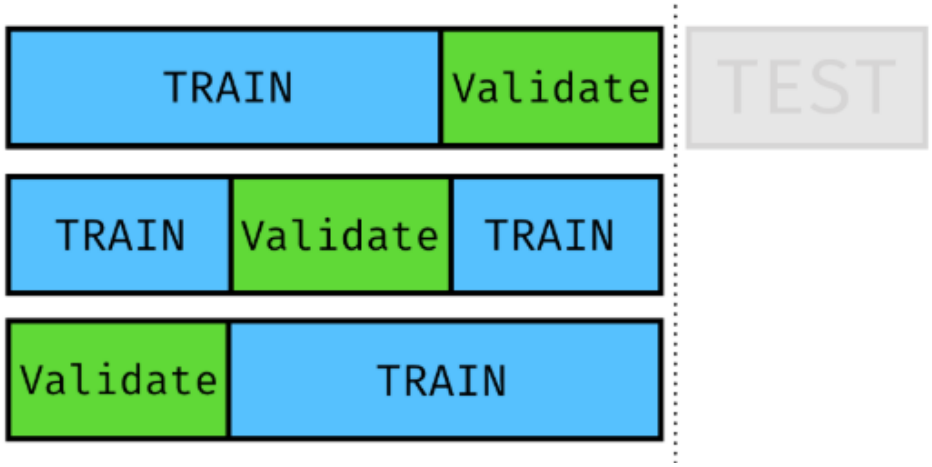



Figura 3: Separación de Entrenamiento(TRAIN), Validación(VALIDATION)

Para construir $n$ pliegues de una validación cruzada, se definen $n$ subconjuntos con datos mutuamente excluyentes y escogidos aleatoriamente del conjunto de entrenamiento.
Finalmente, el rendimiento del modelo sobre los datos VALIDATION se promedian sobre el número total de pliegues para dar una medida aproximada del rendimiento en datos no vistos.
La validación cruzada es una medida muy eficaz para combatir el problema de sobreajuste de modelos.
De gran importancia es que se definan los pliegues una sola vez para cada familia y luego todos los modelos, de esa familia, sean evaluados sobre la misma definición de pliegues. Esto garantiza que el rendimiento los modelos dentro de la misma familia sean comparados justamente.
Es posible definir diferentes pliegues para evaluar los modelos de diferentes familias ya que la comparación entre diferentes familias se hace sólo sobre los datos de prueba.

### 3.3. Entrenamiento Final

El entrenamiento de un modelo final para cada familia se hace utilizando todos los datos de entrenamiento y los mejores parámetros hallados en el paso de "Validación Cruzada"

### 3.4. Comparación

Se utiliza el modelo hallado en el paso de .Entrenamiento Final"para predecir la etiqueta de los datos en el conjunto de prueba y comparar los rendimientos de las 5 familias de modelos.

# 4. Implementación de la Estrategia

A continuación se detalla la implementación de la estrategia metodológica para cada una de las familias de modelos. En cada subsección hay información sobre las librerías empleadas, parámetros considerados y los resultados intermedios. Adicionalmente, se pone a disposición un archivo en Markdown con todo el código para producir las imágenes en este reporte:

https://sotomsa.shinyapps.io/Trabajo4/Exploracion.html

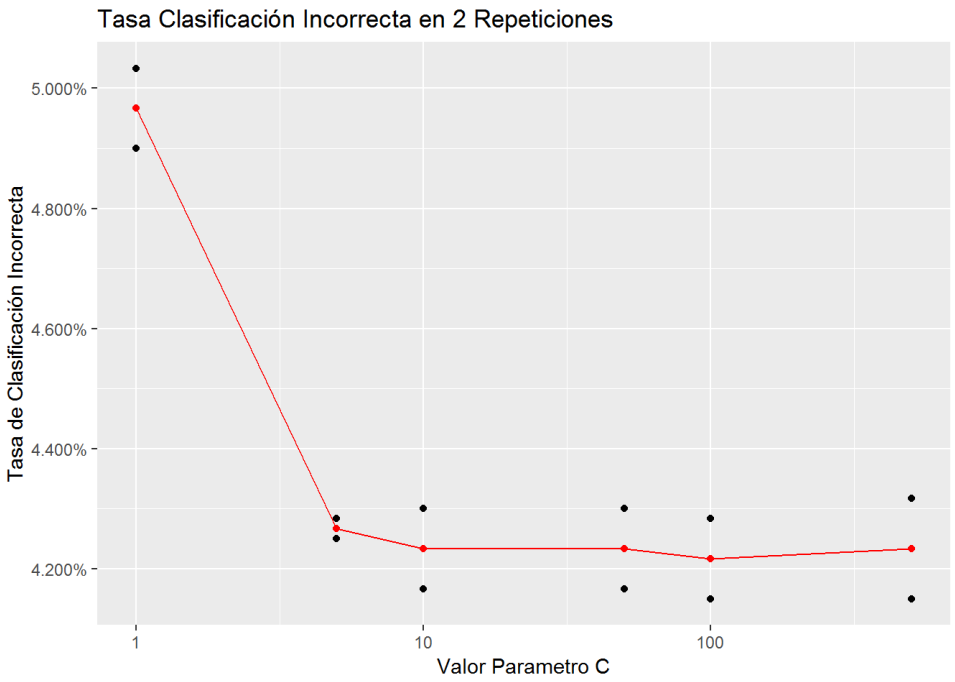



Figura 4: Validación cruzada en SVM

## 4.1. Máquina de Soporte Vectorial (SVM)

Para este modelo se utilizo la libreria Kernlab [CRAb]. Los principales párametros de este modelo son C (el error total admisible de los puntos respecto a los hiperplanos separadores) y el tipo de Kernel que se usa. Por defecto utiliza el kernel Gaussiano. Se decidió evaluar esta familia sobre el valor del parámetro C.

Dado el alto costo de memoria y computacional de este modelo, el entrenamiento se realizo en una instancia de Amazon EC utilizando una implementación de la libreria "Multidplyr"que aprovecha todos los procesadores.

La validación cruzada se realizó usando 5 pliegues y 2 repeticiones sobre el 10 % de los datos y los resultados se pueden ver en la figura 4

## 4.2. Modelo Lineal Generalizado (GLM)

Se utilizo la librería "glmnet"que es una versión mejorada de la librería "GLM". La librería "glment"[CRAa] implementa lo que llaman GLM+Elasticnet. La idea básica detrás de de la versión con Elasticnet es un término de penalización que se agrega a la función objetivo y que permite penalizar los valores de los coeficientes con la norma L1(Lasso) y L2(Ridge) simultáneamente según el valor del parámetro alpha. Para valores de alpha igual a 0, el método es GLM Ridge puro. Para valores alpha iguales a 1, el método es GLM Lasso puro. Para valores intermedios de alpha en (0-1), el modelo se comporta como una red elástica entre Lasso y Ridge. Estas penalizaciones tienen como objetivo obtener modelos con menos sobreajustados ya que limita el crecimiento de los coeficientes.

La validación cruzada se realizó usando 5 pliegues y 2 repeticiones sobre el 10 % de los datos.

## 4.3. Árboles de Clasificación y Regresión (CART)

Para los árboles de clasificación y regresión se utilizó la librería rpart"[CRAd]. Se consideraron solo dos versiones de este modelo. Una versión que utiliza la medida "Gini"para escoger la variable a dividir en cada nodo y la versión que utiliza la medida ïnformation"para hacer la misma tarea.

Dado que esta familia solo considera estos dos modelos, ambos se entrenaron en todos los datos de entrenamiento y se compararon como familias diferentes usando los datos de prueba.

## 4.4. Bosques Aleatorios (Random Forests)

La librería que se empleó para los bosques aleatorios fue ranger"[CRAc]. Esta es una librería es una versión más rápida sobre la librería original randomForest". Para esta familia de modelos se consideró el parámetro num.trees para evaluar los modelos.

La validación cruzada se realizó usando 5 pliegues y 2 repeticiones sobre el 10 % de los datos y los resultados se pueden ver en la figura 5.

## 4.5. Redes Neuronales (NN)

Finalmente, para el modelo de redes neuronales se utilizó la librería "mxnet"[Che+15]. La arquitectura de la red es la arquitectura presentada en clase y la optimización se hizo con base en el articulo de George Liu [Liu].

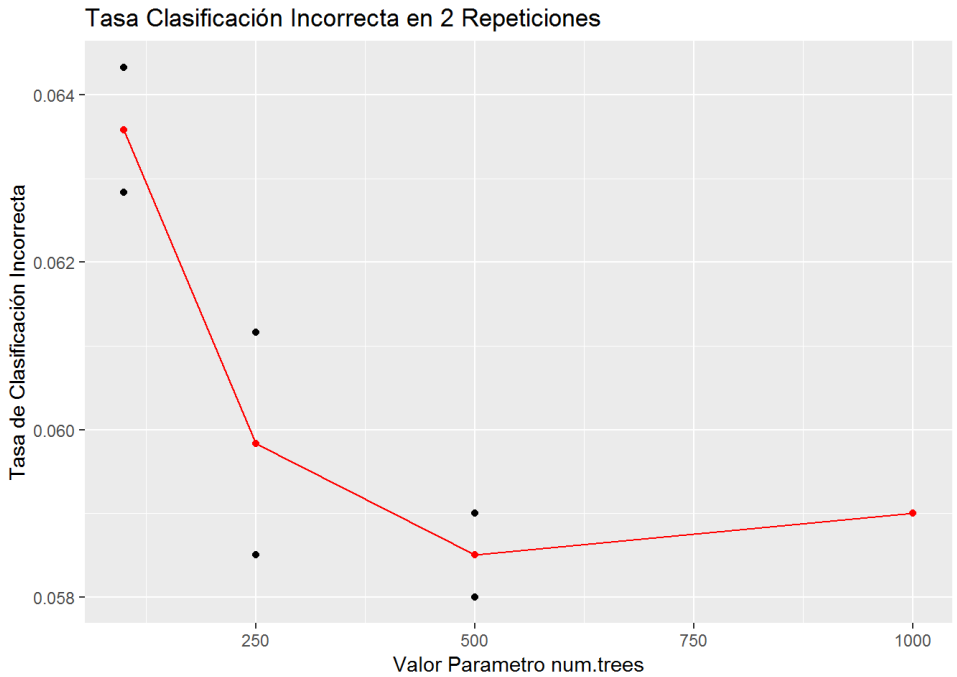



Figura 5: Validación cruzada en Bosques Aleatorios (Random Forests)

La validación cruzada se realizó con 10 pliegues y una repetición sobre el 10 % de los datos a diferencia de la validación cruzada en otras familias de 5 pliegues y 2 repeticiones sobre la misma proporción de datos.

La optimización sugerida por George Liu es un acercamiento donde se optimizan secuencialemnte los párametros, de a uno a la vez (Greedy Approach). Primero se optimiza la tasa de aprendizaje (learning.rate) como se ve en la figura 6, luego el número de datos en cada lote (num.batch) como se ve en la figura 7 y por último se optimiza el número de iteraciones (epochs) como se ve en la figura 8.

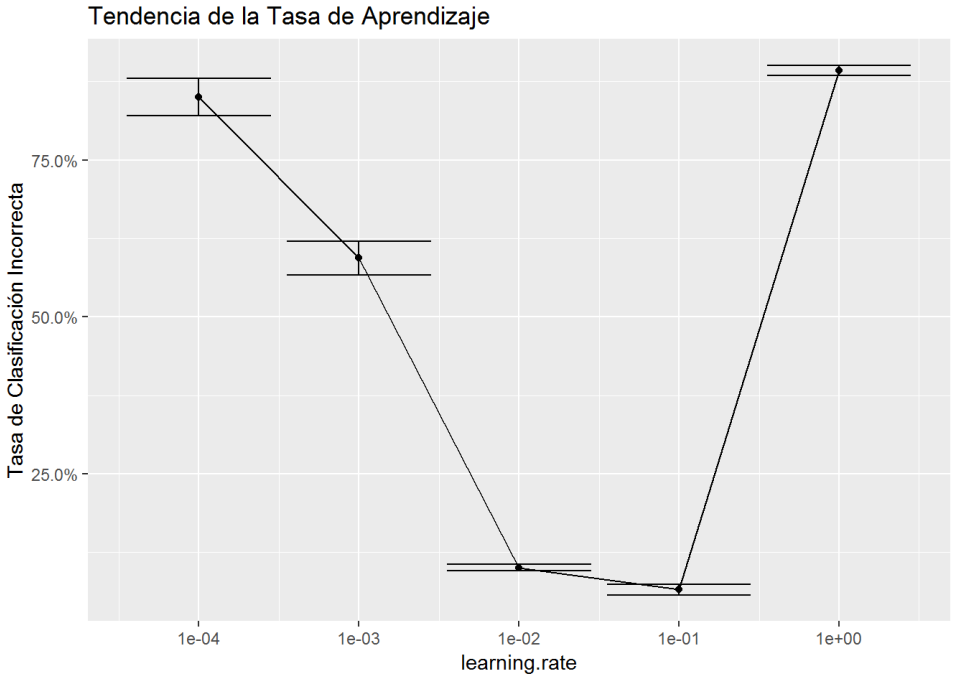



Figura 6: Validación cruzada en Tasa de Aprendizaje

# 5. Resultados

La tabla 1 contiene un panorama de todos los experimentos realizados en este trabajo. Cada columna describe un aspecto diferente de los modelos que se construyeron para predecir los números del 0 al 9 en las imágenes del conjunto de datos MNIST. Por ejemplo, los tiempos de entrenamiento y predicción nos hablan de costo computacional, los parámetros nos dicen los grados de libertad que se evaluaron en el los modelos, el tamaño del modelo (en disco) se refiere a los costos de memoria una vez que el modelo ha sido entrenado. Finalmente, la tasa de clasificación incorrecta es una medida del rendimiento.

Es importante recalcar que los valores obtenidos dependen del tipo y cantidad de datos del dataset utilizado y algunas de las conclusiones que se sacan de los datos no aplican para otros datasets.

## 5.1. Costo Computacional

Podemos medir el costo computacional de un modelo en dos etapas: entrenamiento y predicción. En el caso del entrenamiento, el modelo GLM+Elasticnet fue el que más tiempo tardó con una hora y seis minutos, mientras que CART fue el que menor tiempo tomó con 3.5 minutos. En cuanto a la predicción, el modelo que más tarda es el SVM y los que menos tardaron fueron CART y Redes Neuronales con 0.74 segundos.

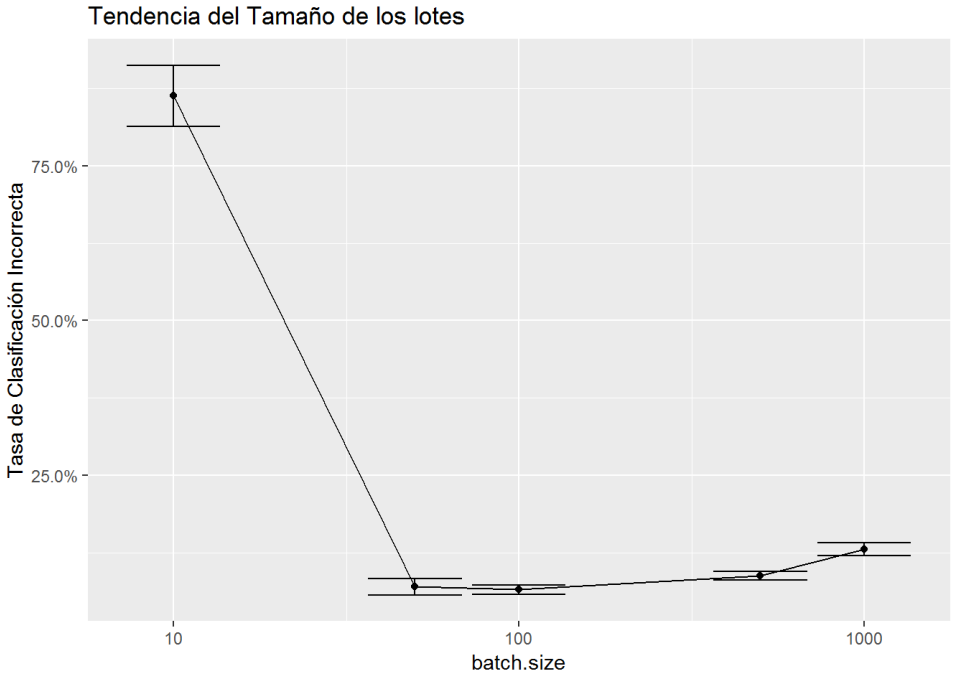



Figura 7: Validación cruzada en Tamaño del Lote

## 5.2. Parámetros Evaluados

Otro punto de comparación es el número de parámetros que se deben ajustar. Es de anotar que los parámetros que evaluamos no son todos los posibles parámetros que tiene cada modelo. Sin embargo, los parámetros escogidos fueron aquellos que consideramos con mayor importancia en el rendimiento final del modelo.

Los modelos SVM y Random Forest son modelos con pocos parámetros y esto los hace muy atractivos porque se facilita el entrenamiento. Esto es aún más notorio en el caso del modelo Random Forest que adicionalmente tienen bajos tiempos de entrenamiento.

En el otro extremo están las Redes Neuronales, donde solo se evaluaron 3 parámetros, pero que tiene muchos más grados de libertad como la distribución de inicialización de los parámetros, número de capas ocultas, número de neuronas en las capas ocultas, el optimizador, el momento del optimizador, entre otros.

## 5.3. Tamaño del Modelo en Disco

Los modelo que requiere mayor espacio en disco son el Random Forests y el SVM, mientras los que tienen menor costo en memoria son el GLM+Elasticnet y CART. El modelo de redes neuronales tiene un tamaño bastante bajo con respecto a Random Forests y el SVM.

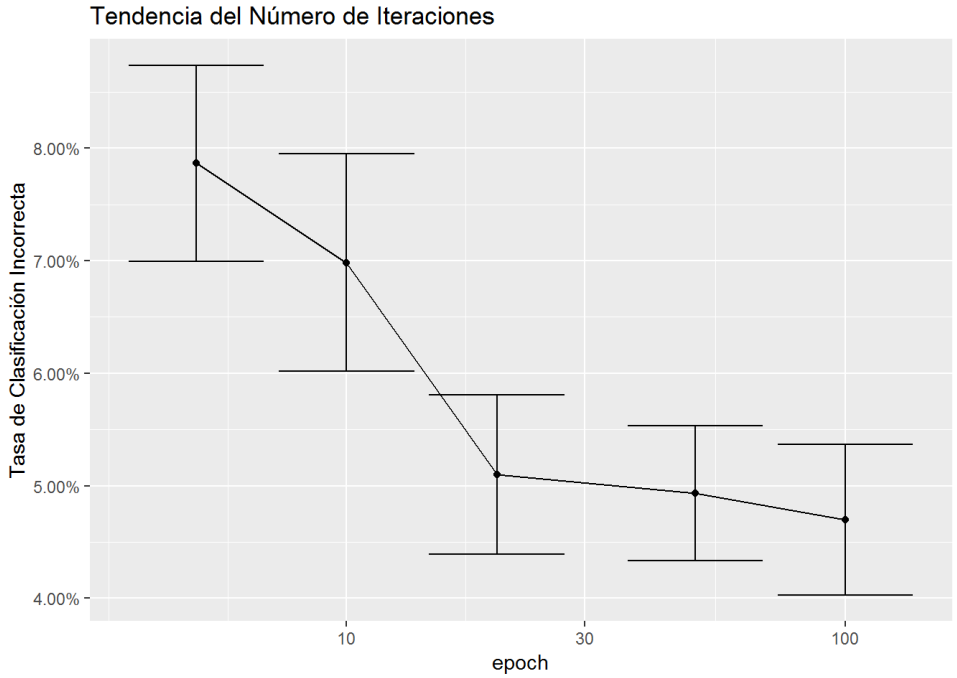



Figura 8: Validación cruzada en el Número de Iteraciones

El tamaño del modelo en disco es de alguna forma un poco subjetivo ya que es posible que los modelos guarden más información de la que necesitan para predecir. Es el caso del SVM, que en teoría, no debería de ocupar mucho en disco.

Por otro lado, tiene mucho sentido que Random Forest necesite mucha memoria dado que se entrenan tantos arboles, aunque también podría estar pasando algo similar a lo que pasa con el SVM.

## 5.4. Tasa de Clasificación Incorrecta

Finalmente, esta la columna que mide el rendimiento de los modelos en datos no vistos. Los tres modelos ganadores son:

1. SVM (1.76 %)
2. Redes Neuronales (2.42 %) - Random Forests (2.82 %)

El SVM tiene la mejor tasa de todos los modelos y las Redes Neuronales con Random Forests tienen casi un empate. Los rendimientos mostrados por GLM+Elasticnet (88.65 %) y CART (34.07 %) realmente no fueron para nada buenos.

| Nombre Modelo | Tiempo Entrenamiento | Tiempo Predicción | Parámetros Evaluados | Tamaño Modelo | Tasa de Clas. Incorrecta |
|---|---|---|---|---|---|
| SVM | 32 min. | 4 min. | C | 10.5Mb | 1.76 % |
| GLM+Elasticnet | 66 min. | 2.6 seg. | alpha, lambda | 1KB | 88.65 % |
| CART | 3.5 min. | 0.74 seg. | gini/information | 260KB | 34.07 % |
| Random Forests | 5 min. | 9 seg. | num.trees | 30MB | 2.82 % |
| Redes Neuronales | 8.5 min. | 0.73 seg. | learning.rate<br>batch.size<br>num.epochs | 500KB | 2.42 % |

Tabla 1: Resumen de Resultados. Tiempo de Entrenamiento sobre los 60K imágenes del Train. Tiempo de Predicción sobre los 10K datos de Test.

## 6. Discusión de Resultados

Sobresale mucho el buen rendimiento de los Random Forests ya que tiene pocos parámetros que evaluar y bajos tiempos de entrenamiento. Esto lo hace un modelo que está en los primeros puestos de la lista a probar, para luego usar su rendimiento como base para comparar otros modelos más costos de construir.

Por el otro lado, se observa que el modelo GLM+Elasticnet no es un modelo que funcione bien con el tipo de datos del MNIST ya que obtuvo un rendimiento muy bajo y fue el modelo que tomó más tiempo en entrenar. Es muy probable que la naturaleza lineal del modelo GLM hace que no pueda lidiar con la, muy probable, naturaleza no lineal del problema.

Aunque el modelo CART no tuvo un muy buen rendimiento en este dataset, los bajos tiempos de entrenamiento y el hecho de tener virtualmente ningún parámetro lo hacen muy atractivo para datasets más sencillos.

Con respecto al modelo SVM, su mayor ventaja es tener pocos parámetros pero sus mayores desventajas son los altos tiempos de entrenamiento, predicción y tamaño en disco. A pesar de ello, éstas parecen estar compensadas por el buen rendimiento mostrado en este problema.

Finalmente, los modelos de Redes Neuronales obtuvieron el segundo mejor rendimiento entre los modelos probados. Estos modelos son, muy probablemente, la primera opción para este tipo de datos dado que ya han sido utilizadas muy exitosamente en conjuntos de muchos datos e imágenes, pero su utilización presenta cierto riesgo ya que es muy posible que no se encuentre un buen modelo aunque éste exista. Las Redes Neuronales presentan un compromiso entre la posibilidad de explorar muchas posibles configuraciones dada su bajo tiempo de entrenamiento y paralelización, y la posibilidad de

no encontrar un buen modelo o encontrar el mejor de todos.

# 7. Aplicación Web

Este trabajo esta acompañado por una aplicación WEB que muestra el funcionamiento de los modelos entrenados en este trabajo. La aplicación se puede encontrar en:

https://sotomsa.shinyapps.io/Trabajo4/

En la figura 9 se puede ver una imagen de la aplicación web.

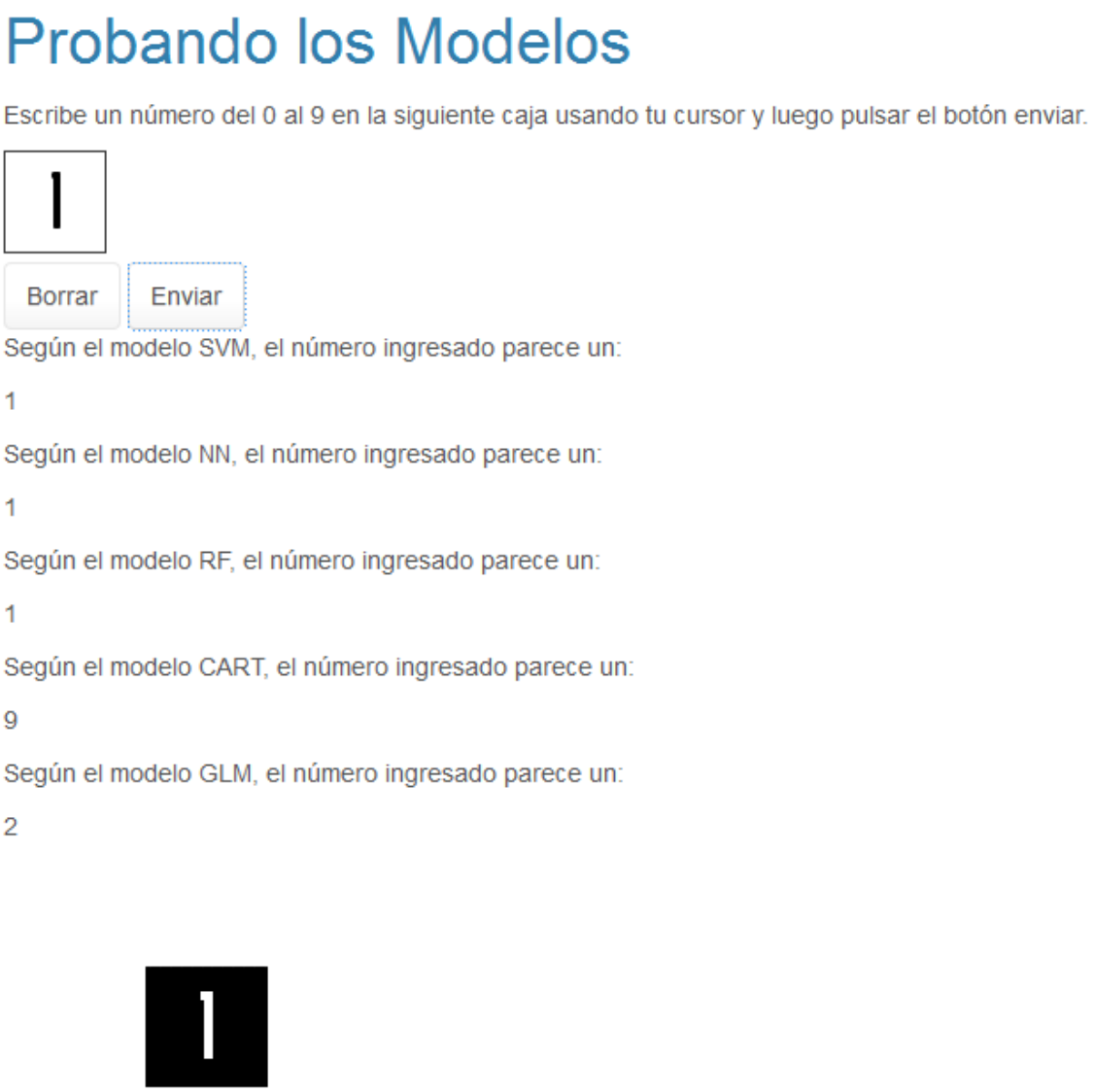



Figura 9: Imagen de la Aplicación Web construida para este Trabajo

## 7.1. Arquitectura Página Web

La aplicación web fue construida usando la librería Shiny de Rstudio. La página cuenta con un elemento del estándar HTML5 llamado canvas que permite dibujar utilizando el cursor del mouse. El código del Javascript del canvas está basado en el elemento canvasSimple de William Malome [Mal].

Una vez dibujado un número en el canvas y que se ha presionado el botón .ºEnviar", un código en Javascript toma los datos del canvas (codificado Base64), y los envía a un elemento reactivo en el servidor RStudio. Allí se decodifica, se re-dimensiona a una imagen de 28x28, se vectoriza y se pone como entrada a las funciones de predicción de los modelos pre-entrenados. Los resultados de estos modelos luego se envían a la interfaz web de la aplicación para que sean enseñados al visitante.

# 8. Propuesta de sistema de procesamiento de correspondencia

Se propone un sistema de procesamiento de correspondencia en Estados Unidos con base en el mejor modelo implementado. Para esto, se considera una serie de etapas descritas a continuación:

## 8.1. Precondiciones del sistema

Se presupone que las cartas tienen un único código postal, el cual puede ser el código básico de 5 dígitos o el extendido de 9 dígitos. Se asume, además, que solo se procesarán cartas en sobres típicos y no correspondencia como paquetes y cajas.

## 8.2. Separación de cartas

Como primer paso del sistema es necesario separar de alguna manera las cartas. El propósito de este paso es garantizar que, en la mayoría de casos, se trabaje con una única carta por imagen.

## 8.3. Adquisición de la imagen

Una vez se garantice que una sola carta será procesada a la vez, es necesario adquirir la imagen digital de la misma. Esto puede ser un problema si la adquisición se realiza solo para una cara de la carta, ya que no existe garantía de que se tome una foto del lado con el código postal. Debido a esto, se propone un sistema de adquisición en el que, una vez tomada la primera foto, esta sea volteada para tomar dos fotos sin mover la cámara.

El mecanismo anterior puede ser complicado de implementar, por lo que una alternativa es ubicar la carta sobre una superficie transparente y no tomar una sino dos fotos, una por cada cara de la carta. Esto se puede lograr cambiando la cámara de ubicación o, en su defecto, trabajando con dos cámaras.

Sea cual sea la implementación, se espera que tras esta fase se tengan dos fotos de la carta, una por cada cara.

## 8.4. Rectificación

Para simplificar las fases posteriores, se rectifican ambas imágenes de la carta. Esto es relativamente fácil de lograr si se segmenta la carta del fondo y se usan las esquinas de la misma como referencias para una transformación de rotación (asumiendo que la cámara esté ubicada perpendicular a la superficie y que la foto sea ortogonal) que haga que el eje más largo de la carta quede paralelo al eje $x$.

## 8.5. Segmentación y agrupación

Se segmentan los componentes de interés en la carta mediante algún algoritmo de segmentación, considerando que en la mayoría de casos los caracteres resaltarán con respecto al fondo. Un preprocesamiento útil antes de este paso puede ser balancear la iluminación por medio de un algoritmo como el filtro homomórfico [PP11]. En cualquier caso, una vez se segmentan los componentes de interés, es posible caracterizar los componentes presentes en la misma. El resultado de este paso son dos imágenes binarias donde se encuentran separados los caracteres candidatos a ser parte de los códigos postales.

Análogamente, es parte importante del proceso agrupar los caracteres candidatos en "palabras". Ya que en la mayoría de los casos estarán agrupados horizontalmente, se pueden utilizar técnicas morfológicas en el eje $x$ para obtener las regiones candidatas.

## 8.6. Lectura del código postal y toma de decisiones

Para cada región candidata, se analiza cada carácter usando nuestro clasificador. De esta manera se pretende discriminar números de caracteres. Esto implica re-entrenar el modelo con un conjunto significativo de datos alfanuméricos.

Para que una región candidata sea considerada como candidata a código postal, todos los caracteres presentes deben ser números o el guión correspondiente al código extendido. Asimismo, estas regiones deben contener 5 o 10 caracteres.

En caso de tener más de una región candidata, ambas son validadas con respeto al catálogo de códigos postales de los Estados Unidos. Si hay una única coincidencia, la carta es despachada. En caso de que no hayan coincidencias o de que haya más de una coincidencia, la carta es separada y el caso es reportado para su análisis manual.

# 9. Conclusiones

- Los modelos que obtuvieron las tasas de error más bajas fueron las Máquinas de Soporte Vectorial (SVM), Redes Neuronales (NN) y los Árboles Aleatorios (RF) en ese orden. Los tres modelos con tasas de clasificación incorrecta de menos del 3 %.
- El modelo más fácil de probar fue el de Árboles Aleatorios dado su rapidez para entrenar, pocos parámetros y alto rendimiento.
- El Modelo Lineal Generalizado (GLM) fue el modelo más lento y de menor rendimiento.
- Se demostró el funcionamiento de los modelos entrenados con una pequeña aplicación web.
- Se propuso un esquema de arquitectura para un sistema de procesamiento automático de correspondencia postal. El sistema requiere el uso de datos adicionales para su ejecución, así como de implementos físicos, pero puede ser de gran utilidad para agilizar los procesos de empresas de envíos.

# 10. Bibliografía

## Referencias

[LeC+98] Yann LeCun y col. "Gradient-Based Learning Applied to Document Recognition". En: *PROC. OF THE IEEE* (1998). URL: http://ieeexplore.ieee.org/document/726791/#full-text-section.

[PP11] C J Prabhakar y Praveen Kumar P. "an Image Based Technique for Enhancement of Underwater Images". En: *International Journal of Machine Intelligence* 3.4 (2011), págs. 217-224.

[Che+15] Tianqi Chen y col. "MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems". En: *Neural Information Processing Systems, Workshop on Machine Learning Systems*. 2015.

[CRAa] CRAN. *CRAN: Glmnet*. URL: https://cran.r-project.org/web/packages/glmnet/index.html.

[CRAb] CRAN. *CRAN: Kernlab*. URL: https://cran.r-project.org/web/packages/kernlab/index.html.

[CRAc] CRAN. *CRAN: Ranger*. URL: https://cran.r-project.org/web/packages/ranger/index.html.

[CRAd] CRAN. *CRAN: Rpart*. URL: https://cran.r-project.org/web/packages/rpart/index.html.

[Liu] George Liu. *Optimizing Neural Networks — Where to Start?* URL: https://towardsdatascience.com/optimizing-neural-networks-where-to-start-5a2ed38c8345.

[Mal] William Malome. *Create a Drawing App with HTML5 Canvas and JavaScript*. URL: http://www.williammalone.com/articles/create-html5-canvas-javascript-drawing-app/.