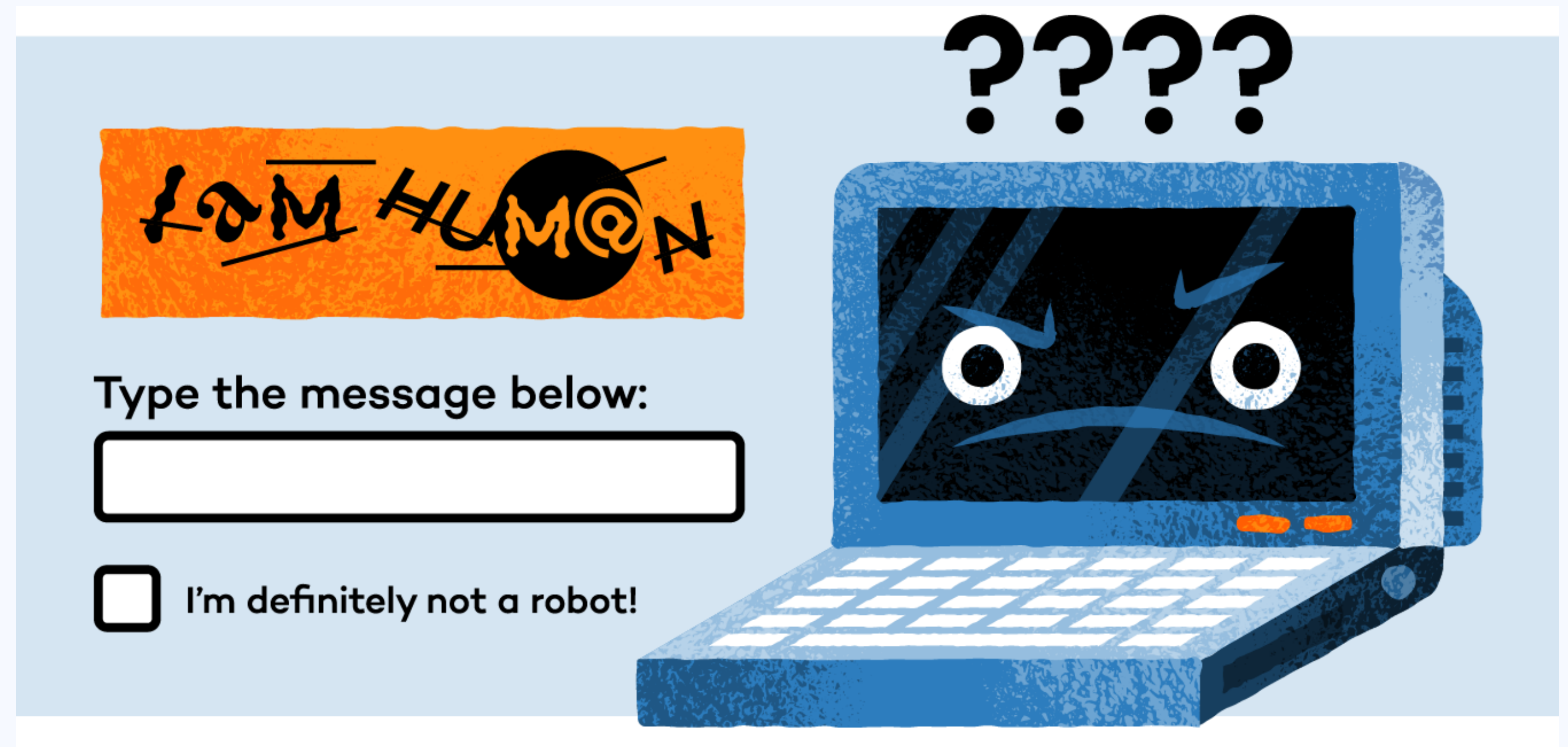


CAPTCHA SOLVER

CON PYTHON Y SKLEARN



Dennis Xiloj | 22006829

20 de junio 2022

Introducción



- En términos generales un captcha es una herramienta para prevenir el abuso en aplicaciones web por medio de una pregunta que sólo debería ser posible responder por un ser humano.
- Famosamente inventado por Luis Von Ahn
- Quizá el servicio mas famoso, reCaptcha, fue introducido en 2009 por Google, implementando la versión mas tradicional del mismo: **el captcha a base de texto**.
- Google ya ha eliminado el uso de este tipo, en favor de variantes más seguras, sin embargo muchas páginas y apps no se han actualizado por diversos motivos incluyendo preocupaciones en cuanto a privacidad.
- En este proyecto intentaremos **comprobar que es posible saltar** este tipo de protección, desde una perspectiva de ethical hacking y cyber seguridad, utilizando herramientas de relativo fácil acceso como el lenguaje python y la librería de machine learning scikit-learn

Objetivo

Crear un modelo predictivo para resolver captchas

Usando técnicas de ciencia de datos y machine learning para entrenar un algoritmo que sea capaz de resolver captchas por sí mismo, para ello:

- Identificar una librería de generación de captchas como objetivo y trabajar sobre la misma para generar un dataset de entrenamiento.
- Tratar las muestras para su uso en el entrenamiento del modelo.
- Elegir el algoritmo mas adecuado, entrenarlo y verificar su precisión.



Vectores de ataque

Simple y directo, usando un OCR

Hoy en día existen muchas herramientas para OCR (Reconocimiento Óptico de Caracteres) que se pueden aplicar

Pro: Muy facil aplicacion, ya existen mutiples soluciones

Contra: Poco efectivo ya que los captchas están específicamente diseñados para contrarrestar este tipo de ataque

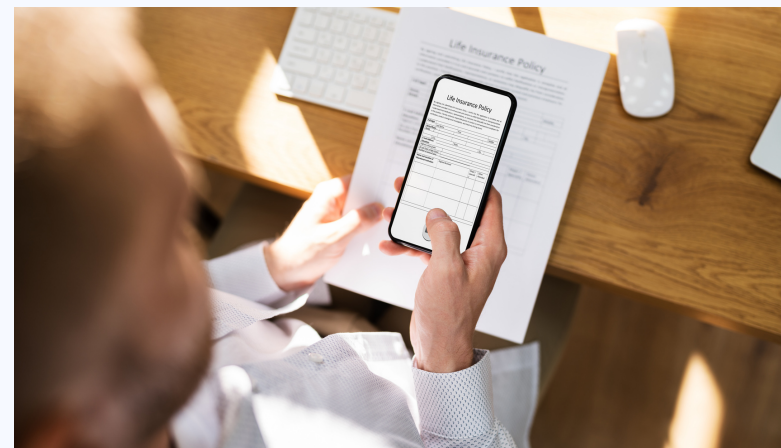


Mediante un servicio de resolucion manual

Existen soluciones que pueden resolver hasta los captchas mas complejos, ya que en realidad son humanos quienes resuelven los captchas

Pro: Efectividad muy alta, ya que lo resuelve un humano.

Contra: Suelen ser dificiles de implementar de forma adecuada por limitaciones tecnológicas y de horarios de trabajo/velocidad de respuesta.

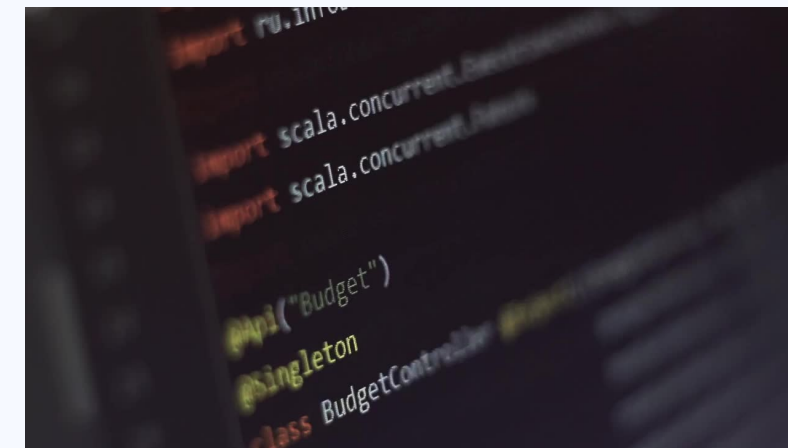


Mediante un modelo de ML entrenado específicamente

Esta solución es la que validaremos en este proyecto

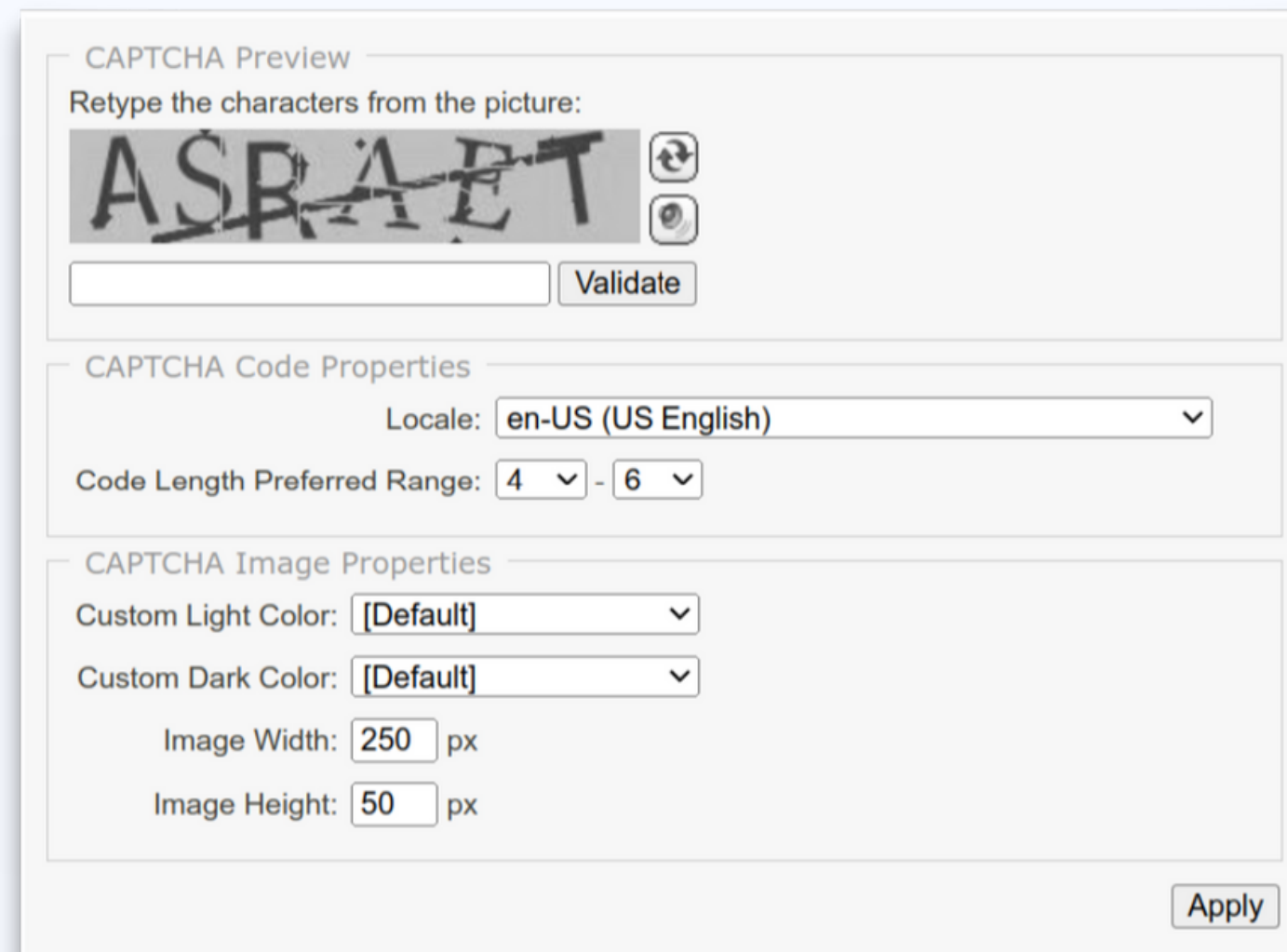
Pro: Ejecución muy rápida y de muy alto rendimiento

Contra: Puede ser difícil obtener las muestras adecuadas para entrenar de forma adecuada al algoritmo



Captcha a validar

(A.K.A. Nuestra víctima)



The image shows a web-based configuration interface for BotDetect CAPTCHA. It is divided into three main sections: CAPTCHA Preview, CAPTCHA Code Properties, and CAPTCHA Image Properties. The CAPTCHA Preview section shows a sample CAPTCHA image with the text 'ASRAET' and a 'Validate' button. The CAPTCHA Code Properties section includes a 'Locale' dropdown set to 'en-US (US English)' and a 'Code Length Preferred Range' set to '4 - 6'. The CAPTCHA Image Properties section includes 'Custom Light Color' and 'Custom Dark Color' dropdowns, both set to 'Default', and 'Image Width' (250 px) and 'Image Height' (50 px) input fields. An 'Apply' button is located at the bottom right.

CAPTCHA Preview

Retype the characters from the picture:

ASRAET

Validate

CAPTCHA Code Properties

Locale: en-US (US English)

Code Length Preferred Range: 4 - 6

CAPTCHA Image Properties

Custom Light Color: [Default]

Custom Dark Color: [Default]

Image Width: 250 px

Image Height: 50 px

Apply

BotDetect CAPTCHA

- Existe desde 2004, y (segun su página) es usado en sitios como el gobierno de USA y Western Union.
- Tiene multiples variantes para lenguajes como Java y PHP
- Es de pago, USD \$400 al año, pero se puede usar gratuitamente con funciones limitadas.
- Puede generar mas de 50 tipos de captchas
- Elegida debido a su amplia variedad de captchas y configuraciones disponibles
- Pero no es la única, existen muchas mas librerías, muchas de ellas gratuitas y open source.

El dataset

AACKR

CABTY

CLAOU

DFIAZ

DFSXA

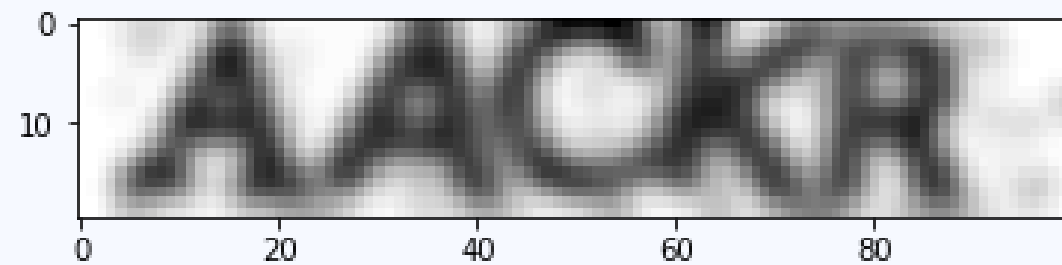
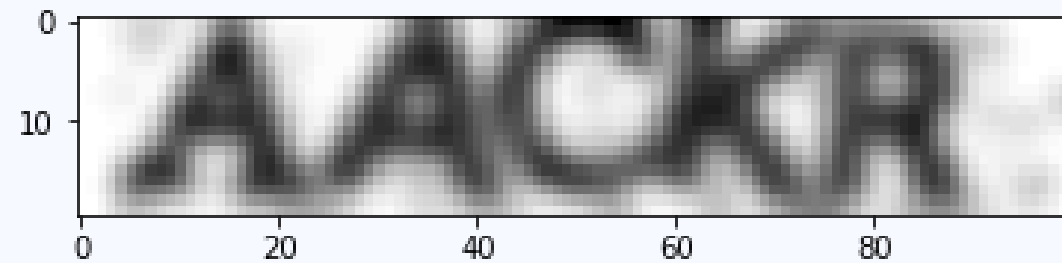
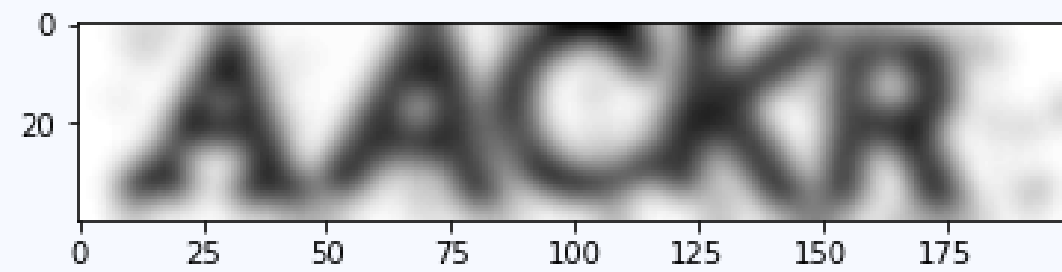
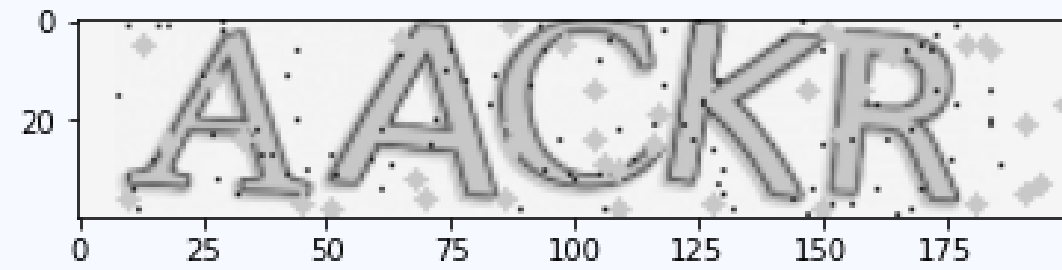
Generación

El dataset fue generado utilizando la misma librería insertándole código para exportar los captchas a disco duro y nombrandolos de forma adecuada, en formato EstiloCaptcha--Texto.png, por ejemplo:

- AncientMosaic--DFSXA.png

Se generaron 5,182 muestras de 5 letras del mismo tipo de captcha, mediante una función que permite tener 200 muestras de cada letra, 40 en cada posición

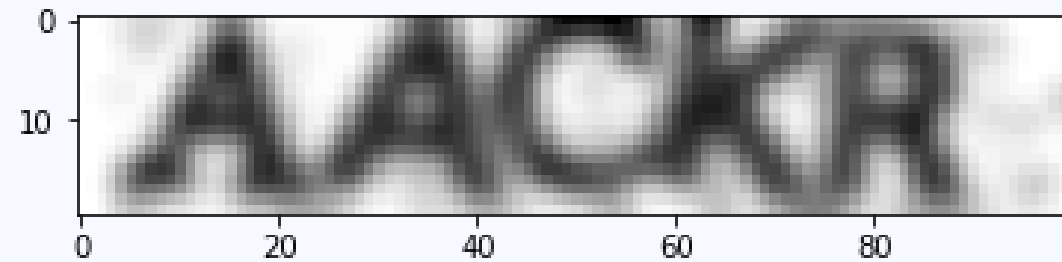
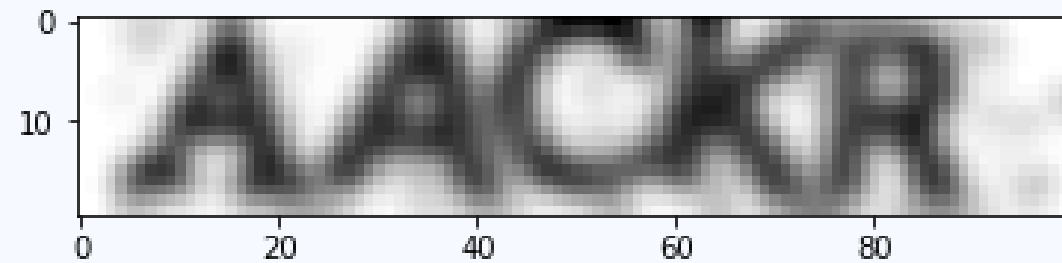
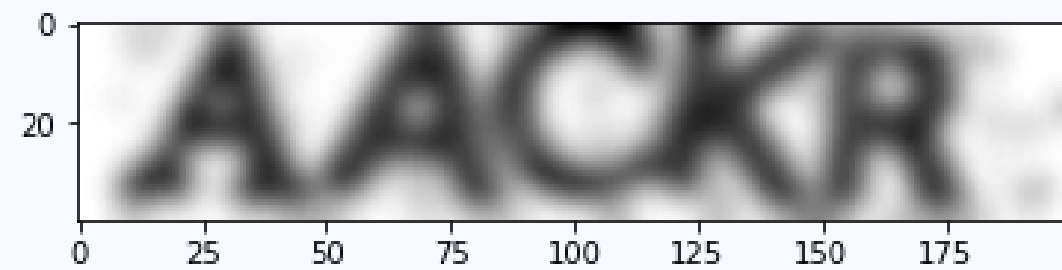
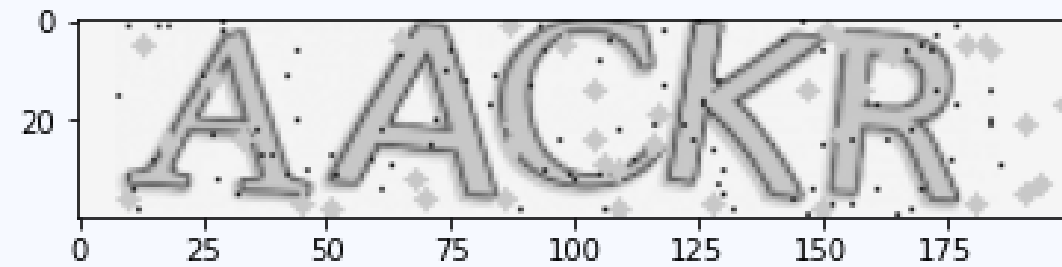
El dataset



Preparación y limpieza

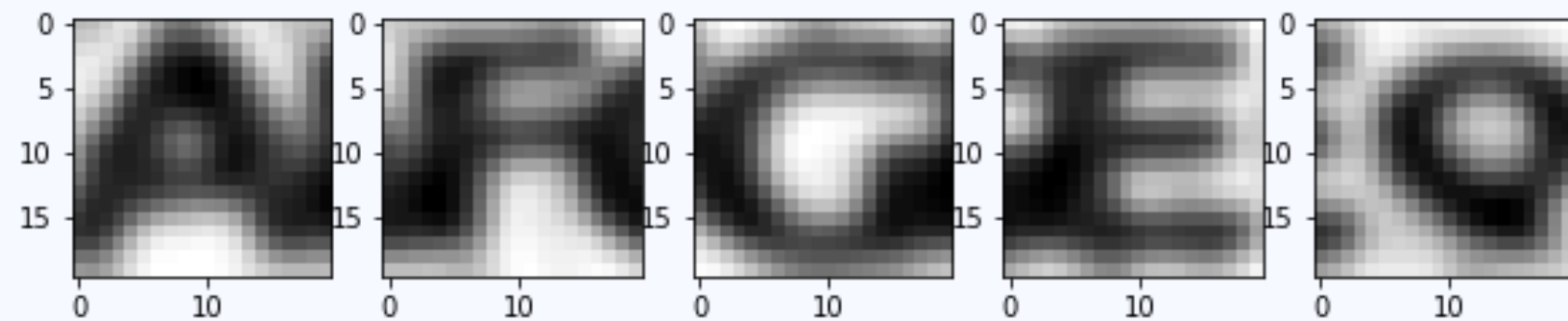
1. Carga de la imagen, originalmente a 200x40px
2. Cambio a Grayscale (para este captcha no se nota, pero pasamos la imagen de 3 canales a 1)
3. Blur gaussiano a 2 ds
4. Escalado a 200x20px
5. Centrado a valores -1 a 1
6. Corte de letras individuales

El dataset



Recorte de letras

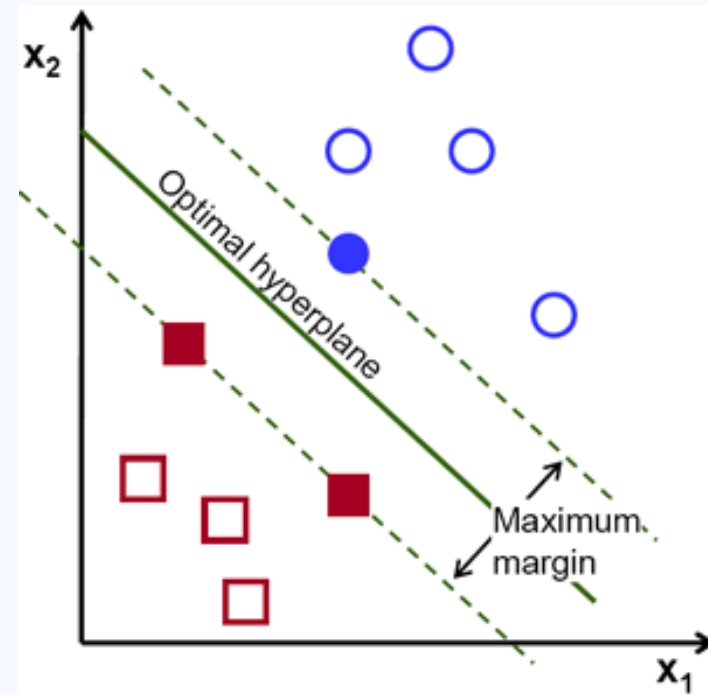
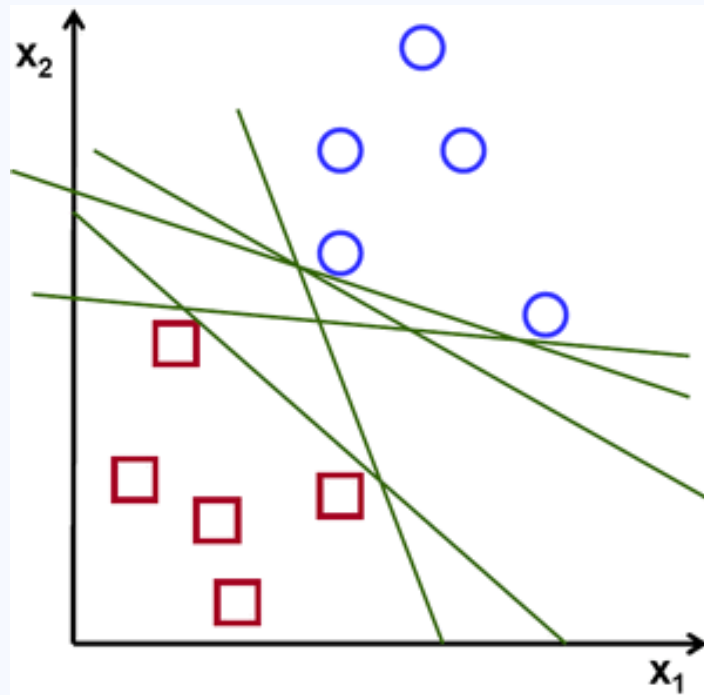
1. El algoritmo será entrenado para reconocer cada letra por independiente, por lo que debemos recortar las muestras de cada letra
2. Eliminamos 12px de cada lado y partimos el resto en partes iguales +un pequeño margen, lo que nos permite tener muestras de letras que ligeramente se superponen o se mueven (como las letras AA en el ejemplo a la izquierda)
3. Nuestra X será un array numpy de todas las letras extraídas



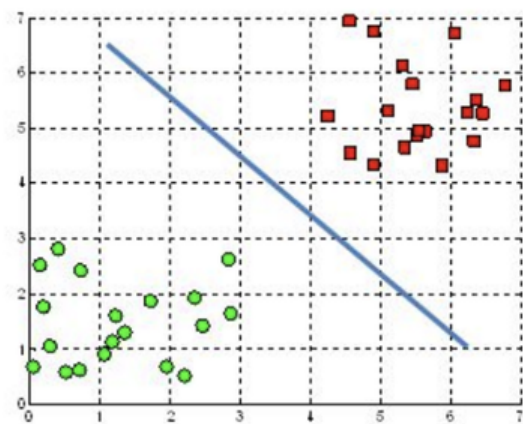
Modelo de ML

Support Vector Classification

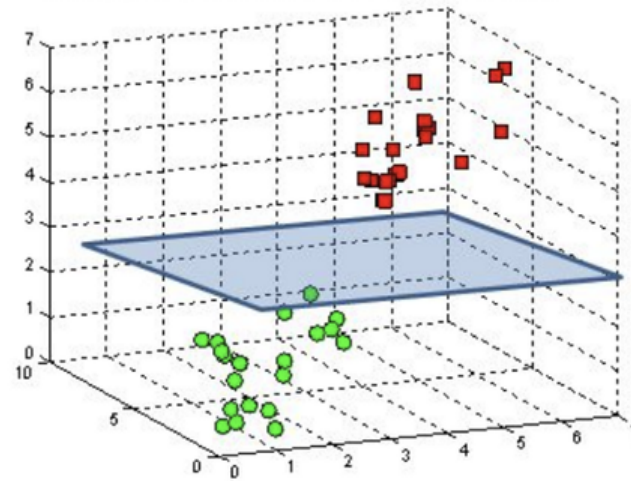
Es un algoritmo de clasificación que busca el plano óptimo por el que cortar las clases, en este caso no busca el plano entre los centros de las clases sin el plano optimo entre los vectores, que son los outliers de cada clase.



A hyperplane in \mathbb{R}^2 is a line



A hyperplane in \mathbb{R}^3 is a plane



- Este algoritmo soporta muy bien datasets con muchas variables, en nuestro caso son elementos de $20 \times 20 = 400$ variables, y especialmente es adecuado en escenarios donde hay mas clases que variables
- Normalmente usado en reconocimiento de imagenes

Entrenamiento



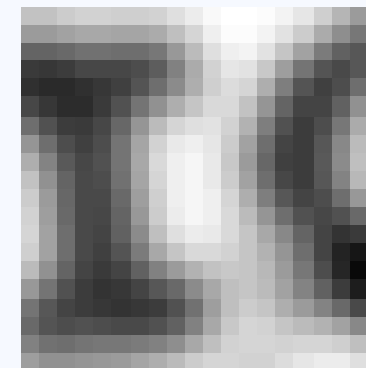
```
1 # clasificador, entrenaod mediante OneVsRestClassifier, lo que
2 # nos permite trabajar sobre los datos de entrenamiento en paralelo
3 # de toda la base de datos.
4 clf = OneVsRestClassifier(svm.SVC(gamma=0.01), n_jobs=12)
5
6 # Learn the digits on the train subset
7 clf.fit(X_train, y_train)
8
9 # Predict the value of the digit on the test subset
10 predicted = clf.predict(X_test)
11
12 # plot predictions
13 _, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
14 for ax, image, prediction in zip(axes, X_test, predicted):
15     ax.set_axis_off()
16     image = image.reshape(20, int(len(image)/20))
17     ax.imshow(image, cmap=plt.cm.gray)
18     ax.set_title(f"Prediction: {prediction}")
```

Accuracy avg:

0.87

Ejemplo:

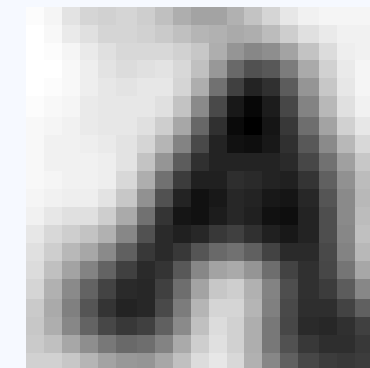
Prediction: I



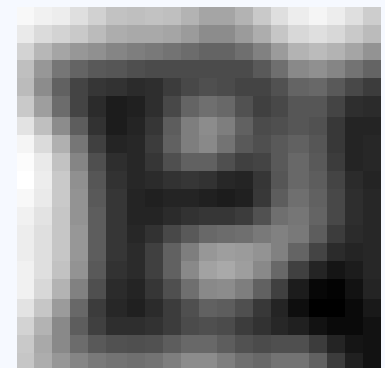
Prediction: N



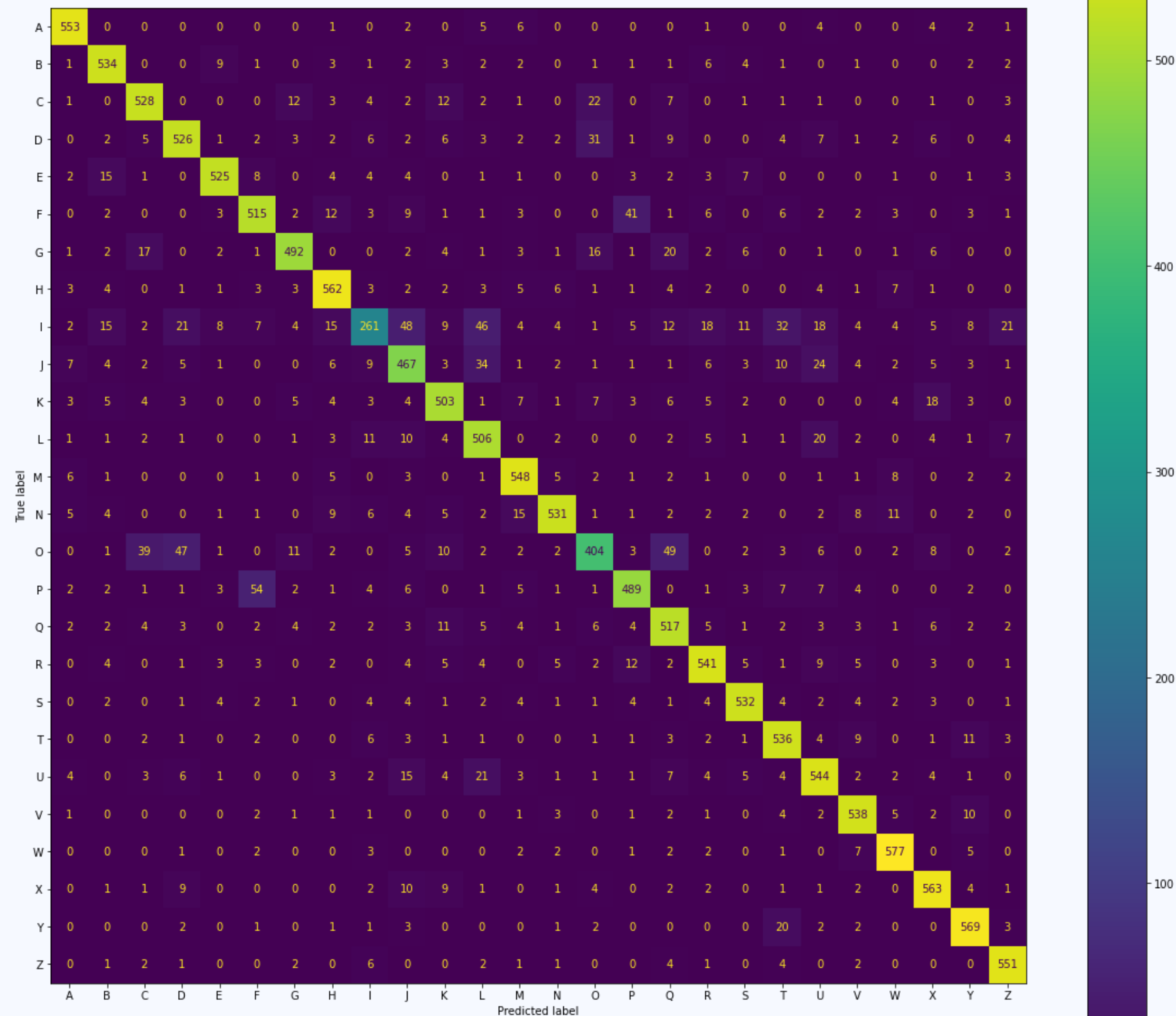
Prediction: A



Prediction: E



Matriz de confusion



Precisión

0.76

Menor

0.90

Mayor

Promedio:

0.86

Errores mas problematicos:

I, J, L

O,C,D,Q

Q,P

DEMO!

Conclusiones

Seguridad

Los captchas de texto no son seguros, por lo menos no debemos confiar en ellos para aplicaciones de seguridad crítica



Implementación del Algoritmo

Se puede generar un algoritmo que sea capaz de identificar un captcha, en este ejemplo al menos el 50% de las veces, lo que puede burlar fácilmente estos sistemas



Posibles mejoras

Se puede mejorar mas el algoritmo limpiando mejor los datos, usando mas muestras y de mas tipos de captchas

