



# Prueba Técnica Departamento de Datos no Estructurados

<p style=»text-align: justify;»> El Departamento de Analítica No estructurada busca profesionales con fuertes capacidades técnicas y sobretodo una fuerte capacidad analítica. Por consiguiente esta prueba intenta poner a prueba la forma en la que plantea y soluciona problmeas. Es importante que presente el código que usa para resolver el problema con el único motivo de medir sus capacidades.

Se recomienda que sea ordenado en su código y siga los lineamientos establecidos, aún así la prueba tiene un grado de flexibilidad. Se evaluará el orden y la creatividad a la hora de presentar la información. </p>

## Nota

Tenga en cuenta que éste ejercicio es hipotético y el banco no usará su trabajo más que para evaluar sus habilidades para el cargo

## Objetivo

El objetivo de esta prueba es lograr un filtro que discrimine automáticamente un tipo de documento sin información relevante: páginas en blanco. Se busca que este filtro reciba como entrada una carpeta con imágenes de documentos diversos y produzca como salida dos carpetas, una con imágenes de páginas en blanco y otra con imágenes de páginas con contenido.

Páginas con solo el membrete del documento se consideran páginas en blanco, así como las que, al momento de ser escaneadas, alcanzan a reflejar contenido ininteligible del reverso de la página.

# Clasificación de Imágenes

## 0.Librerías

```
In [37]: !pip install -r requirements.txt
```

```
In [10]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os

from numpy import mean
from numpy import std
from scipy.stats import truncnorm

from PIL import Image
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split

import glob

import matplotlib.image as mpimg
```

# 1.Importe las imágenes

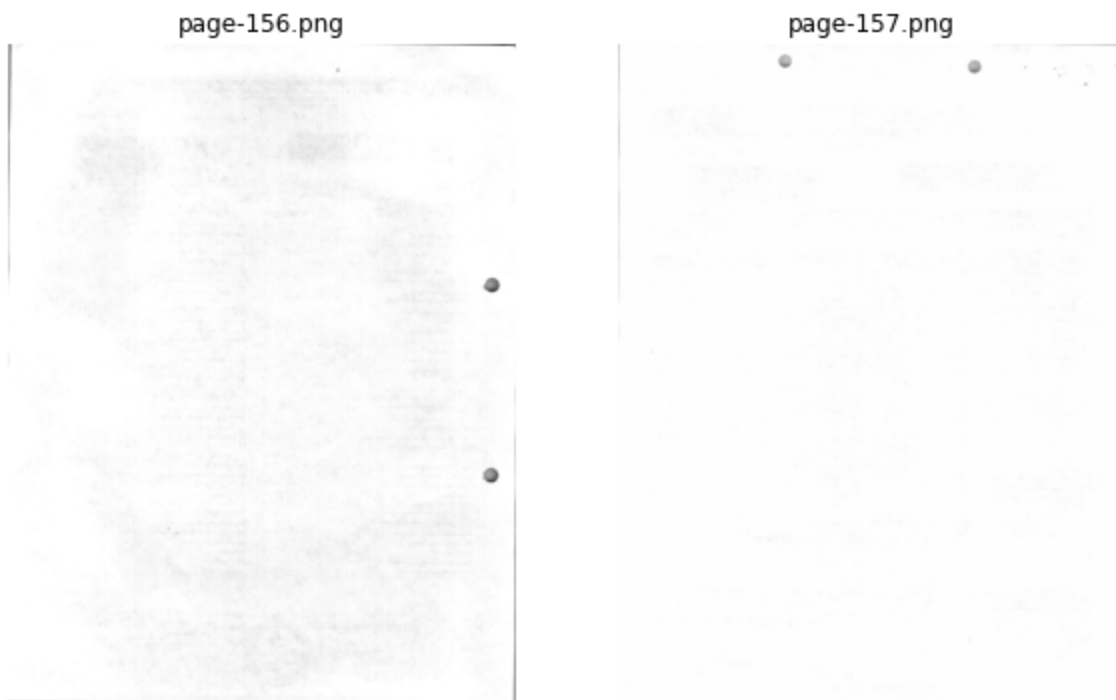
En la carpeta adjunta se encontrará con documentos tanto en blanco como con contenido. Su objetivo es generar un modelo que clasifique los elementos en "Con Contenido" y "Sin Contenido". Para eso puede utilizar reglas o modelos específicos.

```
In [12]: def mostrar_imagenes(carpeta, num_imagenes=2):
    imagenes = [f for f in os.listdir(carpeta) if f.endswith('.png')]
    imagenes = imagenes[:num_imagenes]

    fig, axes = plt.subplots(1, num_imagenes, figsize=(10, 10))
    for ax, img in zip(axes, imagenes):
        img_path = os.path.join(carpeta, img)
        img_data = mpimg.imread(img_path)
        ax.imshow(img_data, cmap='gray')
        ax.axis('off')
        ax.set_title(img)
    plt.show()

# Mostrar imágenes de la carpeta Blanco
mostrar_imagenes('Imagenes/Blanco')

# Mostrar imágenes de la carpeta Documentos
mostrar_imagenes('Imagenes/Documentos')
```



## 1. Introduction

Computation has become a central tool in economics. From the solution of dynamic equilibrium models in macroeconomics or industrial organization, to the characterization of equilibria in game theory, or in estimation by simulation, economists spend a considerable amount of their time coding and running fairly sophisticated software.

And while some effort has been focused on the comparison of different algorithms for the solution of common problems in economics (see, for instance, Arsenau, Fortin-Villeneuve, and Robitaille, 2006), there has been little formal comparison of programming languages. This is surprising because there is an ever-growing variety of programming languages and economists are often puzzled about which language is best suited to their needs.<sup>1</sup> Instead of a suite of benchmarks, researchers must rely on personal experiments or on “folk wisdom.” For example, it is still commonly believed that Fortran is the fastest available language or that C++ is too hard to learn given its potential advantages.

In this paper, we take a first step at correcting this unfortunate situation. The target audience for our results is younger economists (graduate students, junior faculty) or researchers who have used the computer less often in the past for numerical analysis and who are searching for guidelines in their first incursions into computation.

We solve the stochastic neoclassical growth model, the workhorse of modern macroeconomics, using C++11, Fortran 2008, Java, Julia, Python, Matlab, Mathematica, and R.<sup>2</sup> We implement the same algorithm, value function iteration with grid search for optimal future capital, in each of the languages<sup>3</sup> and measure the execution time of the codes in a Mac and in a Windows computer. The advantage of our algorithm, value function iteration with grid search, is that it is “representative” of many economic computations: expensive loops, large matrices to store in memory, and so on. Thus, while our investigation does not entail a full suite of benchmarks, both our model and our solution method are among the best available choices for our investigation. In addition, we use two machines, a Mac and a Windows computer, as the two most popular environments for software development for economists (in the Mac we compile and run some of the code from the command line, thus implying results very close to those that would come from equivalent Unix/Linux machines).

In section 4, we report speed results for each language (including several implementations

<sup>1</sup>This also stands in comparison with work in other fields, such as in Pechen (2008) or Lichten and Shum (2011), or web projects such as The Computer Language Benchmarks Game (see <http://benchmarksgame.alioth.debian.org/>).

<sup>2</sup>From now on, we drop the year of the standard of the language unless it is needed.

<sup>3</sup>The Python and Mathematica versions have two variants of the code: one with exactly the same algorithm as the other languages and one with more variations. These two variants will illustrate the importance of algorithmic programming. The change in Python is small, but larger in Mathematica.

at least one literal is set to TRUE by the assignment  $\phi$ . The clauses can be included with equal probability in analogy to the  $I_{\text{LIT}}$  or  $I_{\text{LIT}}$  distributions above [26, 26], or different probabilities can be assigned to the clauses with one, two, or three literals set to TRUE by  $\phi$ , in an effort to better hide the satisfying assignment [3, 31].

A particularly appealing modification of the planted random distribution is to the semirandom instance, where a planted random instance is perturbed by an adversary who is constrained to add only clauses which are consistent with the planted satisfying assignment and not to delete anything [36].

## Problem 1 (3SAT).

INPUT: 3-CNF Boolean formula  $F = C_1 \vee C_2 \vee \dots \vee C_m$ , where each clause  $C_i$  is of the form  $C_i = \ell_i \vee \ell_i' \vee \ell_i''$ , and each literal  $\ell_i$  is either a variable or the negation of a variable.  
OUTPUT: A truth assignment of variables to Boolean values which makes at least one literal in each clause TRUE, or a certificate that no such assignment exists.

## 2 KEY RESULTS

It is convenient to consider three categories of heuristics for 3SAT, based on the type of input distribution for which the heuristic is likely to be successful.

**Algorithms for sparse satisfiable instances.** In the analysis of heuristics, it is often difficult to cope with the conditioning introduced by backtracking algorithms. However, there are some simple heuristics which make no use of backtracking that are likely to succeed on sparse random instances, that is, instances of 3SAT where the ratio of clauses to variables is sufficiently small.

The *First Literal Heuristic* (FL) is one approach which has no backtracking. It functions as follows: Repeat the following: if the formula contains a literal and not the negation of that literal, set this literal to TRUE and remove all clauses containing the literal. Otherwise set a randomly chosen literal to TRUE, remove all the clauses it satisfies, and remove its negation from all the clauses in which it appears. If a clause becomes empty, halt and declare failure. Otherwise, when no clauses are left, return the satisfying assignment generated. The FL heuristic has been analyzed by many researchers, and the following result appears in [32] and is proved rigorously in [26, 32].

**Theorem 1.** There exists a constant  $c_{FL} \approx 1.887$ , such that for any constant  $\epsilon < c_{FL}$ , **whp** the FL heuristic succeeds on  $I_{\text{LIT}}$ , while for any constant  $\epsilon > c_{FL}$ , **whp** the FL heuristic fails on  $I_{\text{LIT}}$ .

The *Generalized First Clause Heuristic* (GFC) is an alternative approach which does not require backtracking, and works in the following manner: Repeat the following: choose a clause  $C$  uniformly at random from all clauses of shortest length, and choose a literal  $\ell$  uniformly at random from  $C$ . Set the variable corresponding to  $\ell$  so that  $C$  is satisfied, remove all clauses in the instance in which  $\ell$  appears, and remove  $\ell$  from all the clauses in which it appears. If a clause becomes empty, halt and declare failure. Otherwise, when no clauses are left, return the satisfying assignment generated. The GFC heuristic has been analyzed by many researchers, eventually leading to the following result from [34].

**Theorem 2.** There exists a constant  $c_{GFC} \approx 2.003$  such that for any  $\epsilon < c_{GFC}$ , the GFC heuristic succeeds on  $I_{\text{LIT}}$  with asymptotically positive probability, while for any  $\epsilon \geq c_{GFC}$ , the GFC heuristic fails **whp**.

Many algorithms are a class of heuristics which are defined formally in [1]. They generalize this backtracking-free approach. These algorithms and their close relatives in [26, 27] provide a popular approach to rigorously proving lower bounds on the satisfiability threshold (which is currently known to be at least 3.32). They are treated in detail in Chapter 77 of this volume.

```
In [13]: # Definir la lista de rutas de imágenes
imgList = [
    'Imagenes/Documentos/page-2.png'
]

# Función para redimensionar la imagen manteniendo la relación de aspecto
def resizeAspect(im, size):
    w, h = im.size
    aspect = max(size[0] / float(w), size[1] / float(h))
    return im.resize((int(w * aspect), int(h * aspect)), Image.LANCZOS)

# Función para centrar la imagen en un lienzo de tamaño fijo
def centerImage(im, size):
    w, h = im.size
    new_im = Image.new("RGB", size, (255, 255, 255)) # Crear un lienzo blanco
    new_im.paste(im, ((size[0] - w) // 2, (size[1] - h) // 2))
    return new_im

# Procesar solo una imagen de ejemplo
img_path = imgList[0]
img = Image.open(img_path)

# Imprimir las dimensiones de la imagen antes de preprocesar
print(f"Dimensiones de la imagen antes de preprocesar: {img.size}")

# Mostrar la imagen antes del preprocesamiento
plt.figure(figsize=(10, 10))
plt.imshow(img, cmap='gray')
plt.title('Antes del preprocesamiento')
plt.axis('off')
plt.show()

# Preprocesar la imagen
w, h = img.size
if min(w, h) < 1500:
    img = resizeAspect(img, (448, 448))
    w, h = img.size
center = [int(w / 2), int(h / 2)]
box = (center[0] - 224, center[1] - 224, center[0] + 224, center[1] + 224)
cropped_img = img.crop(box)
```

```
# Imprimir las dimensiones de la imagen después de preprocesar
print(f"Dimensiones de la imagen después de preprocesar: {cropped_img.size}")

# Mostrar la imagen después del preprocesamiento
plt.figure(figsize=(10, 10))
plt.imshow(cropped_img, cmap='gray')
plt.title('Después del preprocesamiento')
plt.axis('off')
plt.show()

# Convertir la imagen a un array y aplanarla
img_array = np.array(cropped_img).flatten()

# Determinar la etiqueta basada en el nombre del archivo
etiqueta = 0 if 'Blanco' in img_path else 1

# Imprimir la etiqueta de la imagen
print(f"Etiqueta de la imagen: {etiqueta}")
```

Dimensiones de la imagen antes de preprocesar: (1224, 1584)

### Antes del preprocesamiento

Our study is novel in that it looks at student mistakes from a much larger number of students (over 250,000) from a large number of institutions<sup>1</sup>, thus providing more robust data about error frequencies, and error commonality. The Blackbox work was originally presented with a brief list of the most frequent compiler error messages [3], but in this study we do not simply use compiler error messages to classify errors. Instead, we borrow error classifications from Hristova et al. [6], which are based on surveying educators to ask for the most common Java mistakes they saw among their students.

## 3. METHOD

### 3.1 Student Mistakes

We use the 18 mistakes from our previous study [2] as a basis for our analysis, which in turn were derived from Hristova et al.'s [6] twenty student mistakes (derived from interviewing educators). The eighteen mistakes, labeled A through R, are informally categorized as follows:

**Misunderstanding (or forgetting) syntax:**

- **A:** Confusing the assignment operator (=) with the comparison operator (==).  
For example: `if (a = b) ...`
- **C:** Unbalanced parentheses, curly or square brackets and quotation marks, or using these different symbols interchangeably.  
For example: `while (a == 0)`
- **D:** Confusing "short-circuit" evaluators (&& and ||) with conventional logical operators (& and |).  
For example: `if ((a == 0) & (b == 0)) ...`
- **E:** Incorrect semi-colon after an if selection structure before the if statement or after the for or while repetition structure before the respective for or while loop.  
For example:  
`if (a == b);  
 return 6;`
- **F:** Wrong separators in for loops (using commas instead of semi-colons).  
For example: `for (int i = 0, i < 6, i++) ...`
- **G:** Inserting the condition of an if statement within curly brackets instead of parentheses.  
For example: `if {a == b} ...`
- **H:** Using keywords as method or variable names.  
For example: `int new;`
- **J:** Forgetting parentheses after a method call.  
For example: `myObject.toString;`
- **K:** Incorrect semicolon at the end of a method header.  
For example:  
`public void foo();  
{  
 ...  
}`

<sup>1</sup>We have no way of measuring the number of institutions in the Blackbox data, but simply: the 250,000 students must be split over at least several hundred institutions.

- **L:** Getting greater than or equal/less than or equal wrong, i.e. using >= or <= instead of > and <.  
For example: `if (a >= b) ...`
- **P:** Including the types of parameters when invoking a method.  
For example: `myObject.foo(int x, String s);`

**Type errors:**

- **I:** Invoking methods with wrong arguments (e.g. wrong types).  
For example: `list.get("abc")`
- **Q:** Incompatible types between method return and type of variable that the value is assigned to.  
For example: `int x = myObject.toString();`

**Other semantic errors:**

- **B:** Use of == instead of .equals to compare strings.  
For example: `if (a == "start") ...`
- **M:** Trying to invoke a non-static method as if it was static.  
For example: `MyClass.toString();`
- **N:** A method that has a non-void return type is called and its return value ignored/discarded.  
For example: `myObject.toString();`
- **O:** Control flow can reach end of non-void method without returning.  
For example:  
`public int foo(int x)  
{  
 if (x < 0)  
 return 0;  
 x == 1;  
}`
- **R:** Class claims to implement an interface, but does not implement all the required methods.  
For example: `class Y implements ActionListener  
{`

Note that mistake *N* (ignoring the non-void result of a method) is not always an error (e.g. when you call a remove method that returns the item removed, you may not need to do anything with the return value).

### 3.2 Student data

Data about student mistakes was taken from the Blackbox data set [3], which collects Java code written by users of BlueJ, the Java beginners' IDE. We used data from the period 1<sup>st</sup> Sep. 2013 to 31<sup>st</sup> Aug. 2014 (inclusive), as representing a full year.

We had two methods of detecting mistakes. For four of the student mistakes, *I*, *M*, *O*, *R*, we were able to use the compiler error message directly from Blackbox's compilations to detect the mistake. However, this was not possible for the other errors, as some of them are logical errors that do not cause a compiler error or warning, while in other cases the error messages do not have a one-to-one mapping to our mistakes of interest. Thus for one of the other mistakes (*C*) we performed a post-hoc analysis (matching brackets) and for the final thirteen we used a customized permissive parser

Dimensiones de la imagen después de preprocesar: (448, 448)

data about error frequencies, and error commonality. The Blackbox work was originally presented with a brief list of the most frequent compiler error messages [3], but in this study we do not simply use compiler error messages to classify errors. Instead, we borrow error classifications from Hristova et al. [6], which are based on surveying educators to ask for the most common Java mistakes they saw among their students.

### 3. METHOD

#### 3.1 Student Mistakes

We use the 18 mistakes from our previous study [2] as a basis for our analysis, which in turn were derived from Hristova et al.'s [6] twenty student mistakes (derived from interviewing educators). The eighteen mistakes, labeled A through R, are informally categorized as follows:

**Misunderstanding (or forgetting) syntax:**

- **A:** Confusing the assignment operator (=) with the comparison operator (==).  
For example: `if (a = b) ...`
- **C:** Unbalanced parentheses, curly or square brackets and quotation marks, or using these different symbols interchangeably.  
For example: `while (a == 0)`
- **D:** Confusing "short-circuit" evaluators (&& and ||) with conventional logical operators (& and |).  
For example: `if ((a == 0) & (b == 0)) ...`
- **E:** Incorrect semi-colon after an if selection structure before the if statement or after the for or while repetition structure before the respective for or while loop.  
For example:  
`if (a == b);  
return 6;`
- **F:** Wrong separators in for loops (using commas instead of semi-colons).  
For example: `for (int i = 0, i < 6, i++) ...`
- **G:** Inserting the condition of an if statement within curly brackets instead of parentheses.  
For example: `if {a == b} ...`
- **H:** Using keywords as method or variable names.  
For example: `int new;`
- **J:** Forgetting parentheses after a method call.  
For example: `myObject.toString;`
- **K:** Incorrect semicolon at the end of a method header.  
For example:  
`public void foo();  
{  
 ...  
}`

<sup>1</sup>We have no way of measuring the number of institutions in

- **P:** Including the types of parameters when invoking a method.  
For example: `myObject.foo(int x, String s);`

**Type errors:**

- **I:** Invoking methods with wrong arguments (e.g. wrong types).  
For example: `list.get("abc")`
- **Q:** Incompatible types between method return and type of variable that the value is assigned to.  
For example: `int x = myObject.toString();`

**Other semantic errors:**

- **B:** Use of == instead of .equals to compare strings.  
For example: `if (a == "start") ...`
- **M:** Trying to invoke a non-static method as if it was static.  
For example: `MyClass.toString();`
- **N:** A method that has a non-void return type is called and its return value ignored/discarded.  
For example: `myObject.toString();`
- **O:** Control flow can reach end of non-void method without returning.  
For example:  
`public int foo(int x)  
{  
 if (x < 0)  
 return 0;  
 x == 1;  
}`
- **R:** Class claims to implement an interface, but does not implement all the required methods.  
For example: `class Y implements ActionListener  
{`

Note that mistake *N* (ignoring the non-void result of a method) is not always an error (e.g. when you call a remove method that returns the item removed, you may not need to do anything with the return value).

#### 3.2 Student data

Data about student mistakes was taken from the Blackbox data set [3], which collects Java code written by users of BlueJ, the Java beginners' IDE. We used data from the period 1<sup>st</sup> Sep. 2013 to 31<sup>st</sup> Aug. 2014 (inclusive), as representing a full year.

We had two methods of detecting mistakes. For four of the student mistakes, *I*, *M*, *O*, *R*, we were able to use the compiler error message directly from Blackbox's compilations to detect the mistake. However, this was not possible for the other errors, as some of them are logical errors that do not cause a compiler error or warning, while in other cases the error messages do not have a one-to-one mapping to our mistakes of interest. Thus for one of the other mistakes (*C*)

Etiqueta de la imagen: 1

Como en este ejercicio no nos interesa analizar el contenido de los documentos, realizamos un proceso de re escalamiento de las imagenes para que sean más ligeras y fáciles de interpretar para el modelo.

## 2. Estructure la información

Debido a que las imagenes son archivos separados lo primero que debe hacer es importarlas (recuerde que una imagen es esencialmente un arreglo de vectores), puede esturcturarlas a su gusto y marcar de ser necesario aquellas que va usar como test de pruebas.

```
In [38]: import os
import random
from PIL import Image
import numpy as np

# Función para redimensionar la imagen manteniendo la relación de aspecto
def resizeAspect(im, size):
    w, h = im.size
    aspect = max(size[0] / float(w), size[1] / float(h))
    return im.resize((int(w * aspect), int(h * aspect)), Image.LANCZOS)

# Función para centrar la imagen en un lienzo de tamaño fijo
def centerImage(im, size):
    w, h = im.size
    new_im = Image.new("RGB", size, (255, 255, 255)) # Crear un lienzo blanco
```

```

new_im.paste(im, ((size[0] - w) // 2, (size[1] - h) // 2))
return new_im

run_preprocess = False

if run_preprocess == False:

    # Definir las carpetas de origen
    carpetas_origen = ['Imagenes/cropped/Blanco', 'Imagenes/cropped/Documentos']
    carpeta_destino = 'Imagenes/validar_final'

    # Crear las carpetas de destino si no existen
    os.makedirs(os.path.join(carpeta_destino, 'Blanco'), exist_ok=True)
    os.makedirs(os.path.join(carpeta_destino, 'Documentos'), exist_ok=True)

    # Seleccionar 10 imágenes al azar de cada carpeta para la validación final
    for carpeta in carpetas_origen:
        imagenes = [img for img in os.listdir(carpeta) if img.endswith('.png')]
        imagenes_validar = random.sample(imagenes, 10)

        for img_name in imagenes_validar:
            img_path = os.path.join(carpeta, img_name)
            if 'Blanco' in carpeta:
                destino_path = os.path.join(carpeta_destino, 'Blanco', img_name)
            elif 'Documentos' in carpeta:
                destino_path = os.path.join(carpeta_destino, 'Documentos', img_name)
            shutil.copy(img_path, destino_path)
        # print(f"Copiada: {img_path} a {destino_path}")

if run_preprocess:

    # Definir las carpetas de origen
    carpetas_origen = [
        'Imagenes/Documentos',
        'Imagenes/Blanco'
    ]
    carpeta_destino = 'Imagenes/cropped'
    carpeta_validar_final = 'Imagenes/validar_final'

    # Crear las subcarpetas de destino si no existen
    os.makedirs(os.path.join(carpeta_destino, 'Blanco'), exist_ok=True)
    os.makedirs(os.path.join(carpeta_destino, 'Documentos'), exist_ok=True)
    os.makedirs(os.path.join(carpeta_validar_final, 'Blanco'), exist_ok=True)
    os.makedirs(os.path.join(carpeta_validar_final, 'Documentos'), exist_ok=True)

    contador_blanco = 0
    contador_documentos = 0

    # Seleccionar 10 imágenes al azar de cada carpeta para la validación final
    imagenes_validar = {}
    for carpeta in carpetas_origen:
        imagenes = [img for img in os.listdir(carpeta) if img.endswith('.png')]
        imagenes_validar[carpeta] = random.sample(imagenes, 10)

    # Procesar todas las imágenes en las carpetas de origen
    for carpeta in carpetas_origen:
        for img_name in os.listdir(carpeta):
            if img_name.endswith('.png'):
                img_path = os.path.join(carpeta, img_name)
                img = Image.open(img_path)

                # Determinar la etiqueta basada en el nombre del archivo
                etiqueta = 'Blanco' if 'Blanco' in carpeta else 'Documentos'

                # Si la imagen está en la lista de validación, moverla a la carpeta de v

```

```

if img_name in imagenes_validar[carpeta]:
    validar_img_path = os.path.join(carpeta_validar_final, etiqueta, img_name)
    img.save(validar_img_path, 'PNG')
    print(f"Imagen {img_name} movida a la carpeta de validación final.")
    continue

print(f"Dimensiones de la imagen antes de preprocesar: {img.size}")
img = resizeAspect(img, (448, 448))
img = centerImage(img, (448, 448))
print(f"Dimensiones de la imagen después de preprocesar: {img.size}")

# Guardar la imagen procesada en la subcarpeta de destino correspondiente
cropped_img_name = f"{os.path.splitext(img_name)[0]}_{etiqueta}.png"
cropped_img_path = os.path.join(carpeta_destino, etiqueta, cropped_img_name)
img.save(cropped_img_path, 'PNG')

if etiqueta == 'Blanco':
    contador_blanco += 1
else:
    contador_documentos += 1

print(f"Etiqueta de la imagen: {etiqueta}")

print(f"Cantidad de imágenes en la carpeta 'Blanco': {contador_blanco}")
print(f"Cantidad de imágenes en la carpeta 'Documentos': {contador_documentos}")

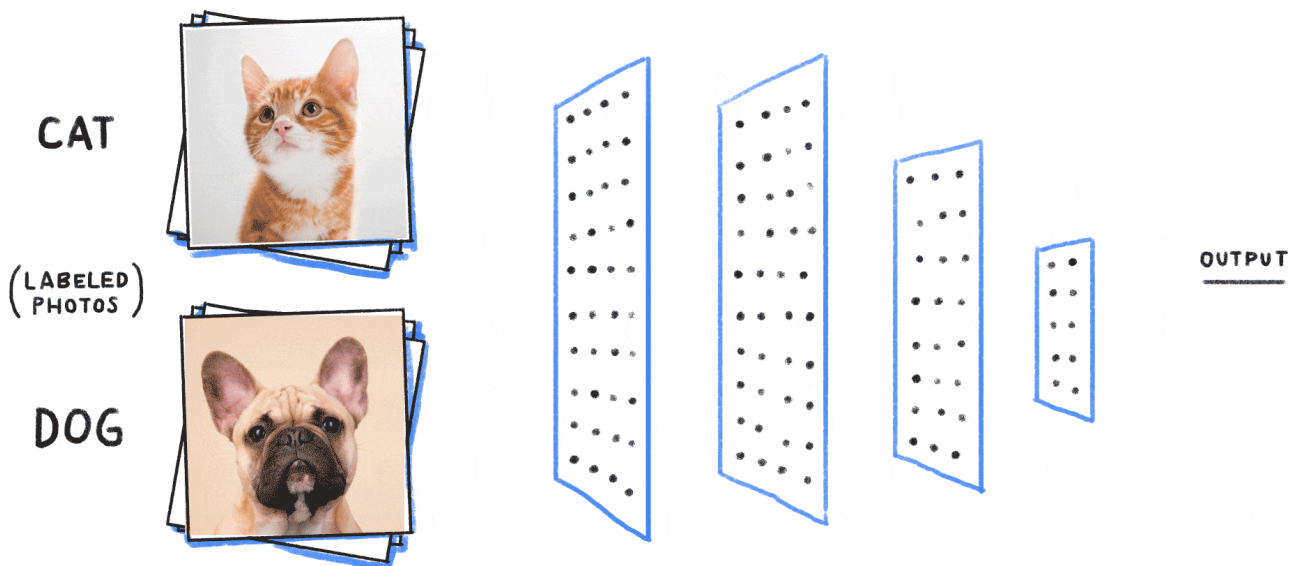
else:
    print("Las imágenes ya se encuentran preprocesadas")

```

Las imágenes ya se encuentran preprocesadas

### 3. Describa su Estrategia de Análisis

Tiene libertad en la metodología para la clasificación. Aún así debe describir brevemente como realizará el análisis. Por ejemplo, si usará un modelo en donde requiera clasificar una cantidad pequeña de la data mencionelo o si planea condicionar la clasifiaciones a reglas indique que reglas usará. Además si usa herramientas externas describalas y explique.





Se emplea un modelo convolucional que clasifica las imágenes basándose su contenido visual.

```
In [32]: from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
import matplotlib.pyplot as plt

# Crear un generador de datos con aumento de datos
datagen = ImageDataGenerator(
    rescale=1.0/255.0,
    validation_split=0.2,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

# Cargar las imágenes de entrenamiento y validación
train_generator = datagen.flow_from_directory(
    'Imágenes/cropped',
    target_size=(224, 224), # Reducir el tamaño de la imagen para acelerar el entrenami
    batch_size=32,
    class_mode='binary',
    subset='training'
)

validation_generator = datagen.flow_from_directory(
    'Imágenes/cropped',
    target_size=(224, 224), # Reducir el tamaño de la imagen para acelerar el entrenami
    batch_size=32,
    class_mode='binary',
    subset='validation'
)

# Contar las imágenes de cada clase
blanco_count = sum(['Blanco' in f for f in train_generator.filesnames])
documentos_count = sum(['Documentos' in f for f in train_generator.filesnames])

print(f"Found {blanco_count} images belonging to 'Blanco' class.")
print(f"Found {documentos_count} images belonging to 'Documentos' class.")

# Definir el modelo con regularización y más capas de Dropout
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

# Compilar el modelo
model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=

# Entrenar el modelo
history = model.fit(
```



```

train_generator,
epochs=25, # Aumentar el número de épocas
validation_data=validation_generator
)

# Guardar el modelo entrenado
model.save('modelo_clasificacion_paginas_mejorado.h5')

```

```

Found 182 images belonging to 2 classes.
Found 45 images belonging to 2 classes.
Found 110 images belonging to 'Blanco' class.
Found 72 images belonging to 'Documentos' class.

```

```
Epoch 1/25
```

```

6/6 _____ 7s 973ms/step - accuracy: 0.5609 - loss: 0.7147 -
val_accuracy: 0.4000 - val_loss: 0.6945

```

```
Epoch 2/25
```

```

6/6 _____ 5s 796ms/step - accuracy: 0.5629 - loss: 0.7211 -
val_accuracy: 0.6000 - val_loss: 0.6747

```

```
Epoch 3/25
```

```

6/6 _____ 5s 782ms/step - accuracy: 0.6345 - loss: 0.6209 -
val_accuracy: 1.0000 - val_loss: 0.5290

```

```
Epoch 4/25
```

```

6/6 _____ 5s 771ms/step - accuracy: 0.8109 - loss: 0.4797 -
val_accuracy: 0.7111 - val_loss: 0.4888

```

```
Epoch 5/25
```

```

6/6 _____ 5s 769ms/step - accuracy: 0.8774 - loss: 0.3628 -
val_accuracy: 0.9111 - val_loss: 0.3154

```

```
Epoch 6/25
```

```

6/6 _____ 5s 778ms/step - accuracy: 0.9592 - loss: 0.2587 -
val_accuracy: 0.9556 - val_loss: 0.2415

```

```
Epoch 7/25
```

```

6/6 _____ 5s 776ms/step - accuracy: 0.9654 - loss: 0.1924 -
val_accuracy: 0.8444 - val_loss: 0.2424

```

```
Epoch 8/25
```

```

6/6 _____ 5s 831ms/step - accuracy: 0.9704 - loss: 0.1646 -
val_accuracy: 0.9556 - val_loss: 0.1449

```

```
Epoch 9/25
```

```

6/6 _____ 5s 767ms/step - accuracy: 0.9531 - loss: 0.1641 -
val_accuracy: 0.9556 - val_loss: 0.1538

```

```
Epoch 10/25
```

```

6/6 _____ 5s 791ms/step - accuracy: 0.9906 - loss: 0.0690 -
val_accuracy: 0.9778 - val_loss: 0.0872

```

```
Epoch 11/25
```

```

6/6 _____ 5s 861ms/step - accuracy: 0.9733 - loss: 0.0982 -
val_accuracy: 0.9556 - val_loss: 0.1202

```

```
Epoch 12/25
```

```

6/6 _____ 5s 875ms/step - accuracy: 0.9523 - loss: 0.1192 -
val_accuracy: 1.0000 - val_loss: 0.0472

```

```
Epoch 13/25
```

```

6/6 _____ 5s 850ms/step - accuracy: 0.9882 - loss: 0.0605 -
val_accuracy: 0.9778 - val_loss: 0.0622

```

```
Epoch 14/25
```

```

6/6 _____ 5s 875ms/step - accuracy: 0.9866 - loss: 0.0594 -
val_accuracy: 0.9778 - val_loss: 0.0481

```

```
Epoch 15/25
```

```

6/6 _____ 5s 831ms/step - accuracy: 0.9766 - loss: 0.0537 -
val_accuracy: 1.0000 - val_loss: 0.0322

```

```
Epoch 16/25
```

```

6/6 _____ 5s 806ms/step - accuracy: 0.9816 - loss: 0.0382 -
val_accuracy: 0.9778 - val_loss: 0.0609

```

```
Epoch 17/25
```

```

6/6 _____ 5s 846ms/step - accuracy: 0.9898 - loss: 0.0430 -
val_accuracy: 0.9778 - val_loss: 0.0492

```

```
Epoch 18/25
```

```

6/6 _____ 5s 869ms/step - accuracy: 0.9963 - loss: 0.0269 -
val_accuracy: 0.9556 - val_loss: 0.0532

```

```

Epoch 19/25
6/6 _____ 5s 822ms/step - accuracy: 0.9809 - loss: 0.0427 -
  val_accuracy: 0.9778 - val_loss: 0.0307
Epoch 20/25
6/6 _____ 5s 795ms/step - accuracy: 0.9937 - loss: 0.0247 -
  val_accuracy: 0.9778 - val_loss: 0.0530
Epoch 21/25
6/6 _____ 5s 832ms/step - accuracy: 0.9893 - loss: 0.0351 -
  val_accuracy: 0.9778 - val_loss: 0.0404
Epoch 22/25
6/6 _____ 5s 834ms/step - accuracy: 0.9875 - loss: 0.0600 -
  val_accuracy: 0.9778 - val_loss: 0.0585
Epoch 23/25
6/6 _____ 5s 826ms/step - accuracy: 0.9963 - loss: 0.0352 -
  val_accuracy: 1.0000 - val_loss: 0.0169
Epoch 24/25
6/6 _____ 5s 789ms/step - accuracy: 0.9964 - loss: 0.0221 -
  val_accuracy: 0.9778 - val_loss: 0.0400
Epoch 25/25
6/6 _____ 5s 874ms/step - accuracy: 0.9984 - loss: 0.0185 -
  val_accuracy: 0.9778 - val_loss: 0.0263

```

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save\_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my\_model.keras')` or `keras.saving.save\_model(model, 'my\_model.keras')`.

## 4. Interpretación de Resultados

Al final tiene que presentar la información en una matriz que muestre la calidad de su clasificación y evaluarla con la medida que guste.

		Predicción	
		Positivos	Negativos
Observación	Positivos	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Se estableció una muestra de 20 imágenes para backtesting llamada `validar_final` con el fin de ver el desempeño del modelo.

```

In [36]: import os
import shutil
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report
import matplotlib.pyplot as plt

# Cargar el modelo entrenado
model = load_model('modelo_clasificacion_paginas_mejorado.h5')

# Crear carpetas de salida
output_dir_blanco = 'Imagenes/Blanco_evaluado'
output_dir_documentos = 'Imagenes/Documentos_evaluado'
os.makedirs(output_dir_blanco, exist_ok=True)

```

```

os.makedirs(output_dir_documentos, exist_ok=True)

def mover_imagenes(model, carpeta_entrada, umbral=0.5):
    y_true = []
    y_pred = []

    # Recorrer recursivamente las subcarpetas
    for root, dirs, files in os.walk(carpeta_entrada):
        for img_name in files:
            img_path = os.path.join(root, img_name)
            try:
                # Verificar que el archivo es una imagen
                if not img_name.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp', '.gif')):
                    print(f"Archivo no soportado: {img_name}")
                    continue

                img = load_img(img_path, target_size=(224, 224)) # Asegurarse de que el
                img_array = img_to_array(img)
                img_array = np.expand_dims(img_array, axis=0) / 255.0

                prediccion = model.predict(img_array)
                prediccion_clase = 0 if prediccion < umbral else 1
                y_pred.append(prediccion_clase)

                # Etiqueta verdadera basada en la carpeta origen
                if 'Blanco' in root:
                    etiqueta_verdadera = 0
                elif 'Documentos' in root:
                    etiqueta_verdadera = 1
                else:
                    print(f"Carpeta desconocida: {root}")
                    continue
                y_true.append(etiqueta_verdadera)

                # Mover la imagen a la carpeta correspondiente
                if prediccion_clase == 0:
                    shutil.move(img_path, os.path.join(output_dir_blanco, img_name))
                else:
                    shutil.move(img_path, os.path.join(output_dir_documentos, img_name))

                # Imprimir información de depuración
                print(f"Imagen: {img_name}, Predicción: {prediccion[0][0]}, Clase Predic
            except PermissionError:
                print(f"Permiso denegado para el archivo: {img_path}")
            except Exception as e:
                print(f"Error procesando el archivo {img_path}: {e}")

    return y_true, y_pred

# Evaluar y mover las imágenes
y_true, y_pred = mover_imagenes(model, 'Imágenes/validar_final', umbral=0.5)

if y_true and y_pred:
    # Calcular la matriz de confusión
    cm = confusion_matrix(y_true, y_pred, labels=[0, 1])
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Blanco', 'Docume

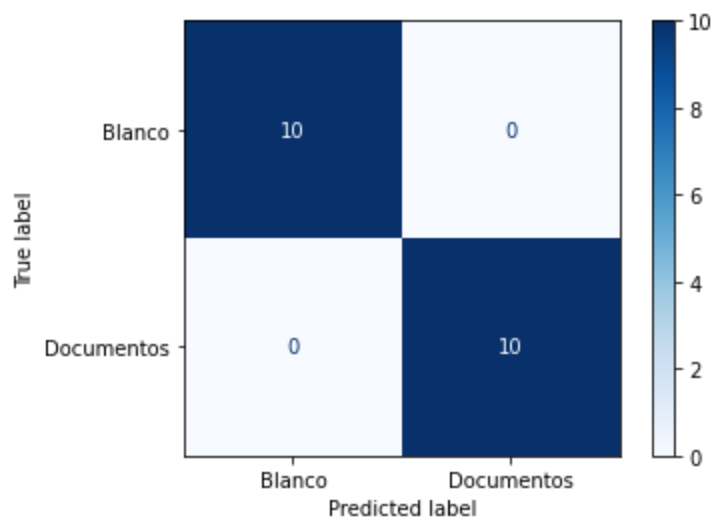
    # Mostrar la matriz de confusión
    disp.plot(cmap=plt.cm.Blues)
    plt.show()

    # Calcular y mostrar las métricas de clasificación
    report = classification_report(y_true, y_pred, target_names=['Blanco', 'Documentos'])
    print(report)
else:
    print("No se encontraron imágenes válidas para evaluar.")

```

WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile\_metrics` will be empty until you train or evaluate the model.

```
1/1 _____ 0s 166ms/step
Imagen: page-158_Blanco.png, Predicción: 0.00839244481176138, Clase Predicha: 0, Clase Verdadera: 0
1/1 _____ 0s 70ms/step
Imagen: page-181_Blanco.png, Predicción: 0.006868349388241768, Clase Predicha: 0, Clase Verdadera: 0
1/1 _____ 0s 57ms/step
Imagen: page-197_Blanco.png, Predicción: 0.006807956378906965, Clase Predicha: 0, Clase Verdadera: 0
1/1 _____ 0s 58ms/step
Imagen: page-220_Blanco.png, Predicción: 0.006538487039506435, Clase Predicha: 0, Clase Verdadera: 0
1/1 _____ 0s 51ms/step
Imagen: page-223_Blanco.png, Predicción: 0.006981114856898785, Clase Predicha: 0, Clase Verdadera: 0
1/1 _____ 0s 75ms/step
Imagen: page-238_Blanco.png, Predicción: 0.007121229078620672, Clase Predicha: 0, Clase Verdadera: 0
1/1 _____ 0s 53ms/step
Imagen: page-252_Blanco.png, Predicción: 0.0069662523455917835, Clase Predicha: 0, Clase Verdadera: 0
1/1 _____ 0s 67ms/step
Imagen: page-257_Blanco.png, Predicción: 0.00653996504843235, Clase Predicha: 0, Clase Verdadera: 0
1/1 _____ 0s 60ms/step
Imagen: page-261_Blanco.png, Predicción: 0.010118985548615456, Clase Predicha: 0, Clase Verdadera: 0
1/1 _____ 0s 64ms/step
Imagen: page-262_Blanco.png, Predicción: 0.01532309502363205, Clase Predicha: 0, Clase Verdadera: 0
1/1 _____ 0s 75ms/step
Imagen: page-119_Documentos.png, Predicción: 1.0, Clase Predicha: 1, Clase Verdadera: 1
1/1 _____ 0s 73ms/step
Imagen: page-34_Documentos.png, Predicción: 0.9999985694885254, Clase Predicha: 1, Clase Verdadera: 1
1/1 _____ 0s 57ms/step
Imagen: page-36_Documentos.png, Predicción: 1.0, Clase Predicha: 1, Clase Verdadera: 1
1/1 _____ 0s 52ms/step
Imagen: page-42_Documentos.png, Predicción: 1.0, Clase Predicha: 1, Clase Verdadera: 1
1/1 _____ 0s 65ms/step
Imagen: page-4_Documentos.png, Predicción: 1.0, Clase Predicha: 1, Clase Verdadera: 1
1/1 _____ 0s 69ms/step
Imagen: page-50_Documentos.png, Predicción: 1.0, Clase Predicha: 1, Clase Verdadera: 1
1/1 _____ 0s 54ms/step
Imagen: page-77_Documentos.png, Predicción: 1.0, Clase Predicha: 1, Clase Verdadera: 1
1/1 _____ 0s 67ms/step
Imagen: page-83_Documentos.png, Predicción: 1.0, Clase Predicha: 1, Clase Verdadera: 1
1/1 _____ 0s 61ms/step
Imagen: page-8_Documentos.png, Predicción: 1.0, Clase Predicha: 1, Clase Verdadera: 1
1/1 _____ 0s 56ms/step
Imagen: page-9_Documentos.png, Predicción: 1.0, Clase Predicha: 1, Clase Verdadera: 1
```



	precision	recall	f1-score	support
Blanco	1.00	1.00	1.00	10
Documentos	1.00	1.00	1.00	10
accuracy			1.00	20
macro avg	1.00	1.00	1.00	20
weighted avg	1.00	1.00	1.00	20

**SE EVIDENCIA UN BUEN RENDIMIENTO DEL MODELO, PREDICIENDO CORRECTAMENTE TODAS LAS IMAGENES DEL BACK TESTING**