

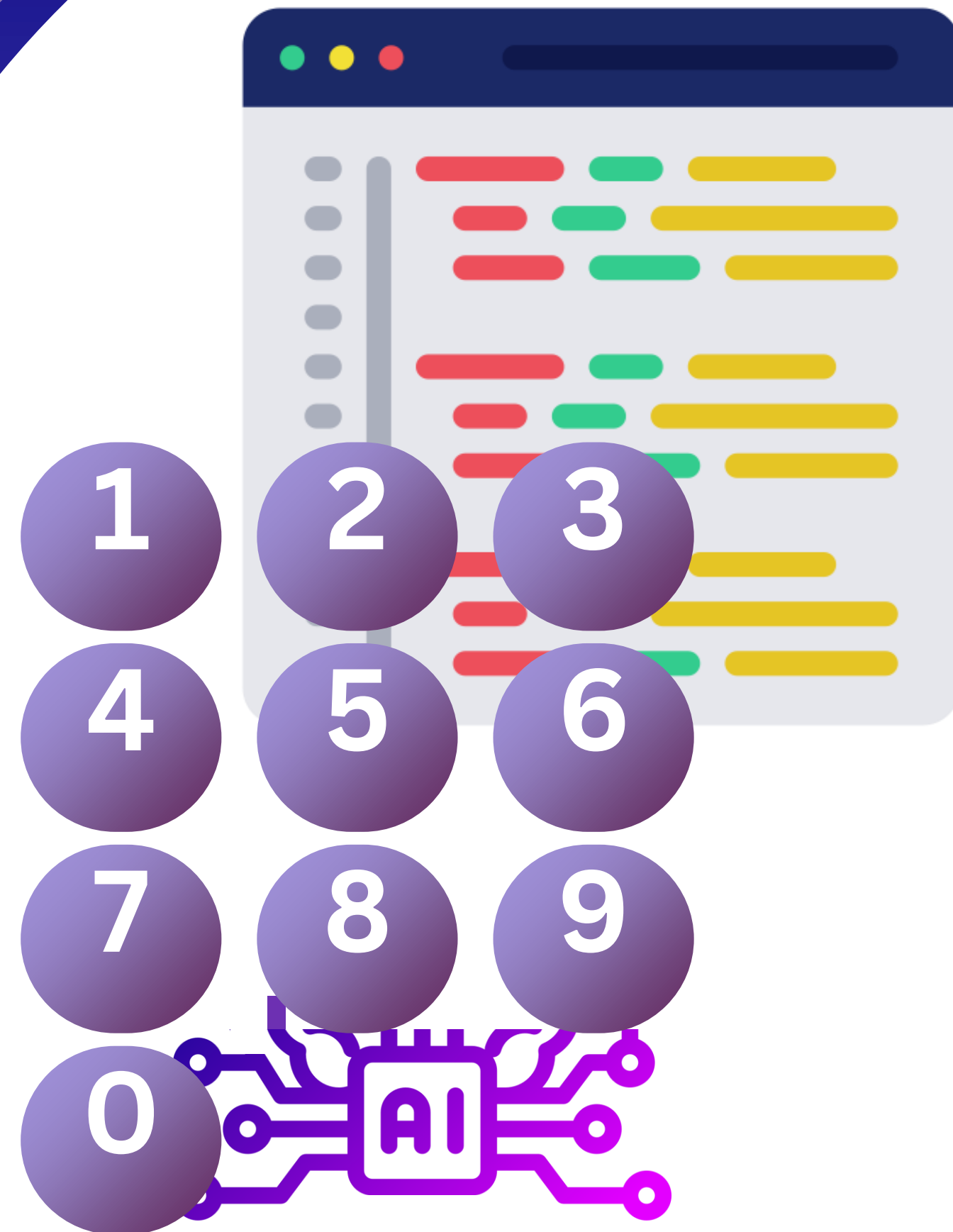
Reconocedor de Dígitos

Elaborado por:

María Tapia

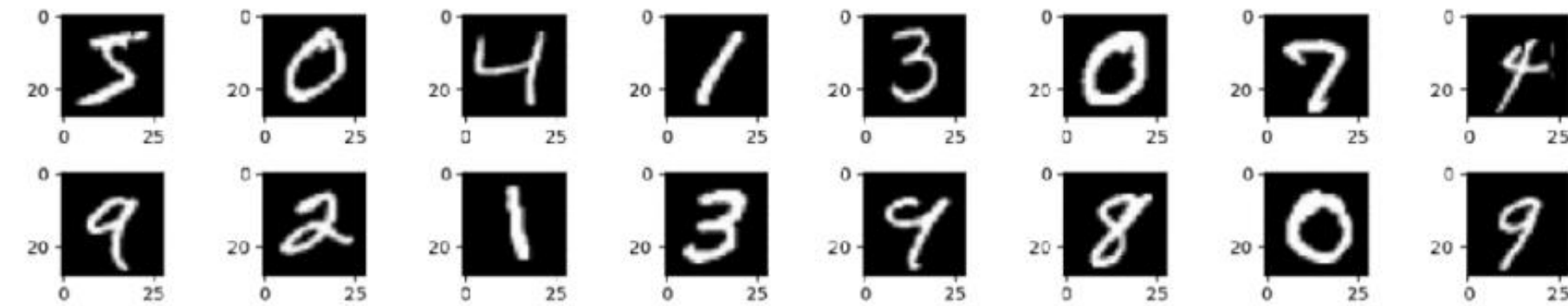
Mariela Gonzáles

Jonathan Sánchez



¿Qué es un reconocedor de dígitos ?

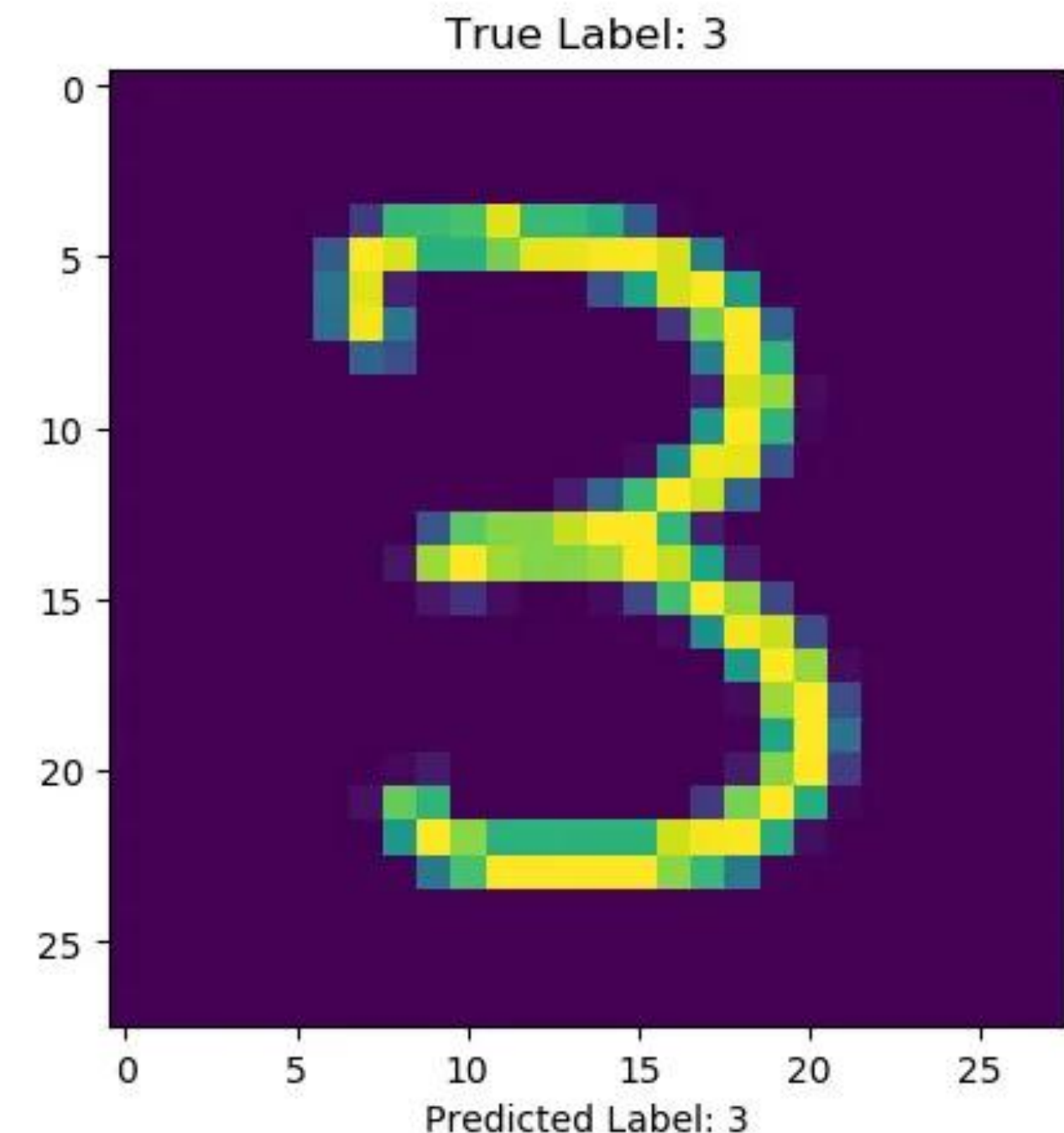
Un reconocedor de dígitos manuscritos es un sistema de inteligencia artificial que tiene la capacidad de interpretar y convertir imágenes de números escritos a mano (del 0 al 9) en un formato digital que una computadora pueda entender y procesar, este se integra en el software de las aplicaciones de banca móvil, así como en los cajeros automáticos más tradicionales.



Descripción del proyecto

El proyecto consiste en identificar correctamente los dígitos de un conjunto de datos de decenas de miles de imágenes escritas a mano.

El desarrollo de este proyecto ha sido completado utilizando el lenguaje python y librerías que han facilitado el procesamiento, filtrado y análisis de los datos.



Punto de partida

- Data set de digitos escritos a mano (MNIST)
- Carga de datos de entrenamiento y validación
- Visualización de datos
- Limpieza de datos

label	pixel0	pixel1	pixel2	pixel3
1	0	0	0	0
0	0	0	0	0
1	0	0	0	0
4	0	0	0	0
0	0	0	0	0
0	0	0	0	0
7	0	0	0	0
-	-	-	-	-

```
print(f" entrenamiento {train_data.shape}")  
print(f" test {test_data.shape}")
```

```
entrenamiento (42000, 785)  
test (28000, 784)
```

Objetivos

General

Desarrollar un modelo de aprendizaje automático capaz de clasificar con precisión imágenes de dígitos manuscritos (del 0 al 9).

Específicos

1

Aprender y aplicar fundamentos de visión artificial

2

Experimentar con diversos algoritmos de clasificación

3

Implementar y optimizar modelos de redes neuronales

4

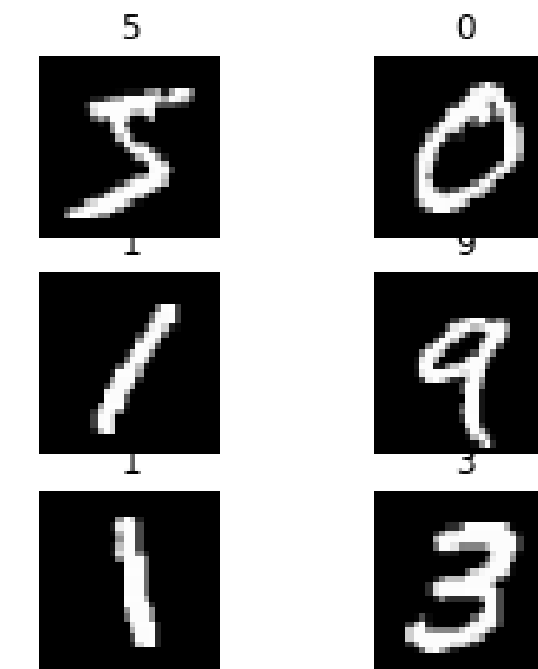
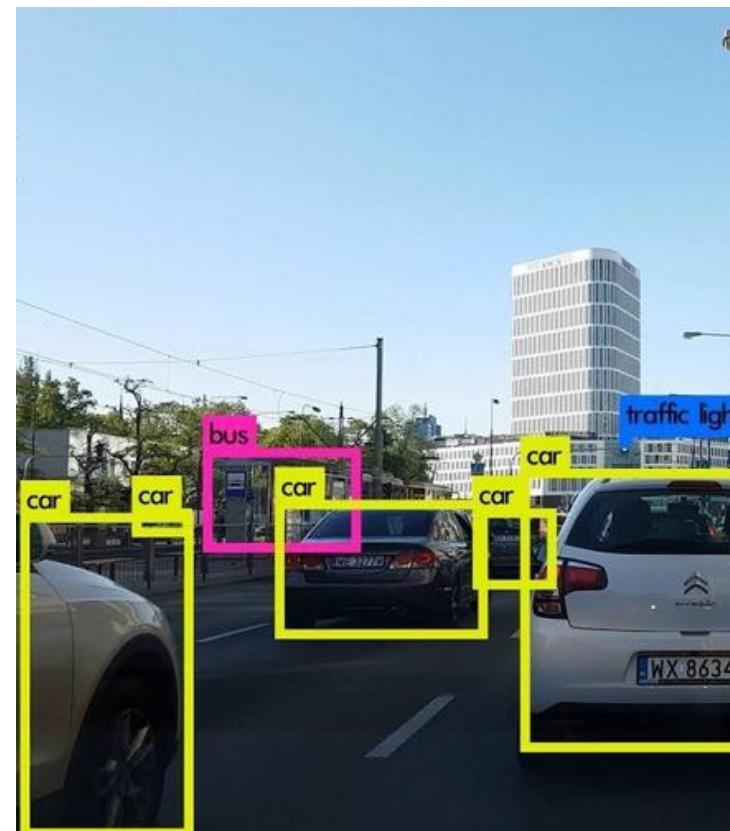
Generar predicciones en el formato requerido

5

Adquirir experiencia práctica en una plataforma de ciencia de datos

Razones de la elección

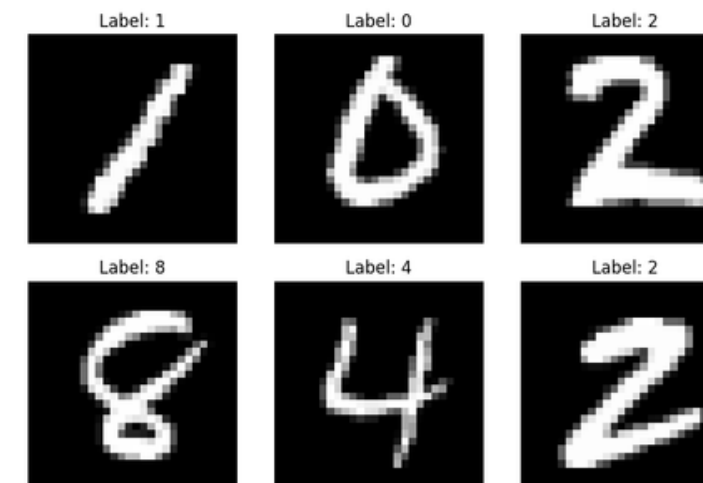
Se ha elegido este proyecto debido a su estrecha relación con el reconocimiento de patrones en imágenes, un campo de creciente importancia en la actualidad. La capacidad de identificar y clasificar dígitos a partir de imágenes sirve como una base sólida para comprender y desarrollar tecnologías más avanzadas, tales como reconocimiento facial, reconstruir imágenes, extracción de texto de imágenes.



Proceso de resolución de problemas

- Importar librerías, montar acceso a drive, cargar archivos de datos
- Visualizar los datos cargados
- Transformaciones y procesamiento
- Selección de características

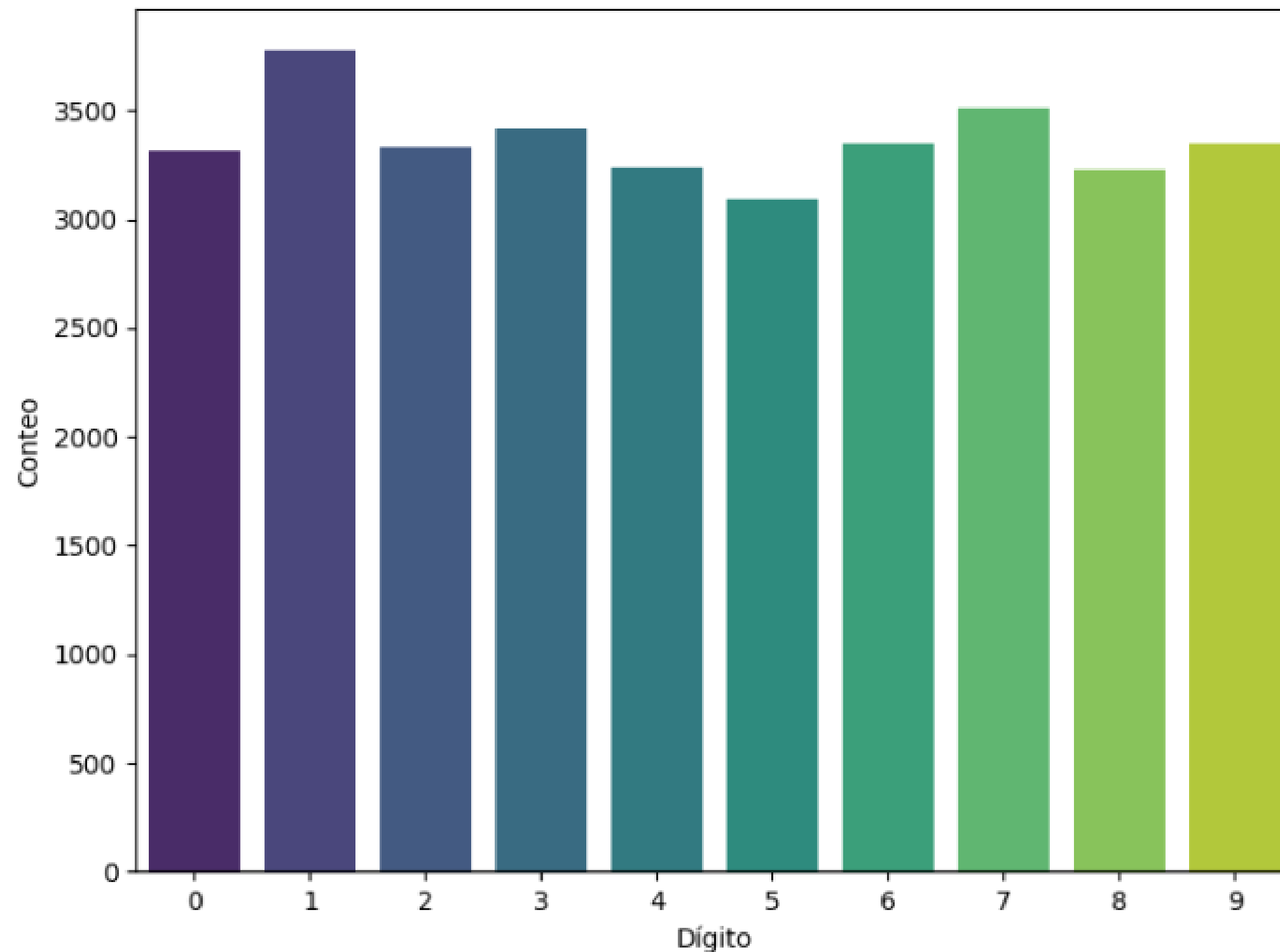
```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
import numpy as np
```








```
# Transformaciones
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])
```

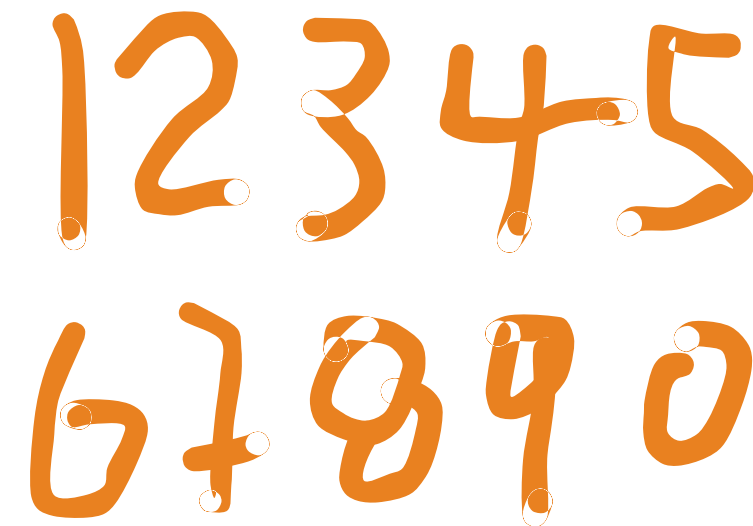
Distribucion de clases en el dataset de entrenamiento

Cantidad de datos de
entrenamientos por cada
numero para entrenar los
modelos



Modelos utilizados para la solución

-  Red Neuronal convolucional
-  K-Nearest Neighbors
-  Random Forest Classifier
-  Support Vector Machine
-  ResNet



Modelo: Red Neuronal Convolutacional (CNN)

- Selección de características implícita.
- Se detectan patrones y características relevantes en las imágenes (nn.Conv2d).
- Se reduce la dimensionalidad de las características, se eliminan píxeles individuales (nn.MaxPool2d).
- El modelo CNN por medio de los kernels de las capas convolucionales, aprende que pixeles son más relevantes que otros.
- Se cambia la forma de los datos, para poder pasarlos a las capas lineales (nn.Flatten()).

```
# Definición del modelo CNN
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv_layers = nn.Sequential(
            nn.Conv2d(1, 32, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
            nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )
        self.fc_layers = nn.Sequential(
            nn.Flatten(),
            nn.Linear(64 * 7 * 7, 128),
            nn.ReLU(),
            nn.Linear(128, 10)
        )

    def forward(self, x):
        x = self.conv_layers(x)
        x = self.fc_layers(x)
        return x
```

Modelo: K-Nearest Neighbors

- **n_neighbors=5:** Especifica la "K" en K-Vecinos Más Cercanos
- **n_jobs=-1:** controla cuántos núcleos de CPU o procesos puede usar el modelo para sus cálculos
- **knn_model.fit(X_train_knn, y_train_knn):** Esta línea entrena el modelo KNN utilizando tus datos de entrenamiento. Aquí es donde el modelo "aprende" de los ejemplos que le proporcionas

```
knn_model = KNeighborsClassifier(  
    n_neighbors=5,  
    n_jobs=-1)  
knn_model.fit(X_train_knn, y_train_knn)
```

Modelo: Random Forest Classifier

- **n_estimators=100:** Número de árboles de decisión que se construirán. Más árboles significan un modelo más robusto y preciso, pero mayor tiempo de entrenamiento y uso de memoria.
- **random_state=42:** esto asegura que si ejecutas el código varias veces con los mismos datos de entrada, obtendrás exactamente los mismos resultados
- **n_jobs=-1:** controla cuántos núcleos de CPU o procesos puede utilizar el modelo.
- **rf_model.fit(X_train_rf, y_train_rf).** Entrenamiento del modelo

```
rf_model = RandomForestClassifier(  
    n_estimators=100,  
    random_state=42,  
    n_jobs=-1)  
rf_model.fit(X_train_rf, y_train_rf)
```

Modelo: Support Vector Machine (SVM)

- **kernel='linear':** El "kernel" define la función que se utiliza para calcular la similitud entre los puntos de datos y para transformar los datos
- **random_state=42:** Al establecer un valor fijo se asegura que los valores sean los mismo al ejecutar varias veces el código.
- **verbose=False:** imprimirá o no mensajes detallados sobre su progreso de entrenamiento en la consola

```
svm_model = SVC(kernel='linear',  
                 random_state=42,  
                 verbose=False)  
  
svm_model.fit(X_train_svm_sub, y_train_svm_sub)  
  
y_pred_svm = svm_model.predict(X_val_svm)
```

Modelo: ResNet (Residual Network)

- **self.conv1 = nn.Conv2d(1, 16, kernel_size=3, stride=1, padding=1, bias=False):** define la primera capa de procesamiento principal de la red neuronal. u función es detectar patrones y características básicas en la imagen de entrada
- **self._make_layer(block, 64, layers[2], stride=2):** esta línea es la encargada de construir una sección profunda y crucial de la red neuronal
- **nn.AdaptiveAvgPool2d((1, 1))**
reducir drásticamente el tamaño espacial de los mapas de características a un único valor por cada canal

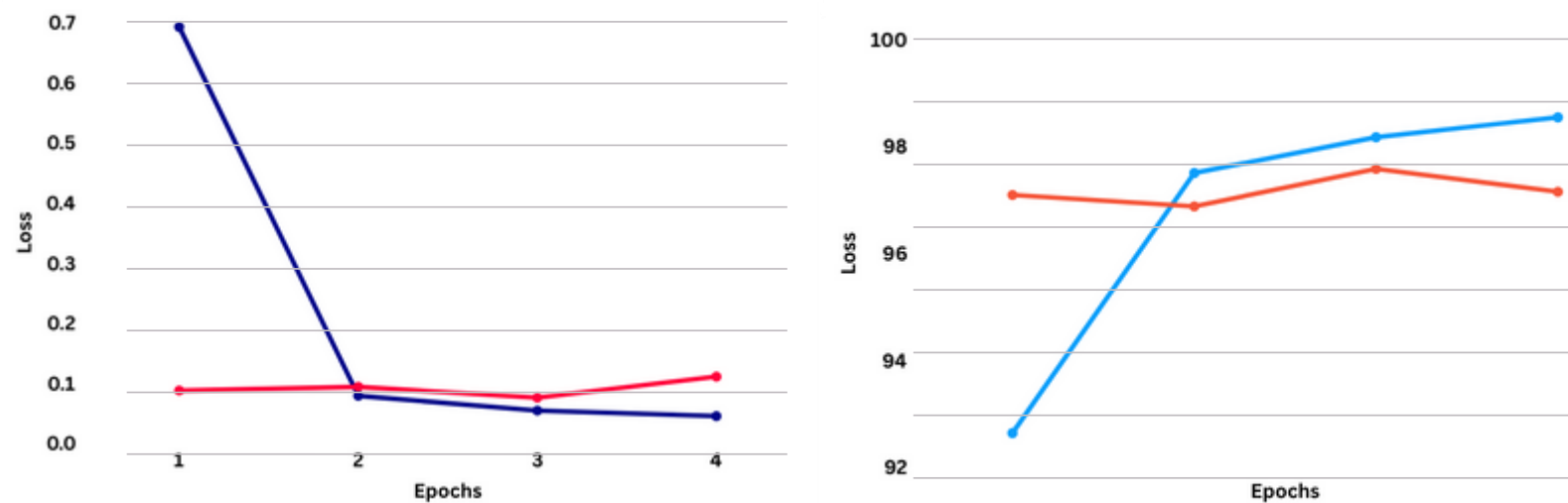
```
class ResNet(nn.Module):  
    def __init__(self, block, layers, num_classes=10):  
        super(ResNet, self).__init__()  
        self.in_channels = 16 # Start with fewer channels  
        self.conv1 = nn.Conv2d(1, 16, kernel_size=3, stride=1, padding=1, bias=False)  
        self.bn1 = nn.BatchNorm2d(16)  
        self.relu = nn.ReLU(inplace=True)  
        self.layer1 = self._make_layer(block, 16, layers[0], stride=1)  
        self.layer2 = self._make_layer(block, 32, layers[1], stride=2)  
        self.layer3 = self._make_layer(block, 64, layers[2], stride=2)  
        self.avg_pool = nn.AdaptiveAvgPool2d((1, 1)) # Global average pooling  
        self.fc = nn.Linear(64, num_classes)  
  
    def _make_layer(self, block, out_channels, blocks, stride):  
        layers = []  
        layers.append(block(self.in_channels, out_channels, stride))  
        self.in_channels = out_channels  
        for _ in range(1, blocks):  
            layers.append(block(out_channels, out_channels, 1))  
        return nn.Sequential(*layers)  
  
    def forward(self, x):  
        out = self.conv1(x)  
        out = self.bn1(out)  
        out = self.relu(out)  
        out = self.layer1(out)  
        out = self.layer2(out)  
        out = self.layer3(out)  
        out = self.avg_pool(out)  
        out = out.view(out.size(0), -1)  
        out = self.fc(out)  
        return out
```

Pérdida de entrenamiento y validación

Red Neuronal Convolutacional (CNN)

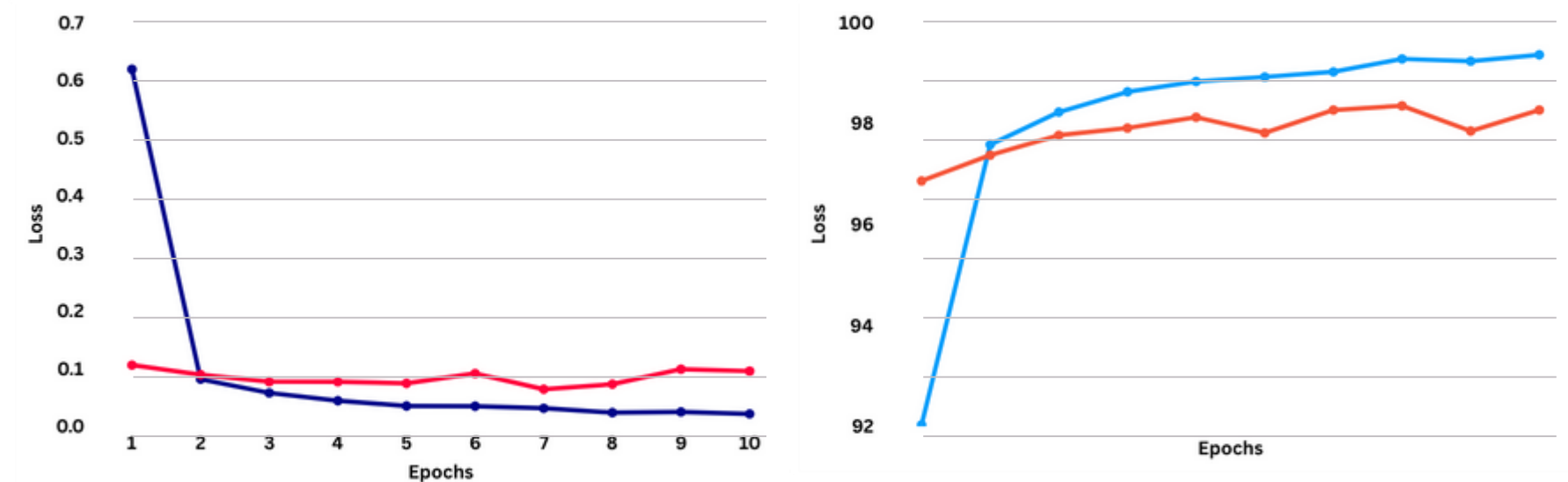
4

Epoch 1/4, Train Loss: 0.7081, Validation Loss: 0.0896
Epoch 2/4, Train Loss: 0.0736, Validation Loss: 0.0984
Epoch 3/4, Train Loss: 0.0487, Validation Loss: 0.0708
Epoch 4/4, Train Loss: 0.0384, Validation Loss: 0.0699



10

Epoch 1/10, Train Loss: 0.6187, Validation Acc: 96.86%
Epoch 2/10, Train Loss: 0.0812, Validation Acc: 97.37%
Epoch 3/10, Train Loss: 0.0575, Validation Acc: 97.76%
Epoch 4/10, Train Loss: 0.0439, Validation Acc: 97.90%
Epoch 5/10, Train Loss: 0.0348, Validation Acc: 98.12%
Epoch 6/10, Train Loss: 0.0344, Validation Acc: 97.81%
Epoch 7/10, Train Loss: 0.0309, Validation Acc: 98.26%
Epoch 8/10, Train Loss: 0.0233, Validation Acc: 98.35%
Epoch 9/10, Train Loss: 0.0246, Validation Acc: 97.85%
Epoch 10/10, Train Loss: 0.0212, Validation Acc: 98.26%

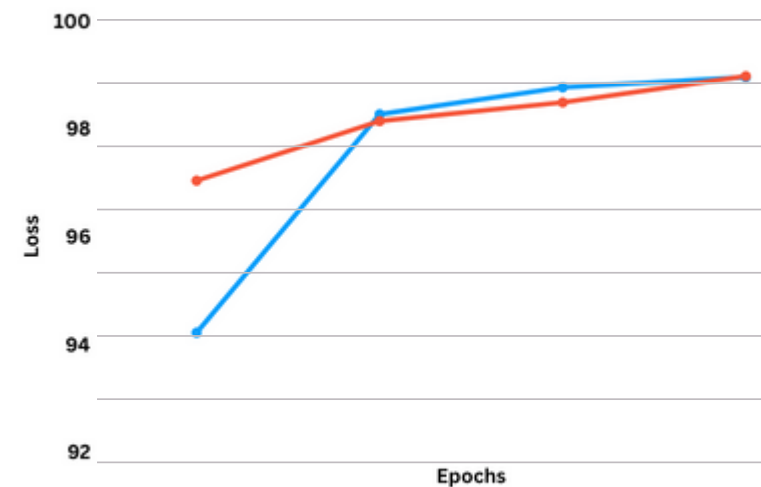
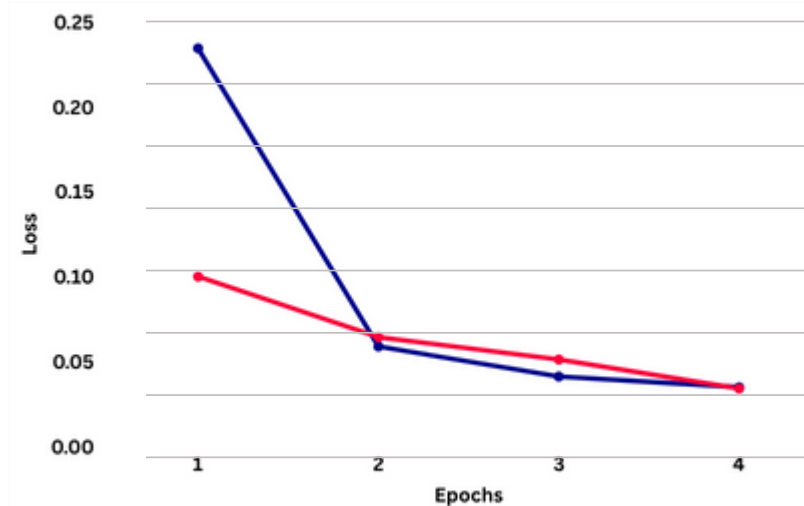


Perdida de entrenamiento y validación

ResNet (Residual Network)

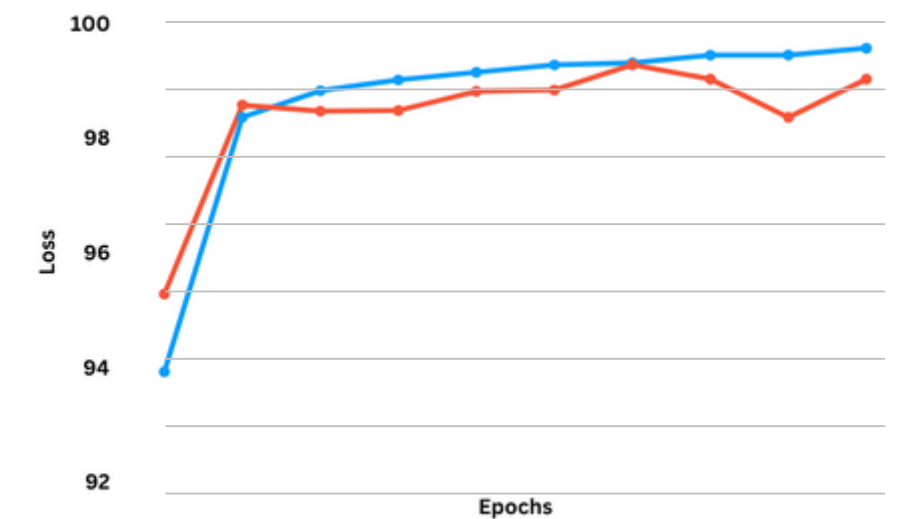
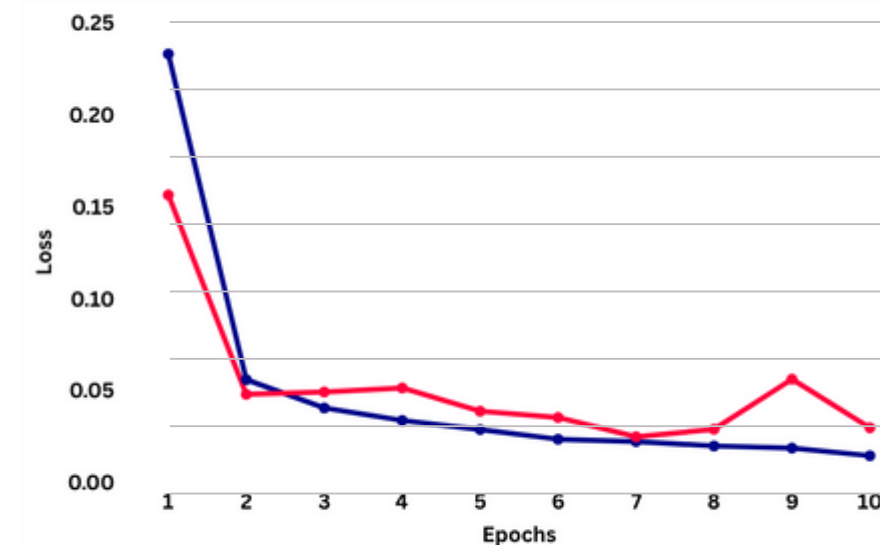
4

Epoch 1/4, Train Loss: 0.2345, Validation Loss: 0.1001
Epoch 2/4, Train Loss: 0.0589, Validation Loss: 0.0644
Epoch 3/4, Train Loss: 0.0412, Validation Loss: 0.0512
Epoch 4/4, Train Loss: 0.0349, Validation Loss: 0.0341



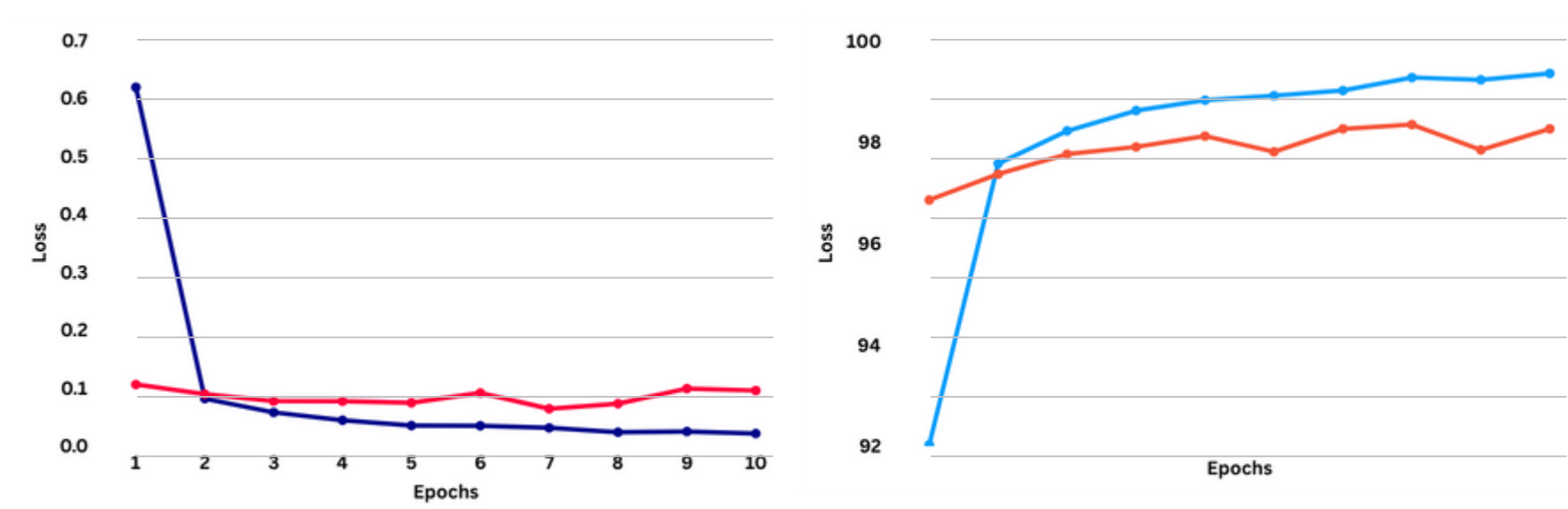
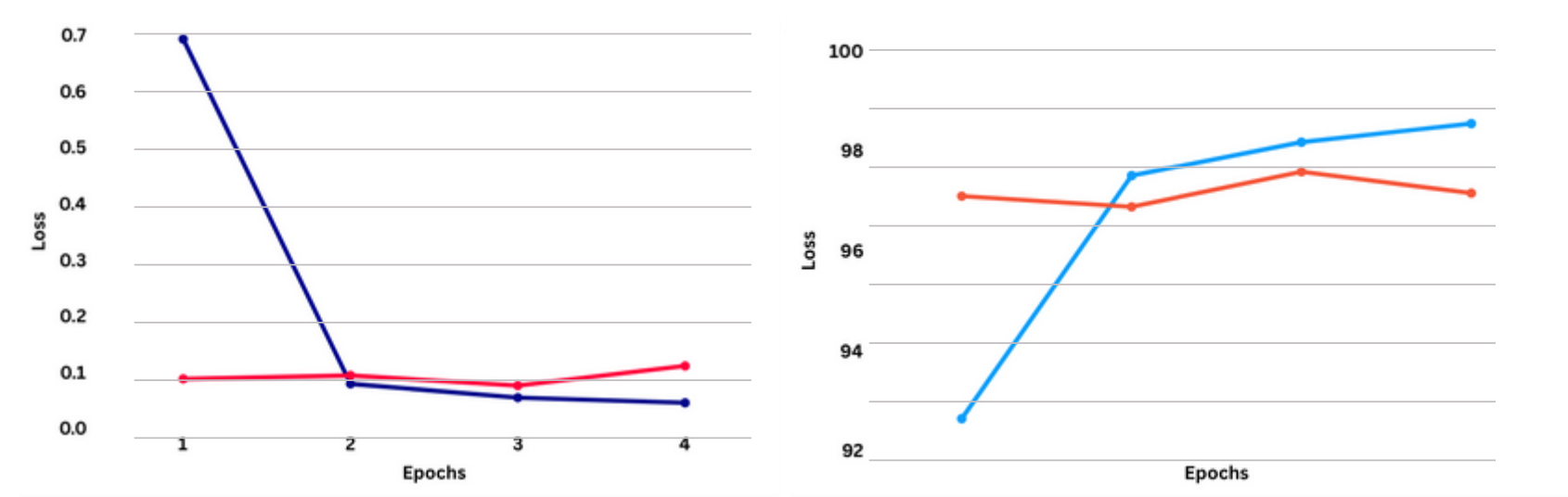
10

Epoch 1/10, Train Loss: 0.2331, Validation Acc: 95.27%
Epoch 2/10, Train Loss: 0.0559, Validation Acc: 98.57%
Epoch 3/10, Train Loss: 0.0405, Validation Acc: 98.46%
Epoch 4/10, Train Loss: 0.0337, Validation Acc: 98.48%
Epoch 5/10, Train Loss: 0.0287, Validation Acc: 98.81%
Epoch 6/10, Train Loss: 0.0234, Validation Acc: 98.83%
Epoch 7/10, Train Loss: 0.0222, Validation Acc: 99.27%
Epoch 8/10, Train Loss: 0.0198, Validation Acc: 99.02%
Epoch 9/10, Train Loss: 0.0186, Validation Acc: 98.36%
Epoch 10/10, Train Loss: 0.0145, Validation Acc: 99.02%

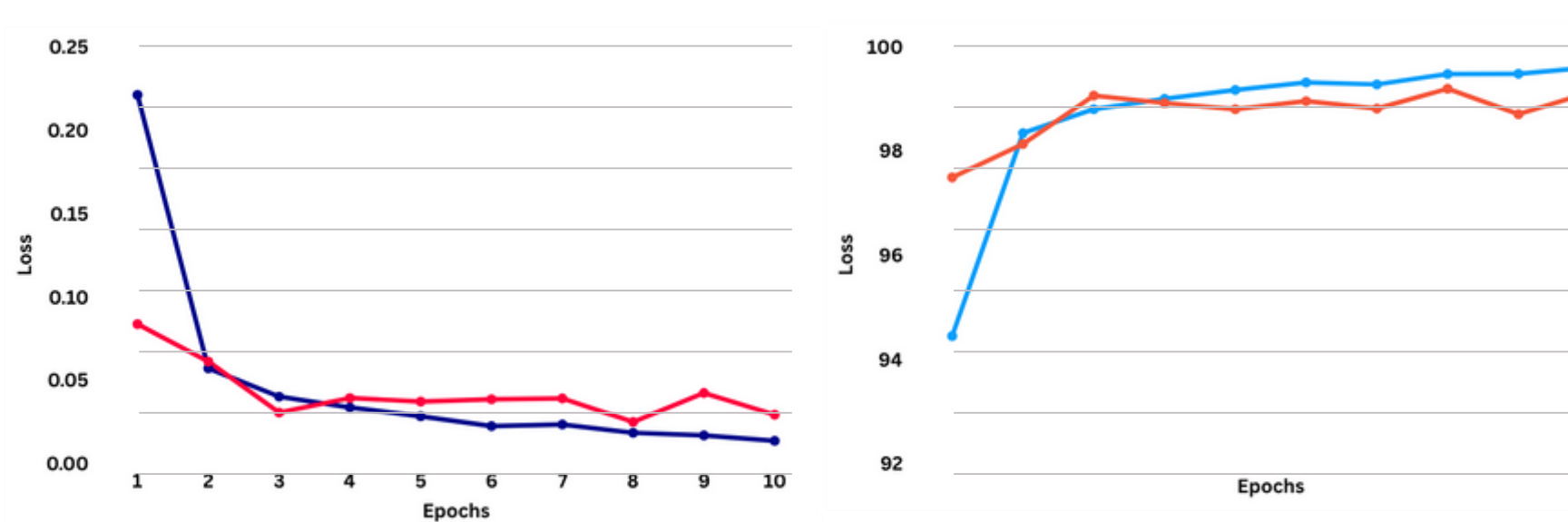
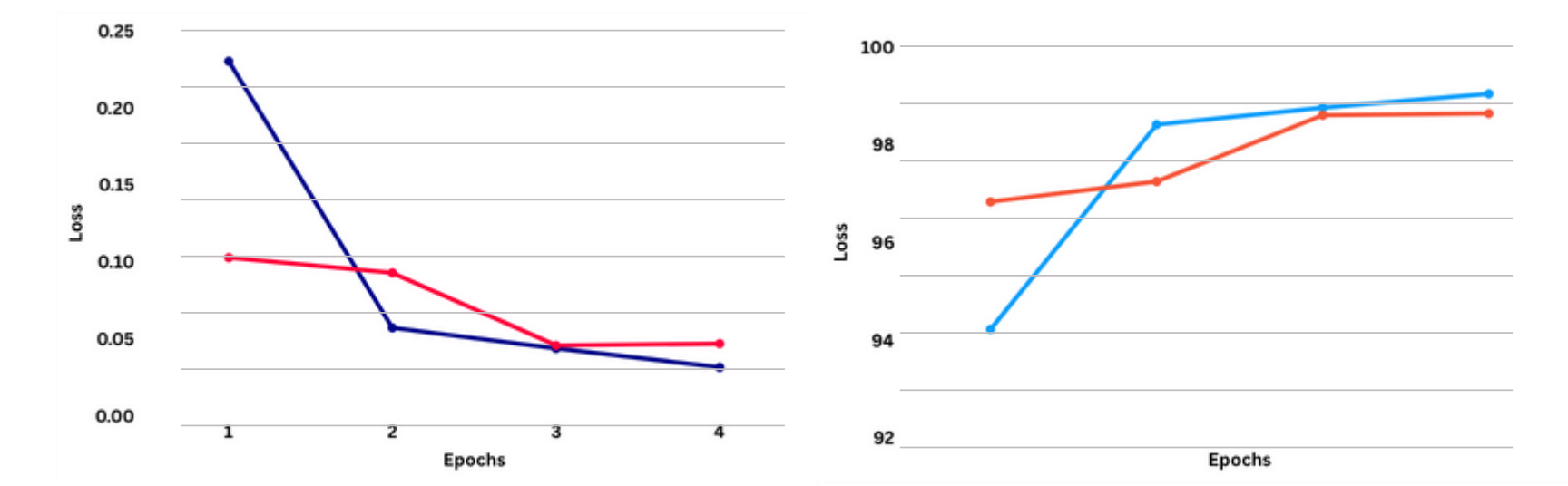


Perdida de entrenamiento y validación

Red Neuronal Convolutacional (CNN)

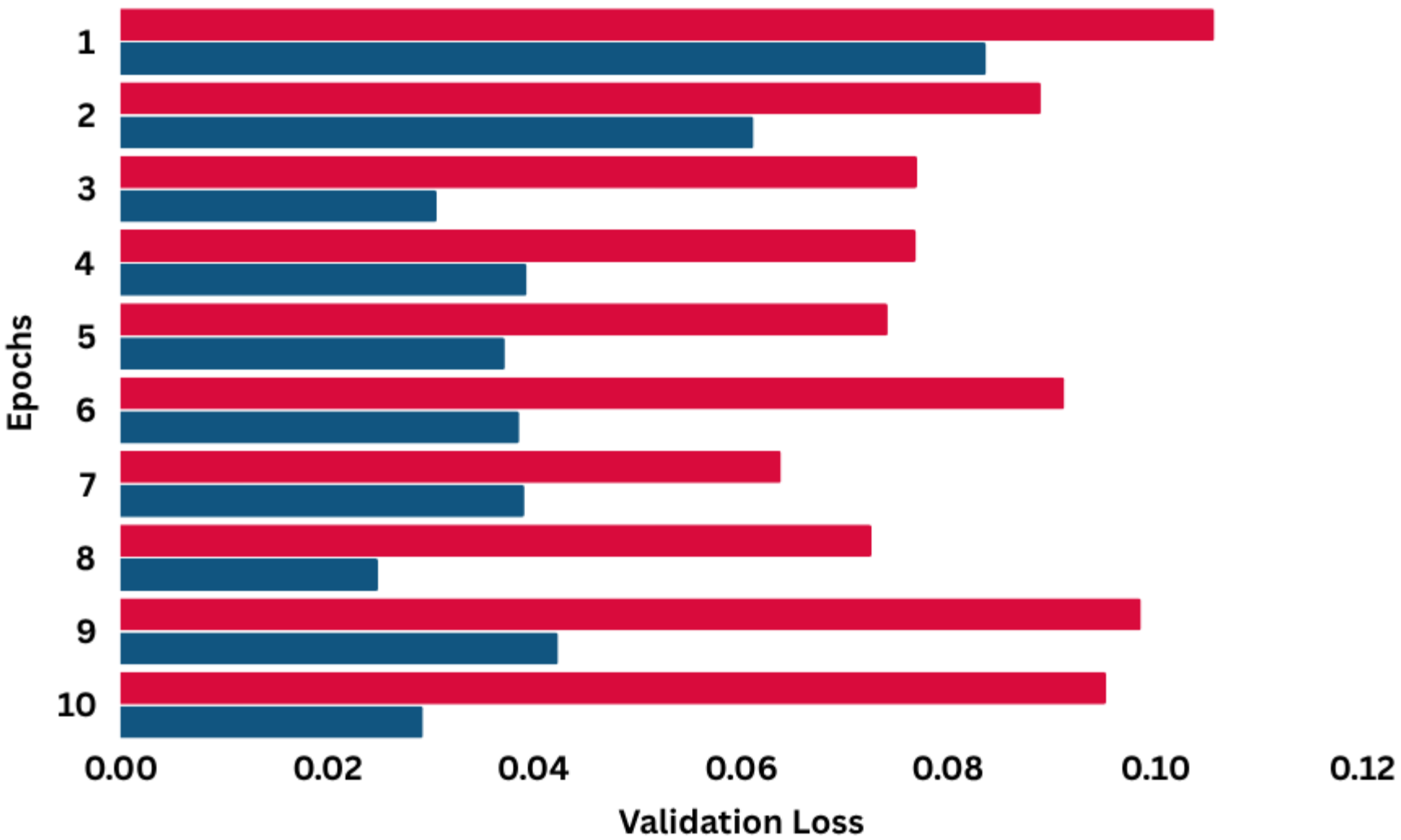
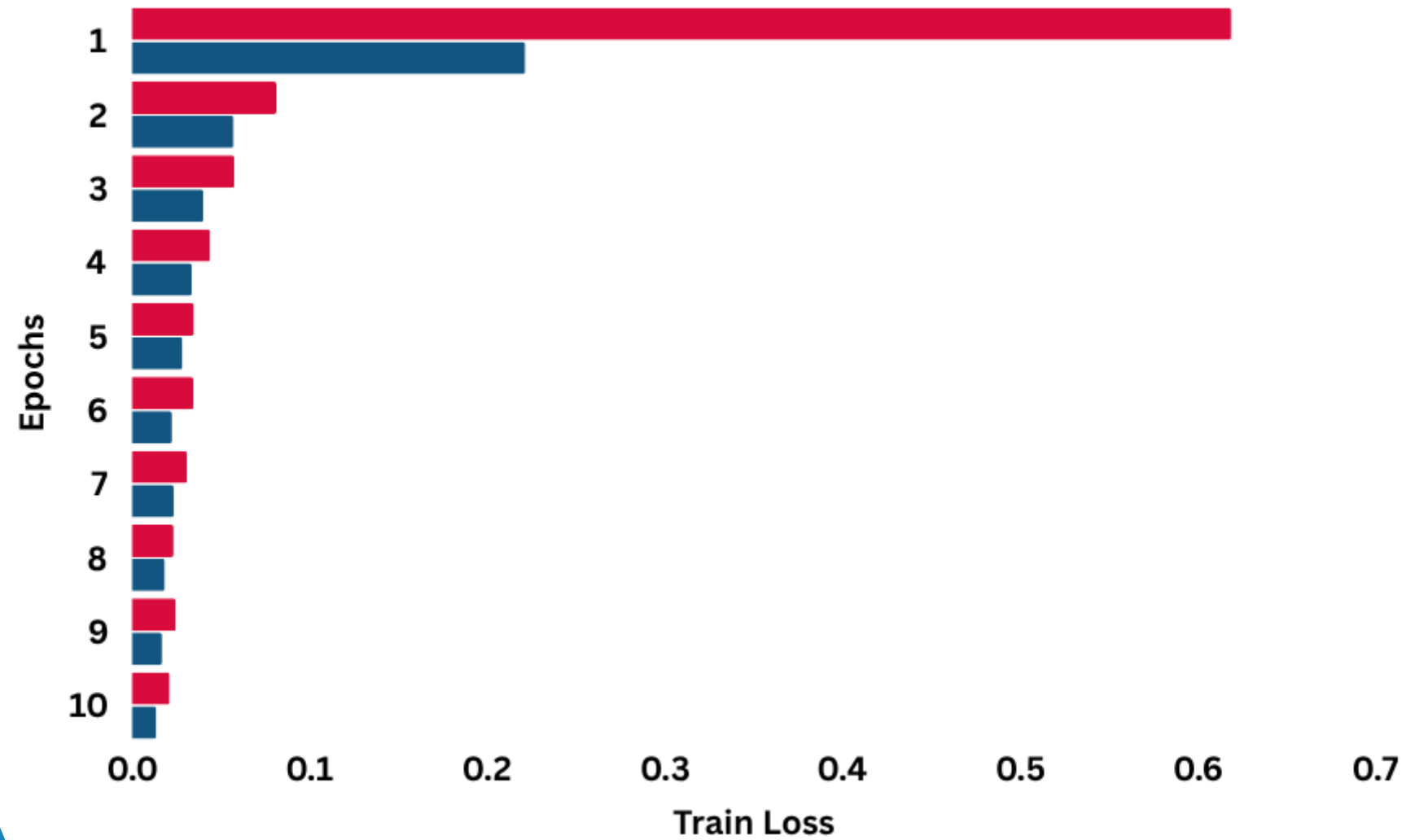


Res Net



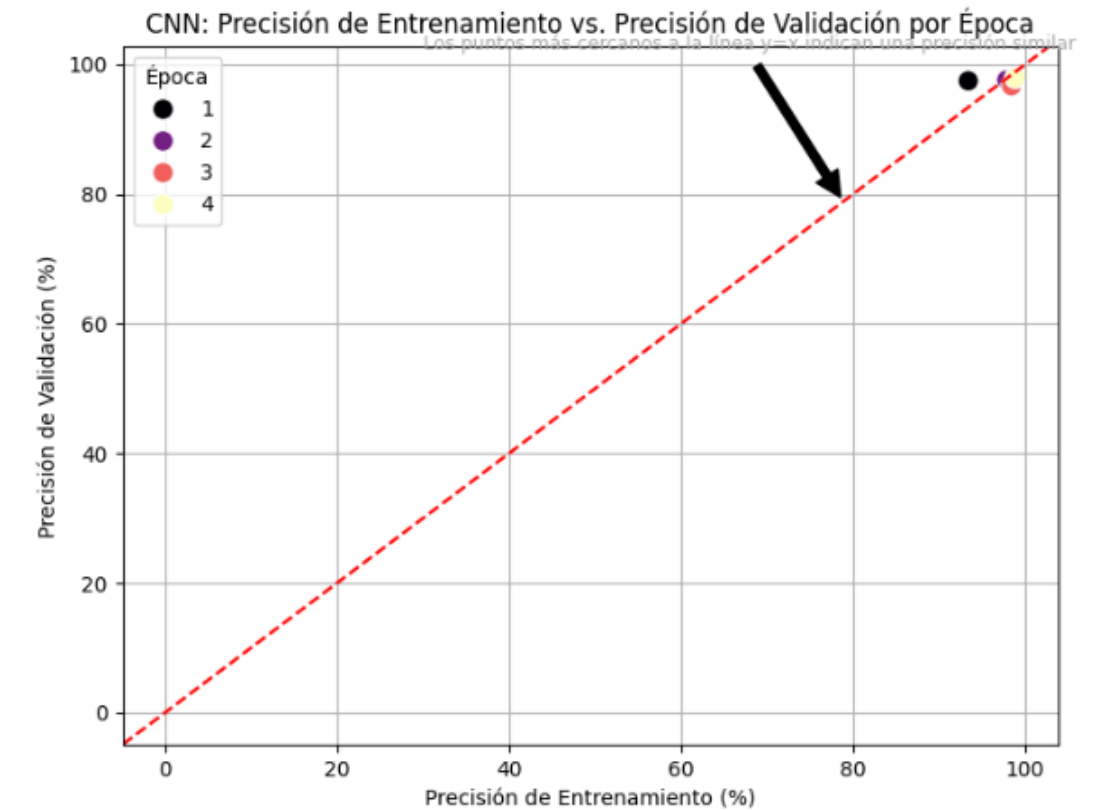
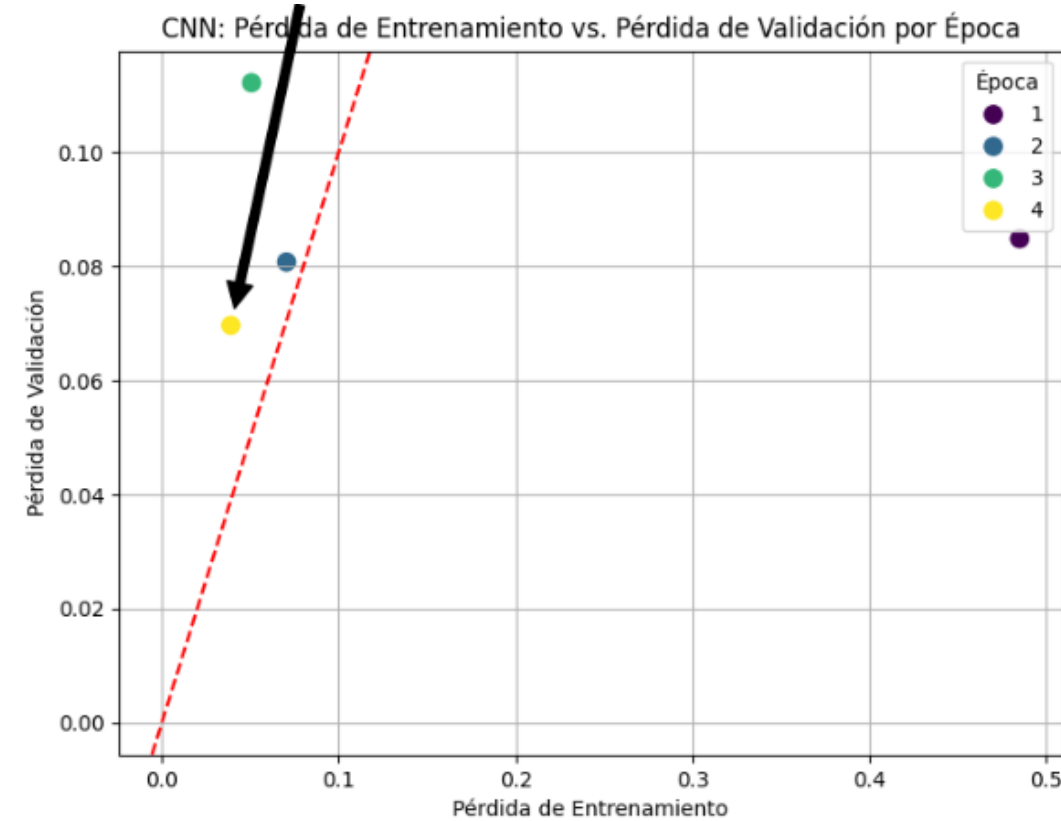
Perdida y validación

- Red Neuronal Convolucional (CNN)
- Res Net

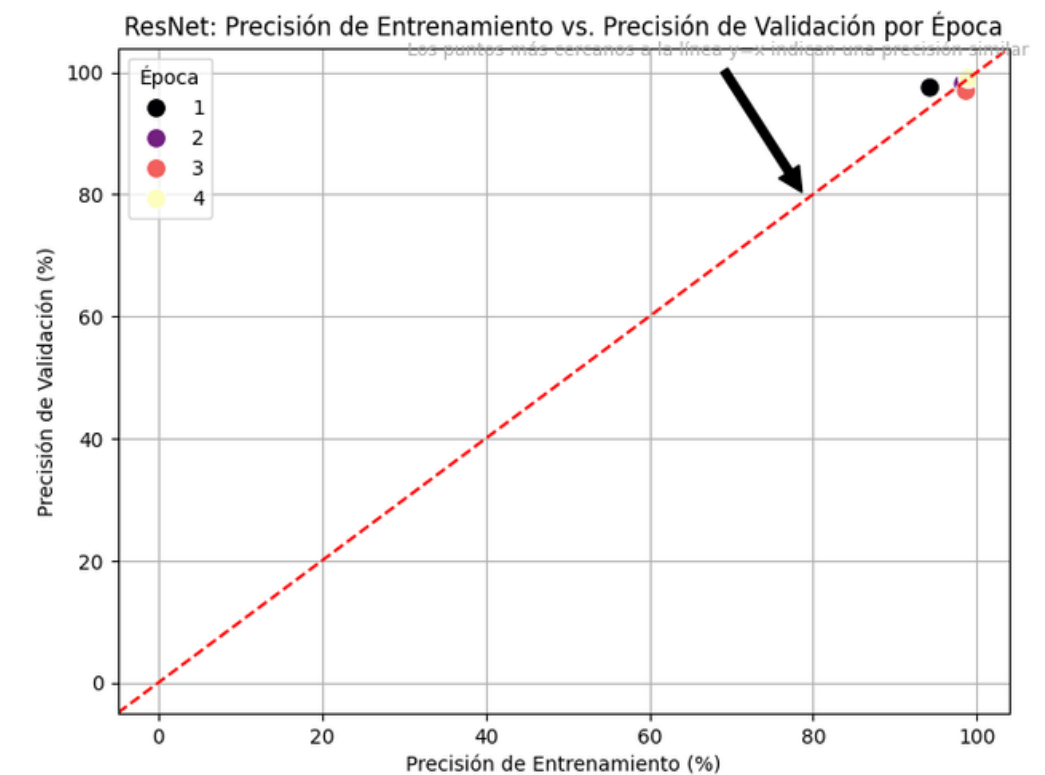
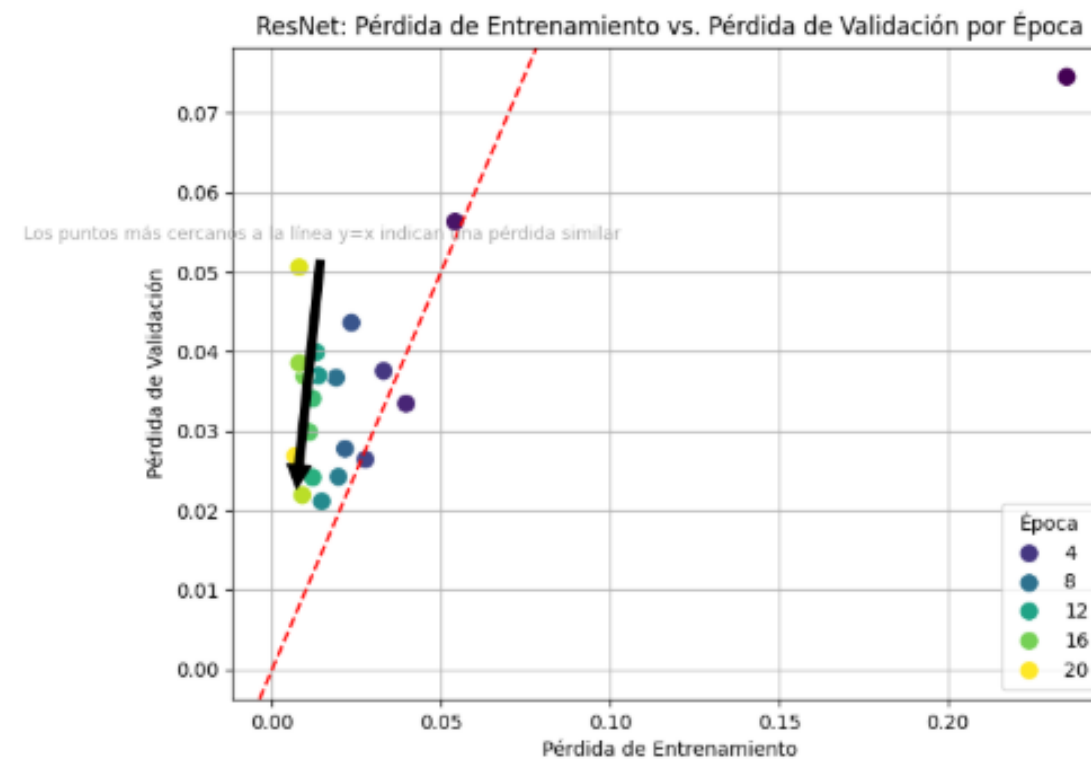


Perdida y validación

Red Neuronal Convolutiva (CNN)



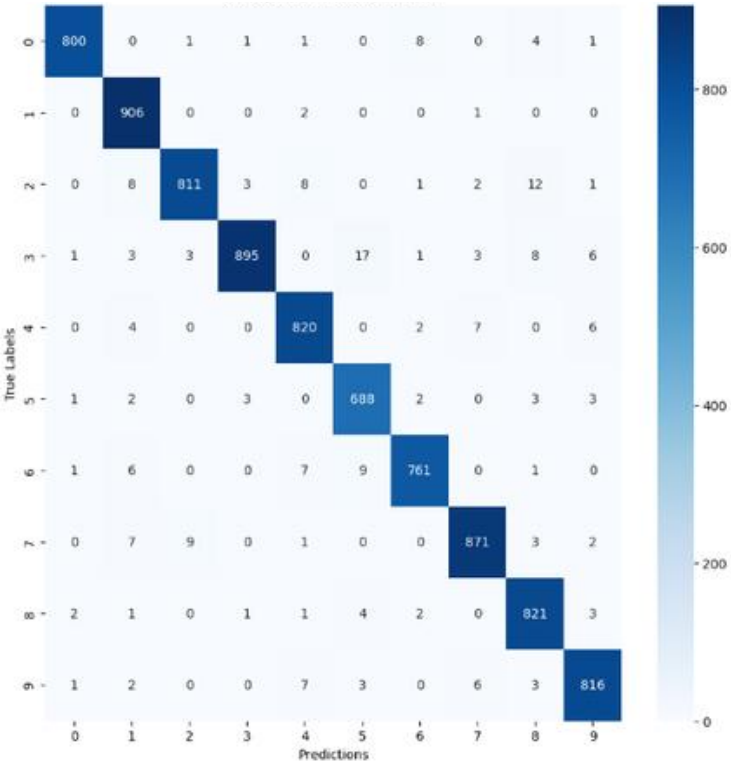
Res Net



Precisión

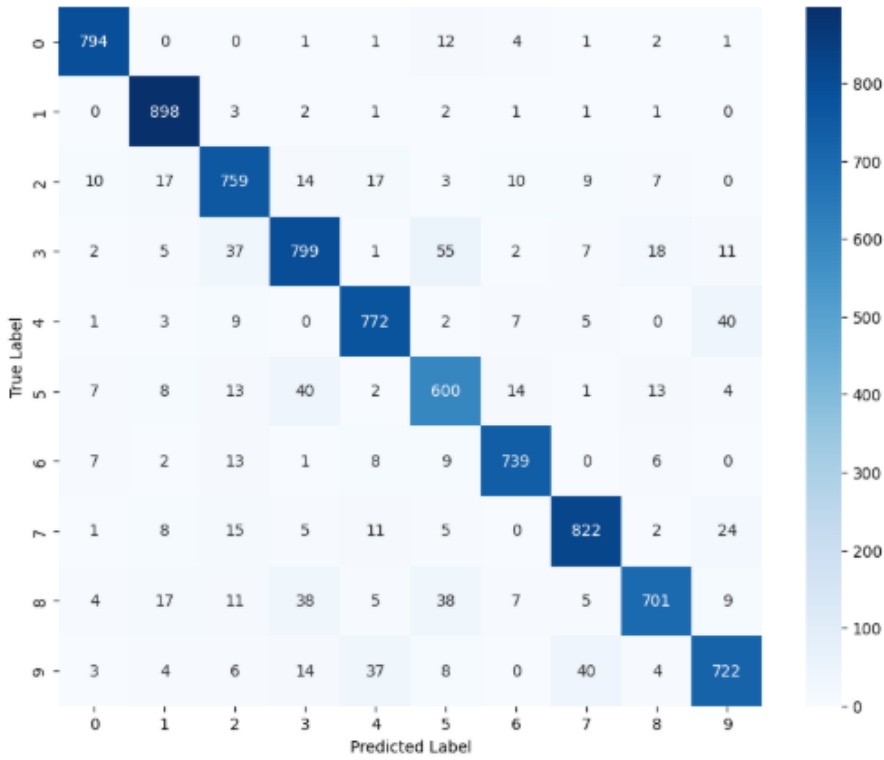
Red Neuronal Convolutacional

Precisión 97%



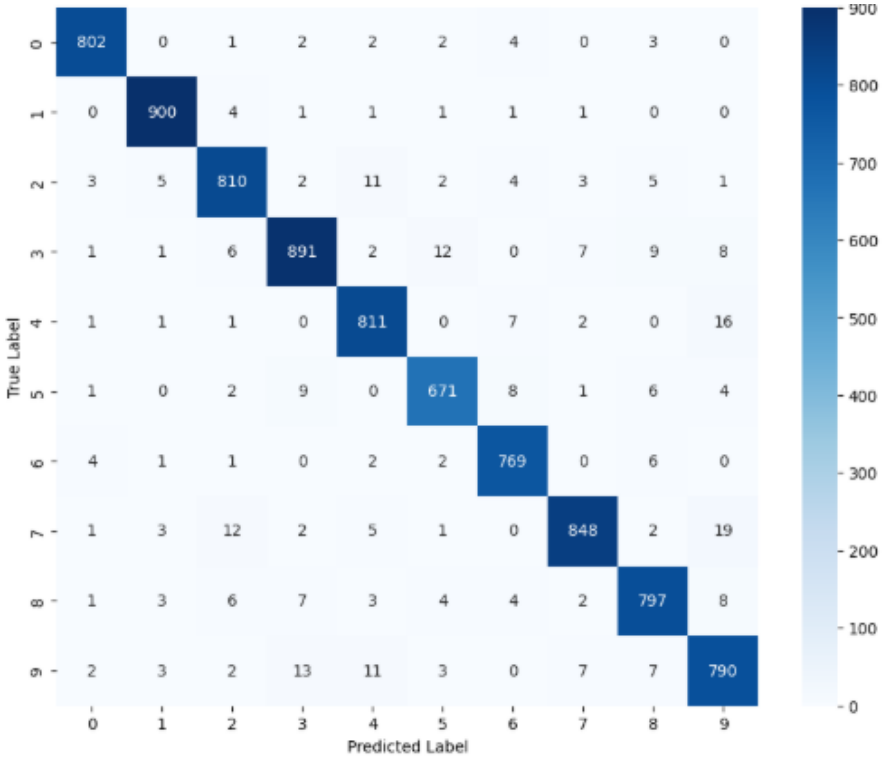
Support Vector Machine (SVM)

Precisión 90%



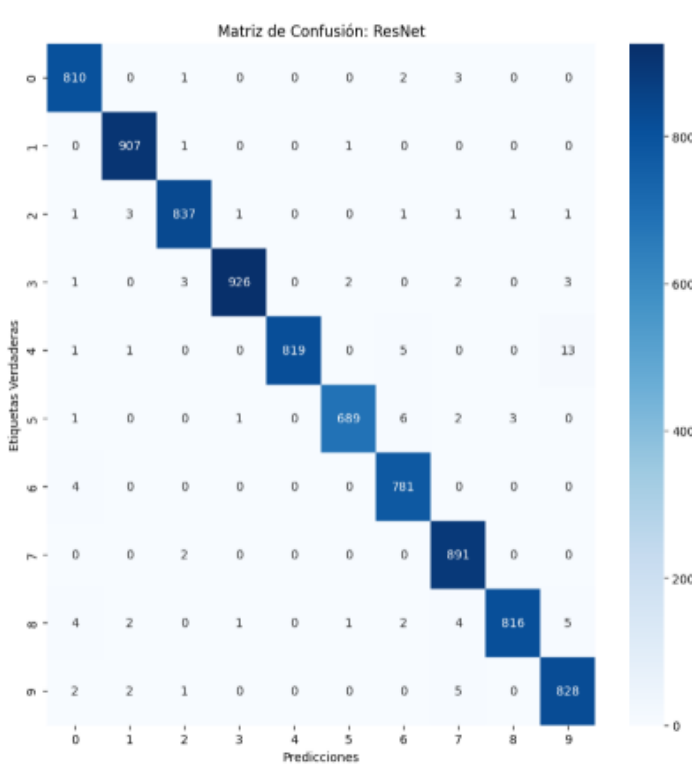
Random Forest Clasifier

Precisión 96%

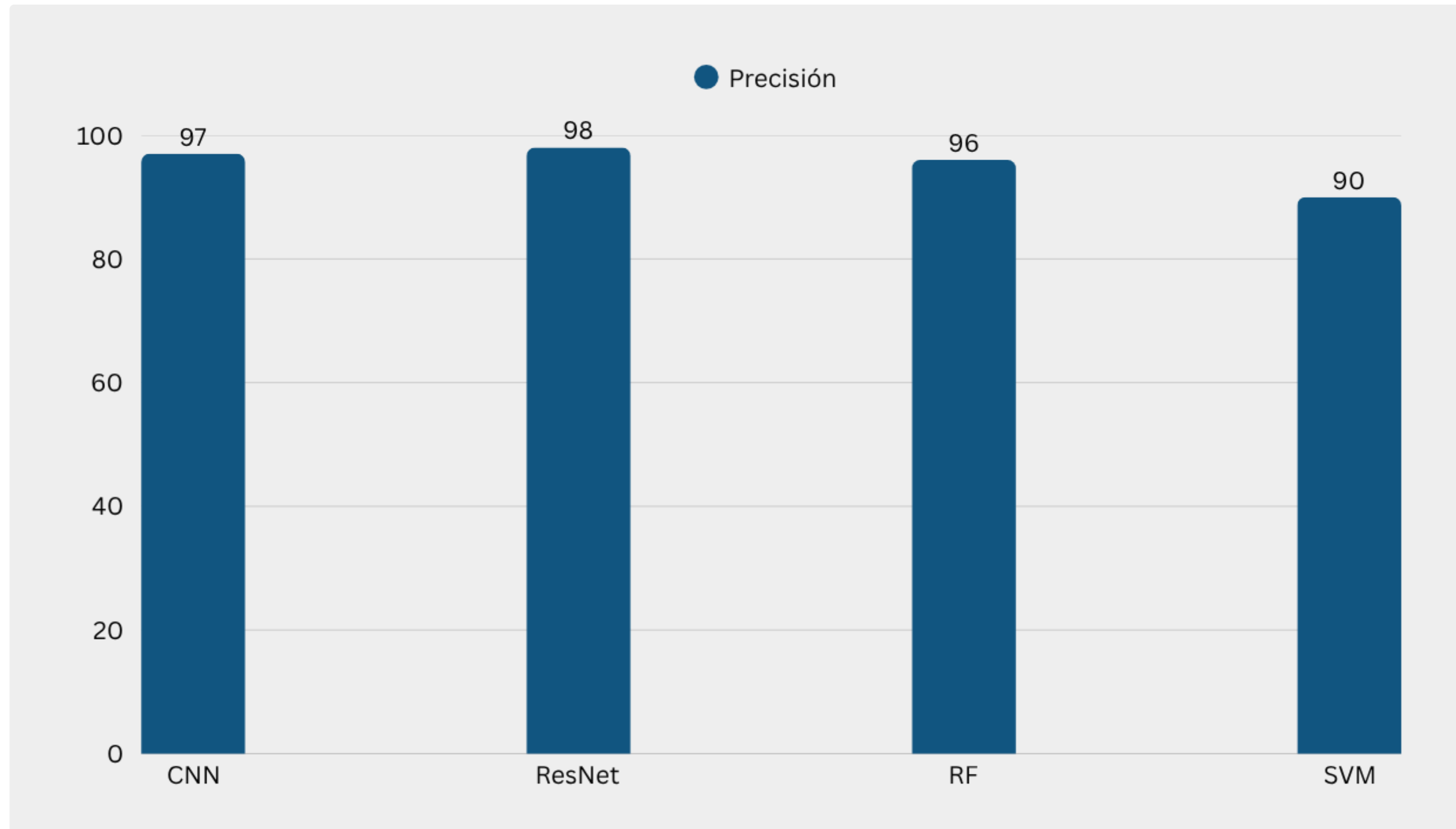


ResNet

Precisión 98%



Precisión



Evaluación en Kaggle

Digit Recognizer

Submit Prediction

...

Overview

Data

Code

Models

Discussion

Leaderboard

Rules

Team

Submissions

<div><div>✓</div></div>	<div>submission_resnet.csv</div> <div>Complete · now</div>	0.98539
<div><div>✓</div></div>	<div>submission_svm.csv</div> <div>Complete · 10m ago</div>	0.90635
<div><div>✓</div></div>	<div>submission_random_forest.csv</div> <div>Complete · 13m ago</div>	0.96257
<div><div>✓</div></div>	<div>submission_knn.csv</div> <div>Complete · 15m ago</div>	0.96507
<div><div>✓</div></div>	<div>sample_submission.csv</div> <div>Complete · 21h ago · Digit recognizer with convolutional neural network</div>	0.97775

790

Jonathanjsanchez08

0.98539

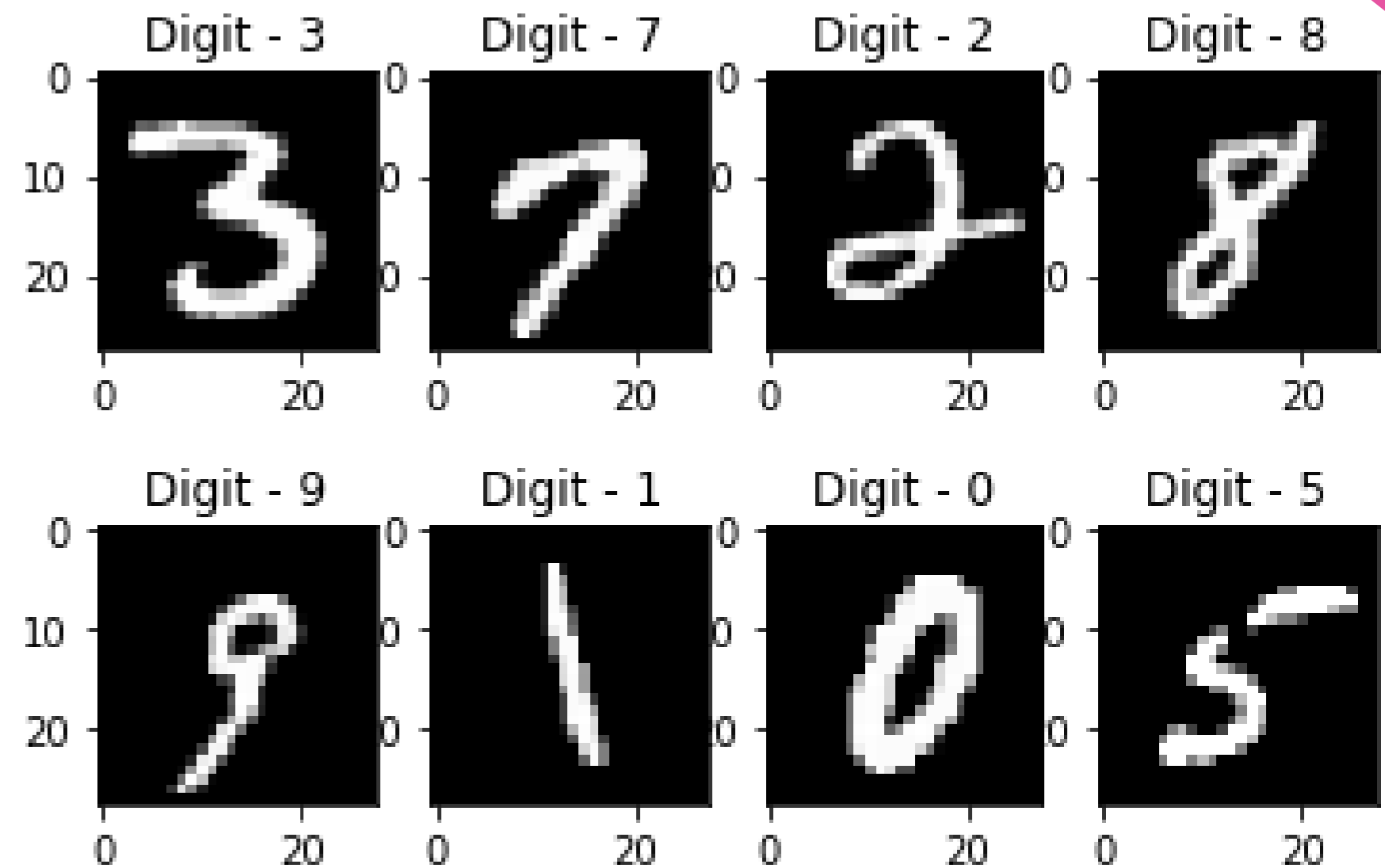
5

14d

(Ranking 790 de 1375 participantes)

Conclusión

Este modelo implementa una Red Neuronal Convolutiva (CNN) para la tarea de reconocimiento de dígitos escritos a mano, utilizando grandes conjuntos de datos de entrenamiento y validación teniendo en cuenta que estos datos deben ser limpiados y optimizados para su efectivo aprendizaje. Los resultados obtenidos muestran que la CNN es efectiva para esta tarea, logrando una alta precisión en la clasificación de dígitos.



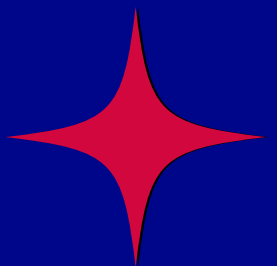
Procesamiento del Lenguaje Natural con Tweets de Desastres

Elaborado por:

María Tapia

Mariela Gonzáles

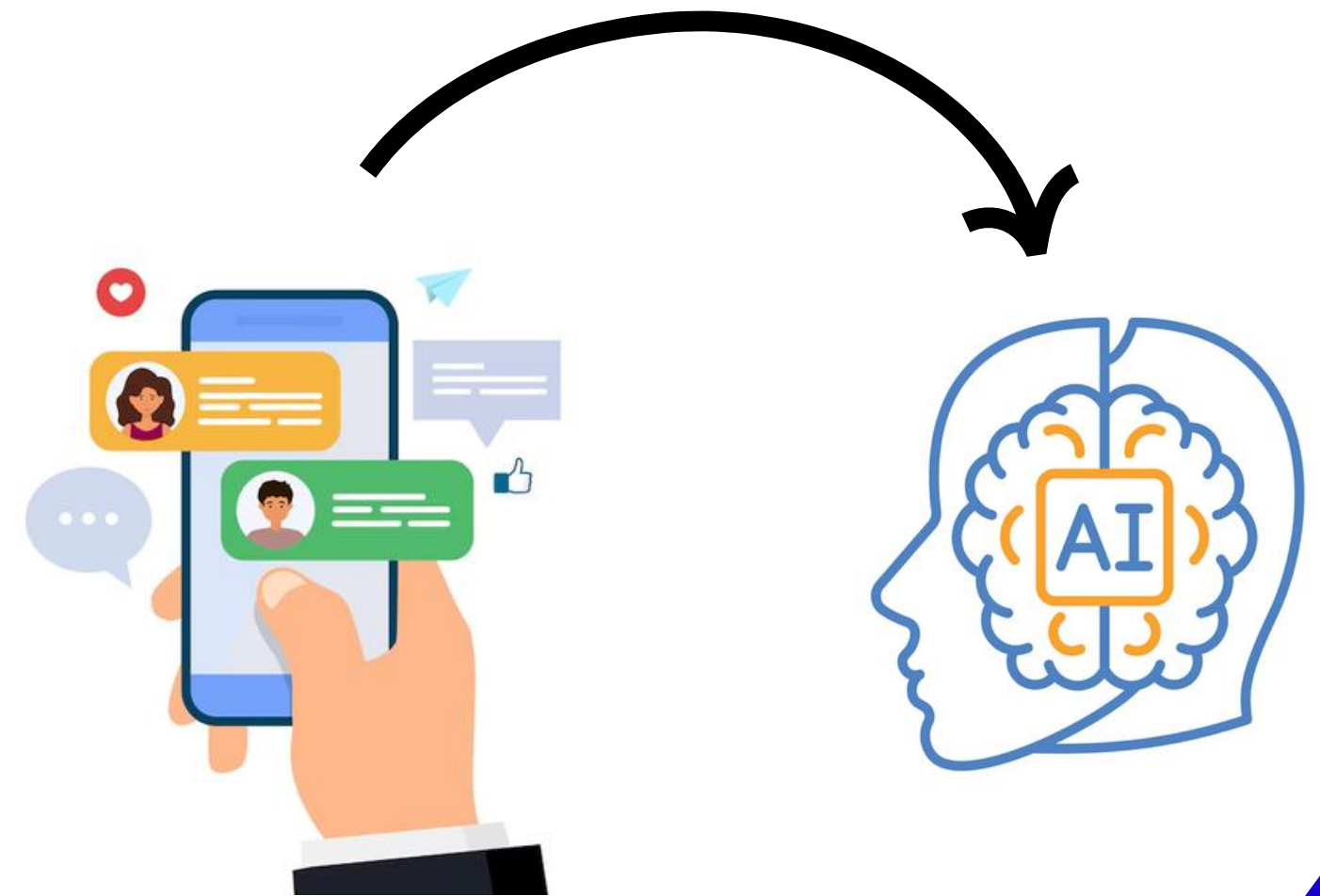
Jonathan Sánchez



Procesamiento de Lenguaje Natural

El procesamiento de lenguaje natural o PLN es un subcampo de la informática y la inteligencia artificial que emplea el machine learning para permitir que las computadoras comprendan y se comuniquen con el lenguaje humano.

En la sociedad actual, las redes sociales son una fuente principal de información. Sin embargo, durante desastres naturales, estas plataformas pueden difundir tanto datos verídicos como información errónea. Para combatir la desinformación en momentos críticos, los modelos de inteligencia artificial (IA) con PLN son esenciales. Estos modelos pueden clasificar la información, identificando datos correctos y ayudando a la población a tomar decisiones informadas durante una crisis.





Descripción del proyecto

El proyecto consiste en clasificar correctamente tweets como pertenecientes a desastres reales o no, a partir de un conjunto de datos .

Este proyecto se ha desarrollado utilizando el lenguaje Python y librerías que han facilitado el procesamiento y análisis de texto, aplicando técnicas de Procesamiento del Lenguaje Natural (NLP). El objetivo es crear un modelo de aprendizaje automático que pueda identificar patrones y características en el texto de los tweets que indiquen si describen un desastre real.



Punto de partida

- Data set
- Carga de datos de entrenamiento y validación
- Visualización de datos
- Clasificación de tweets

id	keyword	location	text
0	,	,	Just happened a terrible car crash
2	,	,	"Heard about #earthquake is different cities, stay s
3	,	,	"there is a forest fire at spot pond, geese are fleein
9	,	,	Apocalypse lighting. #Spokane #wildfires
11	,	,	Typhoon Soudelor kills 28 in China and Taiwan
12	,	,	We're shaking...It's an earthquake
21	,	,	"They'd probably still show more life than Arsena
22	,	,	Hey! How are you?
27	,	,	What a nice hat?
30	,	,	No I don't like cold!
35	,	,	NOOOOOOOOOO! Don't do that!
42	,	,	No don't tell me that!
43	,	,	What if?!

```
print(f"entrenamiento {train_df.shape}")
print(f"test {test_df.shape}")
```

```
Mounted at /content/drive
entrenamiento (7613, 5)
test (3263, 4)
```

Objetivos

General

Desarrollar un modelo de aprendizaje automático que pueda clasificar tweets para determinar si están relacionados con desastres reales

Específicos

1 Exploración y preprocesamiento de datos

2 Desarrollo de modelos de NLP

3 Evaluación del rendimiento de los modelos

4 Generación de predicciones

5 Adquirir experiencia en NLP



Razones de la elección

Este proyecto de clasificación de tweets se eligió debido a su estrecha relación con el análisis de texto y el Procesamiento del Lenguaje Natural (NLP), un campo de creciente importancia en la actualidad. La capacidad de identificar y clasificar tweets relacionados con desastres reales sirve como una base sólida para comprender y desarrollar tecnologías más avanzadas, tales como el análisis de sentimiento en redes sociales, la detección temprana de crisis y la mejora en la respuesta a emergencias.



Proceso de resolución de problemas

- Importar librerías, montar acceso a drive, cargar archivos de datos
- Visualizar los datos cargados
- Limpiar los datos
- Se eliminan URLs y caracteres no alfabéticos.
- Se convierte el texto a minúsculas.
- Se eliminan las "stop words" (palabras comunes que no aportan mucho significado).
- Se aplica lematización para reducir las palabras a su forma base.

```
import nltk
import re
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from google.colab import drive
```

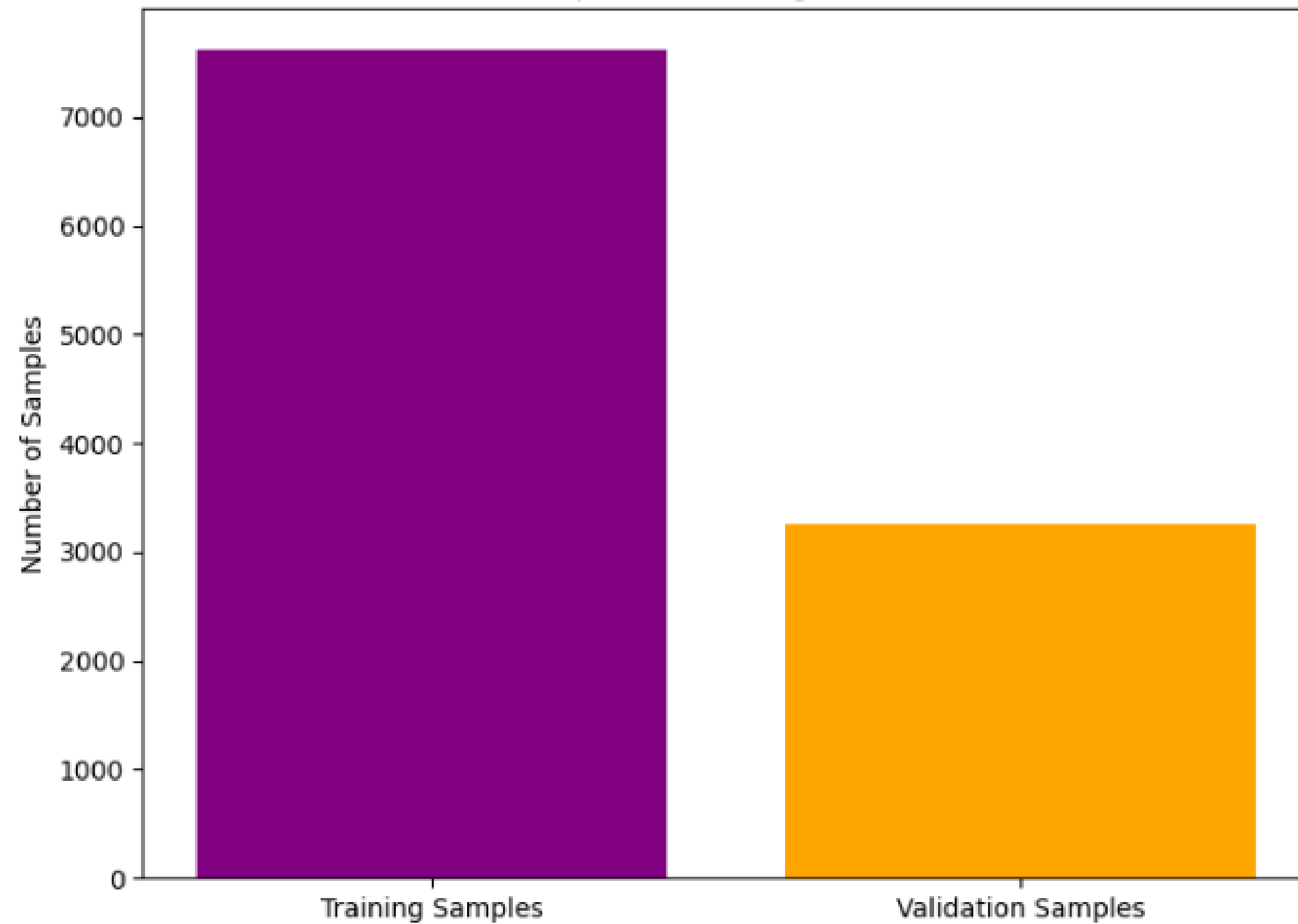
```
0             happened terrible car crash
1 heard earthquake different city stay safe ever...
2 forest fire spot pond goose fleeing across str...
3             apocalypse lighting spokane wildfire
4             typhoon soudelor kill china taiwan
```

```
def preprocess_text(text):
    text = re.sub(r'http\S+', '', text) # Eliminar URLs
    text = re.sub(r'^a-zA-Z\s', '', text) # Eliminar caract
    text = text.lower() # Convertir a minúsculas
    tokens = text.split()
    tokens = [lemmatizer.lemmatize(word) for word in tokens if
    return ' '.join(tokens)
```

Visualización del Dataset



Cantidad de datos de
entrenamiento de tweets de
desastres ciertos y falsos.



¿ Qué es TF id Vectorizer ?

TF-IDF Vectorizer (Term Frequency-Inverse Document Frequency Vectorizer) es una medida de feature weighting que expresa lo relevante que es una palabra en un documento, el cual forma parte de un corpus

Term Frequency X **Inverse Document Frequency**

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

Text1: Basic Linux Commands for Data Science

Text2: Essential DVC Commands for Data Science

	basic	commands	data	dvc	essential	for	linux	science
Text 1	0.5	0.35	0.35	0.0	0.0	0.35	0.5	0.35
Text 2	0.0	0.35	0.35	0.5	0.5	0.35	0.0	0.35

¿Cómo funciona el TF-IDF Vectorizer?

El tf idf vectorizer python tiene en cuenta el número de veces que aparece la palabra (o token) en dicho documento, pero también el total de veces que aparece en todo el corpus.

- Los tokens muy frecuentes a nivel de documento y de corpus (posibles stopwords) obtendrán un valor de TF-IDF Vectorizer bajo.
- Los tokens que aparecen solo en ciertos documentos del corpus tendrán un IDF mayor que aquellos que aparecen en mayor número de documentos.

De modo que:

$$w_{x,y} = \text{tf}_{x,y} \times \log \left(\frac{N}{\text{df}_x} \right)$$

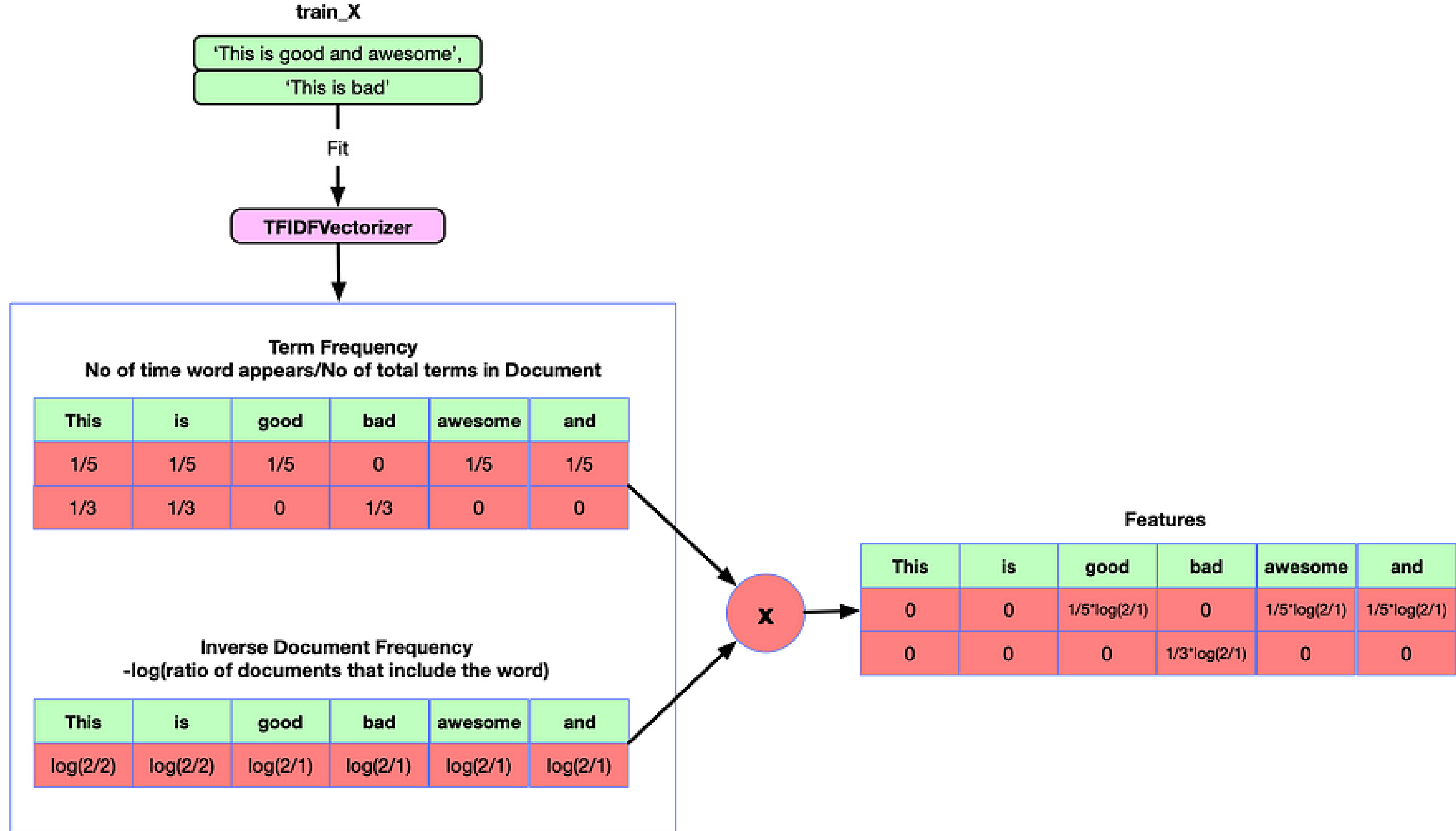
TF-IDF → término x sin documento y

tf_{x,y} = frecuencia de x en y

df_x = número de documentos que contienen x

N = número total de documentos

Diagrama de vectorización



Vectorización con tfidfvectorizer

Dado que los algoritmos de aprendizaje automático requieren entradas numéricas, y los datos con los que trabajamos son texto (los tweets), se empleó la técnica TF-IDF (implementada a través de la librería TfidfVectorizer de scikit-learn) para transformar los textos en vectores numéricos que los modelos puedan procesar.

```
# Vectorización
vectorizer = TfidfVectorizer(max_features=5000)
X = vectorizer.fit_transform(train_df['text_cleaned'])
y = train_df['target']
X_test = vectorizer.transform(test_df['text_cleaned'])

# Imprimir información sobre la matriz resultante
print("Forma de la matriz X (entrenamiento):", X.shape)
print("Forma de la matriz X_test (prueba):", X_test.shape)
print("\nRepresentación vectorial del primer tweet (en formato CSR):")
print(X[0])

# Para ver los valores no nulos de la matriz dispersa del primer tweet:
print("\nValores no nulos del primer tweet:")
first_tweet_vector = X[0]
for col_index, value in zip(first_tweet_vector.indices, first_tweet_vector.data):
    feature_name = vectorizer.get_feature_names_out()[col_index]
    print(f"Palabra '{feature_name}': {value:.4f}")
```

Forma de la matriz X (entrenamiento): (7613, 5000)

Forma de la matriz X_test (prueba): (3263, 5000)

Representación vectorial del primer tweet (en formato CSR):

<Compressed Sparse Row sparse matrix of dtype 'float64'
with 4 stored elements and shape (1, 5000)>

Coords	Values
(0, 3319)	0.5009293110833054
(0, 1321)	0.4695465692508712
(0, 2529)	0.42351661684532593
(0, 113)	0.5909564449756272

Valores no nulos del primer tweet:

Palabra 'reason': 0.5009

Palabra 'earthquake': 0.4695

Palabra 'may': 0.4235

Palabra 'allah': 0.5910

Parámetros aplicados

max_features: int, default=Ninguno

Si no es Ninguno, construya un vocabulario que solo considere la parte superior ordenada por frecuencia de términos en todo el corpus. De lo contrario, se utilizan todas las funciones.max_features

Este parámetro se ignora si el vocabulario no es None.

```
vectorizer = TfidfVectorizer(  
    max_features=1_000_000,  
    ngram_range=(1, 4),  
    sublinear_tf=True)
```

ngram_range: tupla (min_n, max_n), default=(1, 1)

El límite inferior y superior del rango de valores n para diferentes n-gramas a extraer. Todos los valores de n tales que min_n <= n <= max_n se utilizará. Por ejemplo, un de significa solo unigramas, significa unigramas y bigramas, y significa solo bigramas. Solo se aplica si no se puede llamar.ngram_range(1, 1)(1, 2)(2, 2)analyzer

sublinear_tf: bool, default=Falso

Aplique una escala tf sublineal, es decir, reemplace tf por 1 + log(tf).

Funcionamiento de los modelos

Naive Bayes Multinomial

Probabilístico

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- $P(A|B)$: probabilidad de que ocurra A sabiendo que B ya ha ocurrido.
- $P(B|A)$: probabilidad de que ocurra B sabiendo que se ha dado A.
- $P(A)$: probabilidad de que ocurra A.
- $P(B)$: probabilidad de que ocurra B.

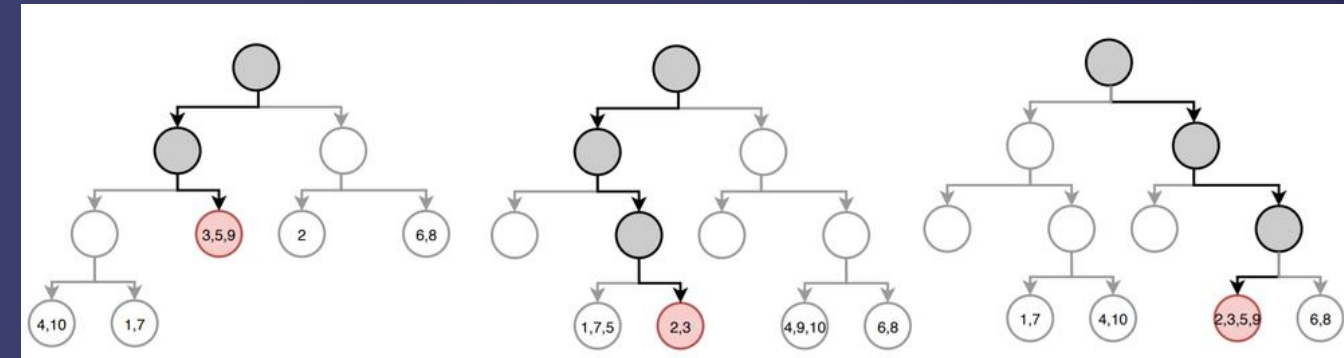
Regresión Logística

Estadístico

$$f(x) = \frac{1}{1 + e^{-x}}$$

- 1: Son los números uno utilizados en la fórmula matemática.
- e: Es la base del logaritmo natural
- - (signo menos): Indica la negación del valor de x en el exponente.

Random Forest



$$\hat{y}_{arbol_1} = \frac{24 + 2 + 20}{3} = 15.33333$$

$$\hat{y}_{arbol_2} = \frac{18 + 24}{2} = 21$$

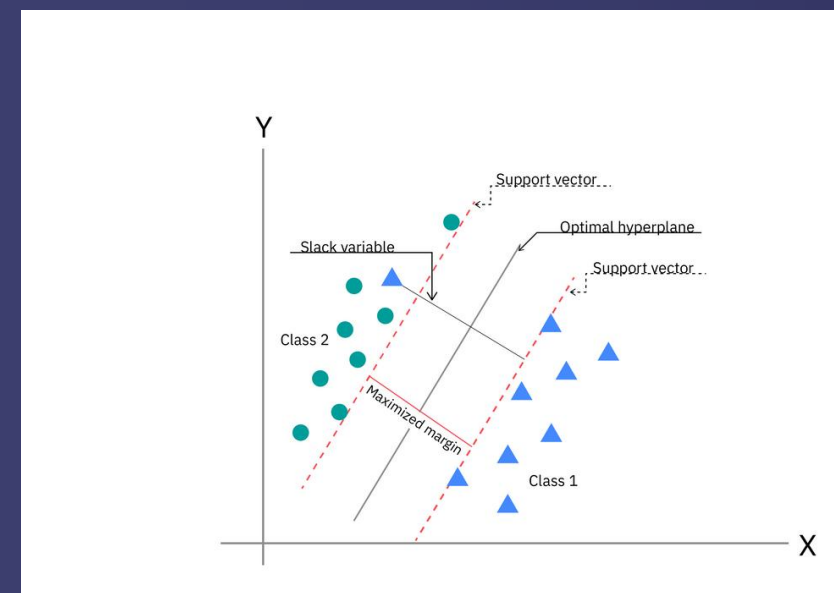
$$\hat{y}_{arbol_3} = \frac{18 + 24 + 2 + 20}{4} = 16$$

Estadístico de aprendizaje supervisado

La predicción final del modelo es la media de todas las predicciones individuales:

$$\hat{\mu} = \frac{15.33333 + 21 + 16}{3} = 17.4$$

Vector Machine



- Datos de Entrada (x): Los puntos que queremos clasificar.
- Vector de Pesos (w) e Intercepto (b): Definen el hiperplano de decisión.
- Margen: La zona de separación maximizada entre las clases.
- Vectores de Soporte: Los puntos cruciales que definen el margen.
- Slack Variables (ξ): Permiten errores de clasificación para datos no linealmente separables.

Naive Bayes Multinomial

```
] X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
model = MultinomialNB()
model.fit(X_train, y_train)
```

	precision	recall	f1-score	support
0	0.78	0.90	0.84	874
1	0.84	0.66	0.74	649
accuracy			0.80	1523
macro avg	0.81	0.78	0.79	1523
weighted avg	0.81	0.80	0.80	1523

F1 SCORE
0.7483333333333333

Regresión Logística

```
# Modelo Logistic Regression
model = LogisticRegression(random_state=42, max_iter=1000)
model.fit(X_train, y_train)
```

	precision	recall	f1-score	support
0	0.79	0.89	0.84	874
1	0.83	0.68	0.75	649
accuracy			0.80	1523
macro avg	0.81	0.79	0.79	1523
weighted avg	0.81	0.80	0.80	1523

F1 SCORE
0.7472527472527473

Random Forest

```
# Modelo Random Forest
model = RandomForestClassifier(n_estimators=100, random_state=42)
# Ajusta n_estimators según sea necesario
model.fit(X_train, y_train)
```

	precision	recall	f1-score	support
0	0.77	0.84	0.80	874
1	0.75	0.67	0.71	649
accuracy			0.77	1523
macro avg	0.76	0.75	0.76	1523
weighted avg	0.76	0.77	0.76	1523

F1 SCORE
0.7259615384615384

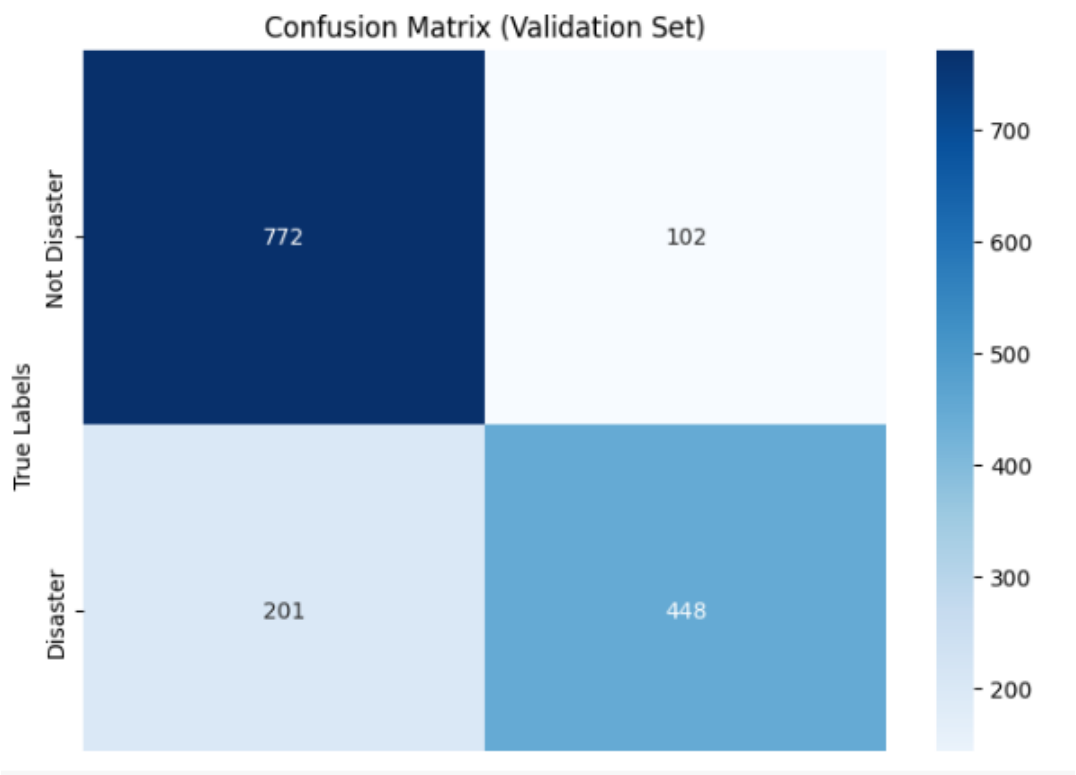
Support Vector Machine

```
# Modelo SVM
model = SVC(random_state=42)
model.fit(X_train, y_train)
```

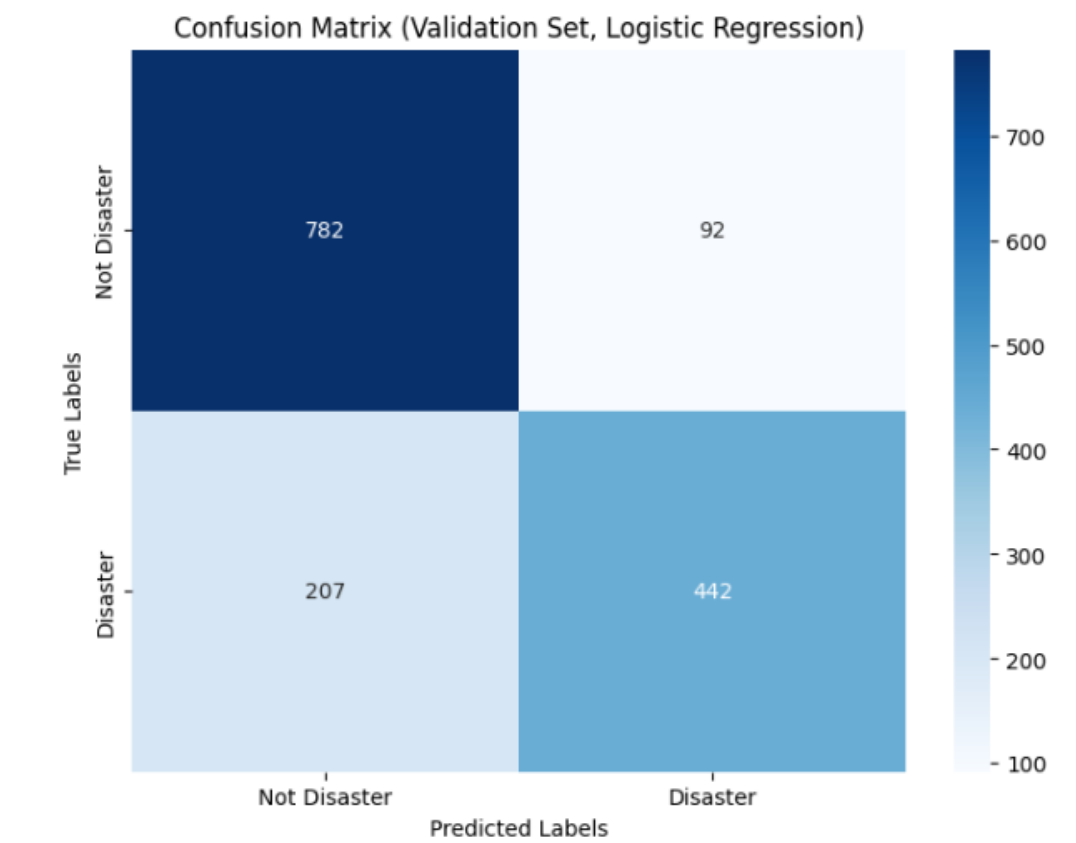
	precision	recall	f1-score	support
0	0.78	0.90	0.83	874
1	0.82	0.66	0.73	649
accuracy			0.80	1523
macro avg	0.80	0.78	0.78	1523
weighted avg	0.80	0.80	0.79	1523

F1 SCORE
0.7339606501283148

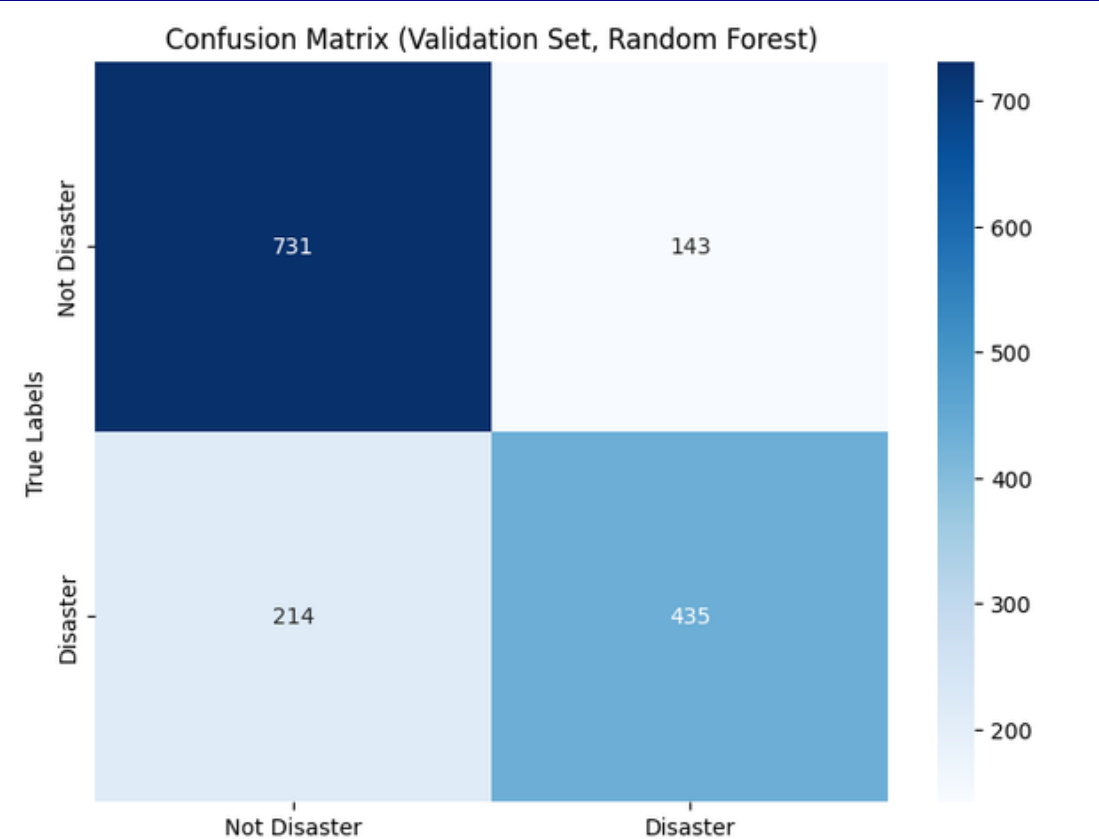
Naive Bayes Multinomial



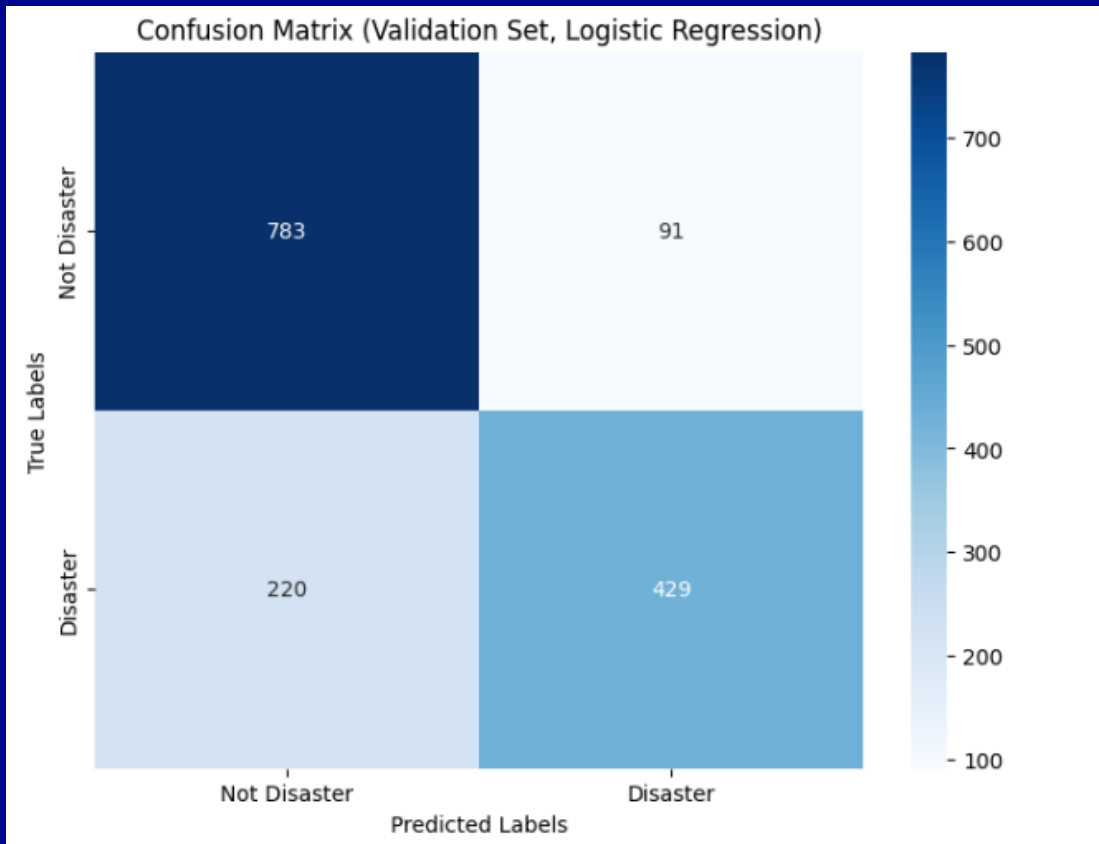
Regresión Logística



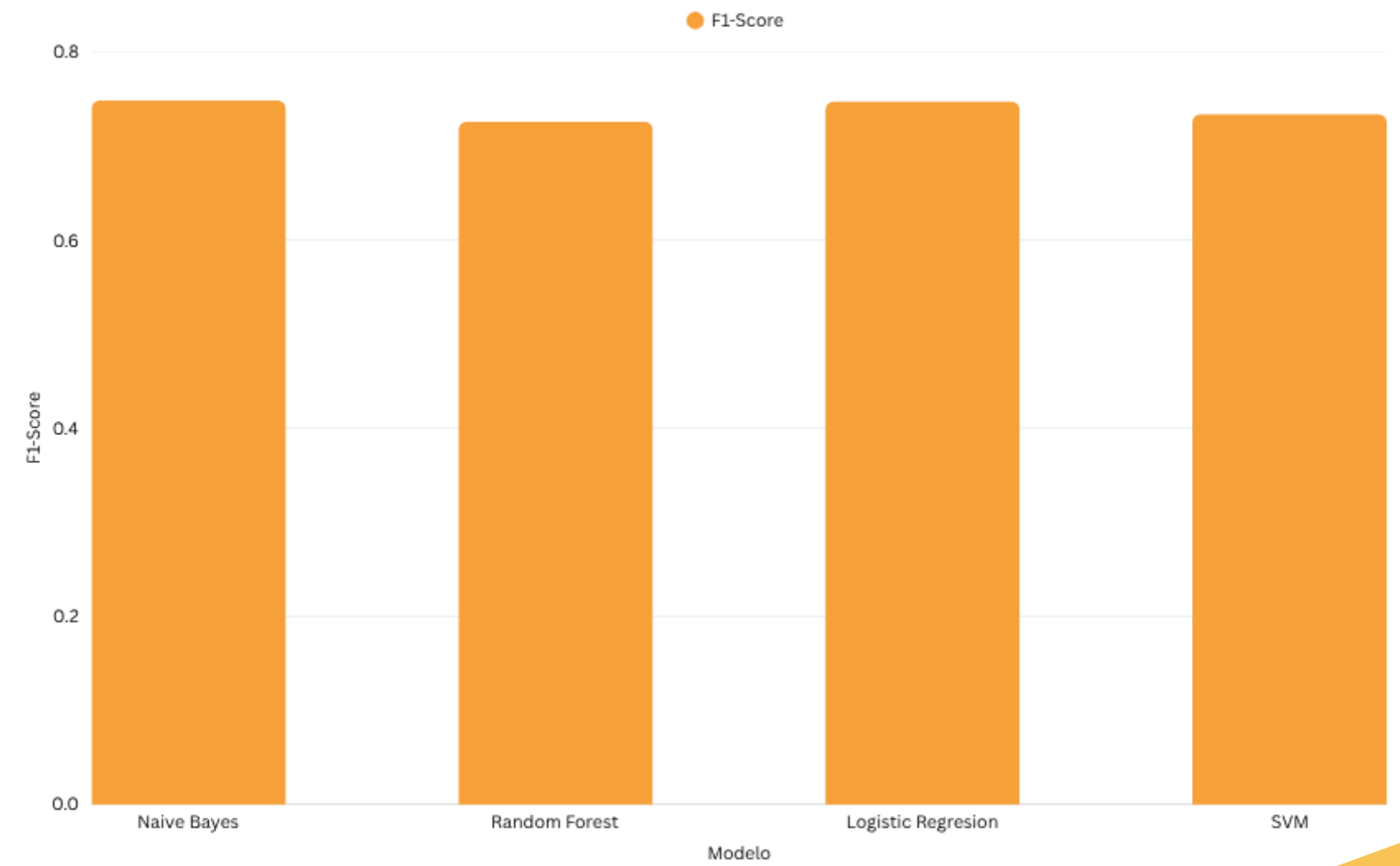
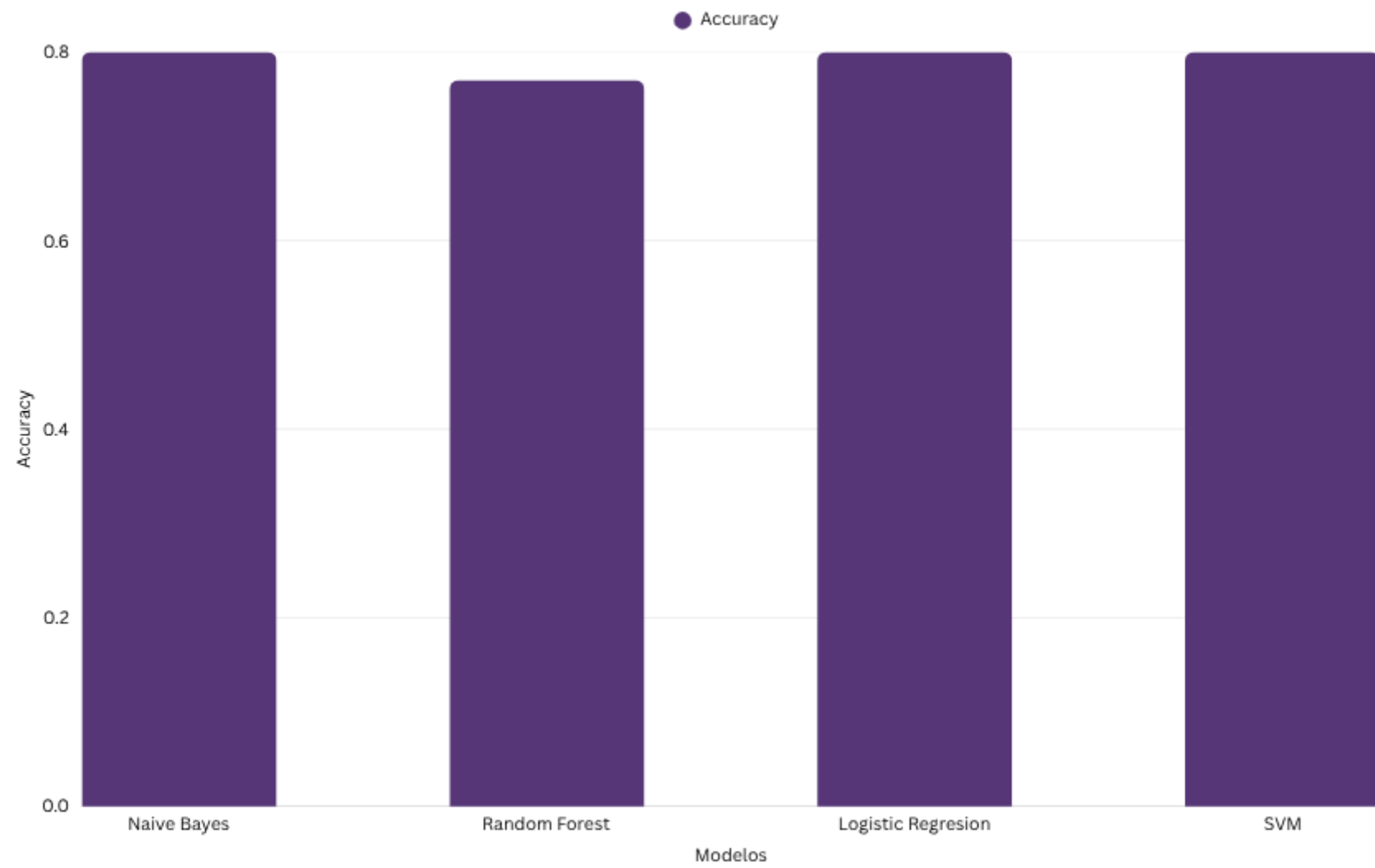
Random Forest



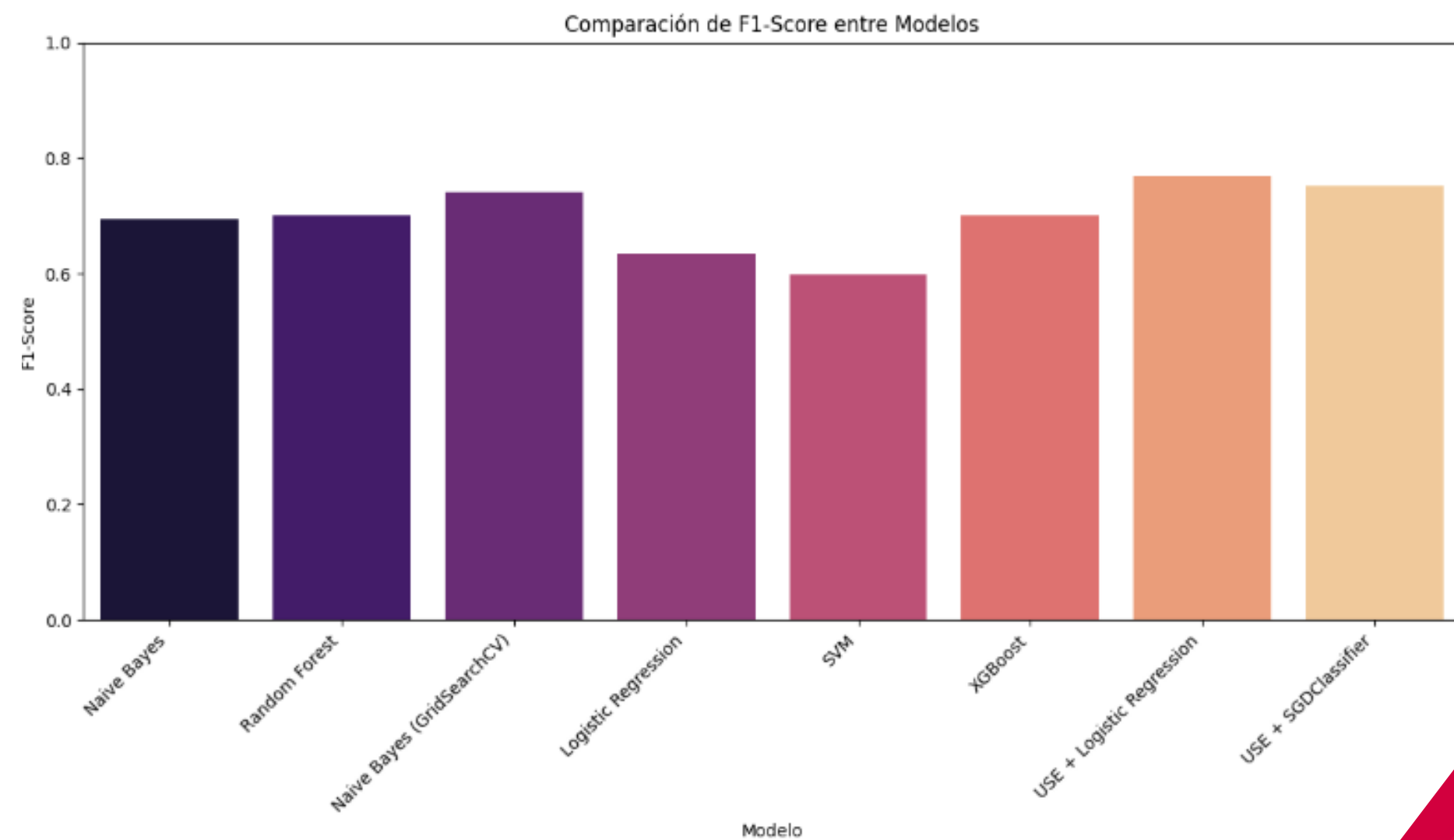
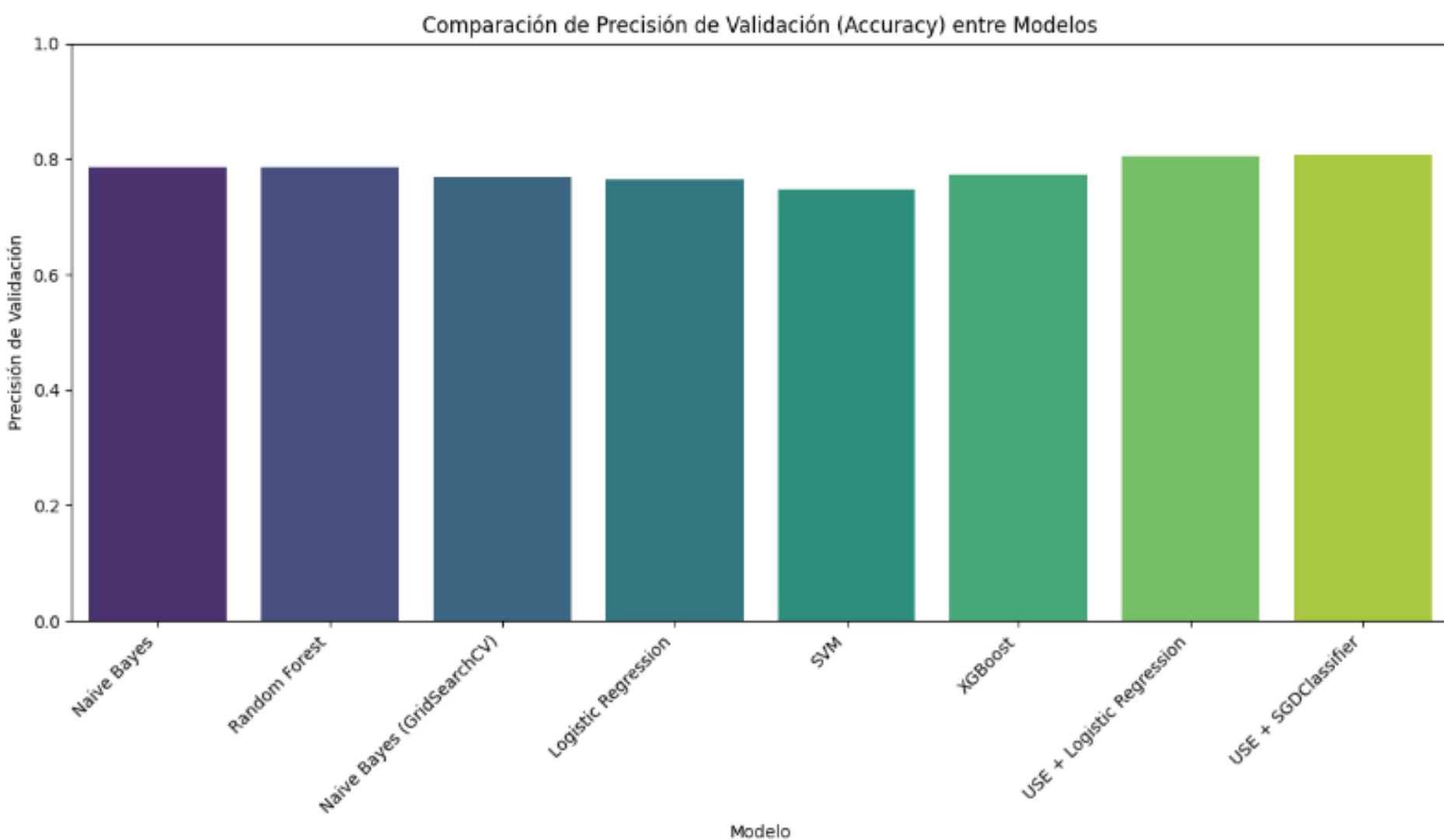
Vector Machine



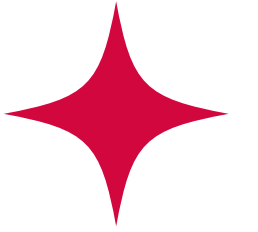
Comparando Modelos



Profundizando: Más Modelos de IA



Sesgos



Falsos positivos

--- Ejemplos de Falsos Positivos (Predicción: Desastre, Real: No Desastre) ---

Tweet Original: Who is bringing the tornadoes and floods. Who is bringing the climate change. God is after America He is plaguing her

#FARRAKHAN #QUOTE

Tweet Limpio: bringing tornado flood bringing climate change god america plaguing farrakhan quote

Tweet Original: she's a suicide bomb

Tweet Limpio: shes suicide bomb

Tweet Original: When your heart is bigger than the obstacles in front of you #euro #dontexpectnothing #july #fire @euro

Tweet Limpio: heart bigger obstacle front euro dontexpectnothing july fire euro

Fasos negativos

--- Ejemplos de Falsos Negativos (Predicción: No Desastre, Real: Desastre) ---

Tweet Original: So you have a new weapon that can cause un-imaginable destruction.

Tweet Limpio: new weapon cause unimaginable destruction

Tweet Original: DT @georgegalloway: RT @Galloway4Mayor: @ÜïThe Col police can catch a pickpocket in Liverpool Stree... <http://t.co/vXIn1gOq4Q>

Tweet Limpio: dt georgegalloway rt gallowaymayor col police catch pickpocket liverpool stree

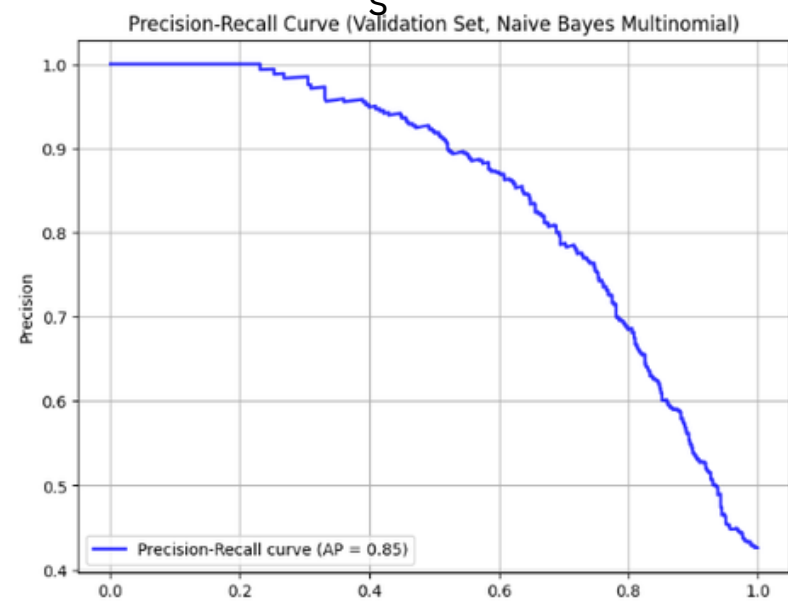
Tweet Original: my favorite lady came to our volunteer meeting

hopefully joining her youth collision and i am excite <http://t.co/Ij0wQ490cS>

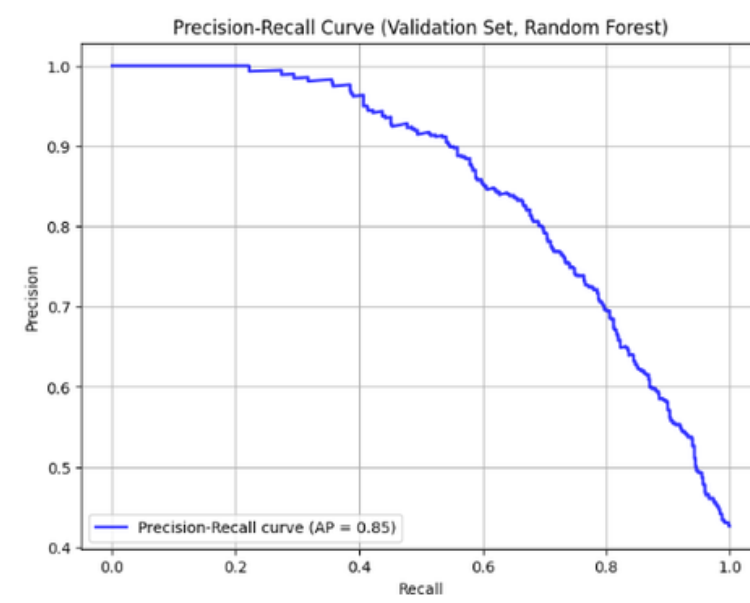
Tweet Limpio: favorite lady came volunteer meeting hopefully joining youth collision excite

Grafica: falsos positivos y falsos negativos

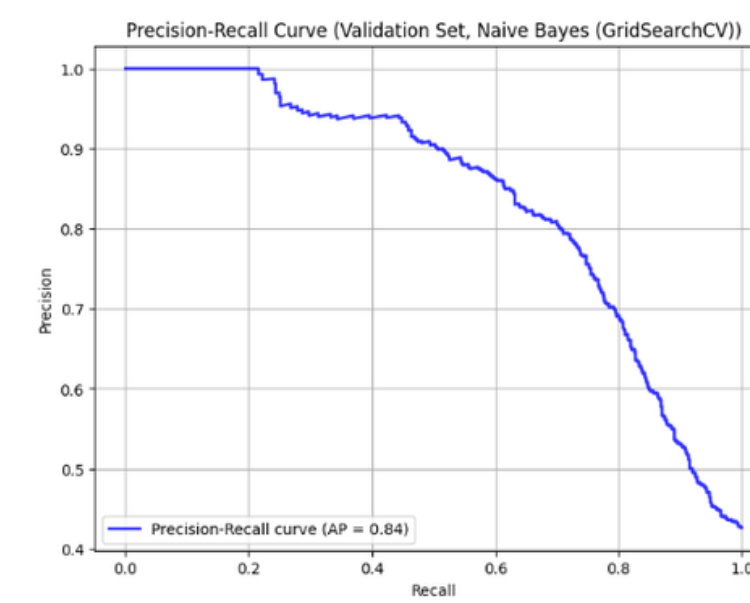
nayve_baye



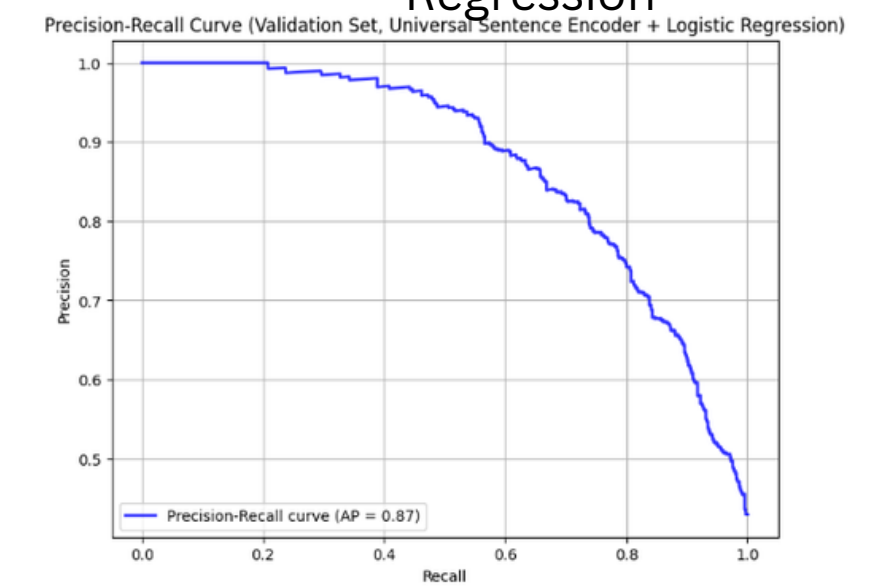
random_forest



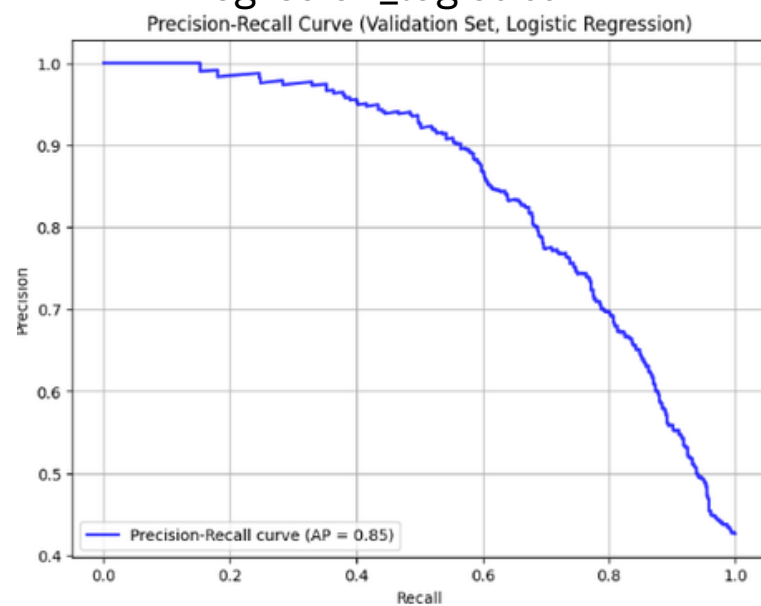
naive_bayes_gridsearch



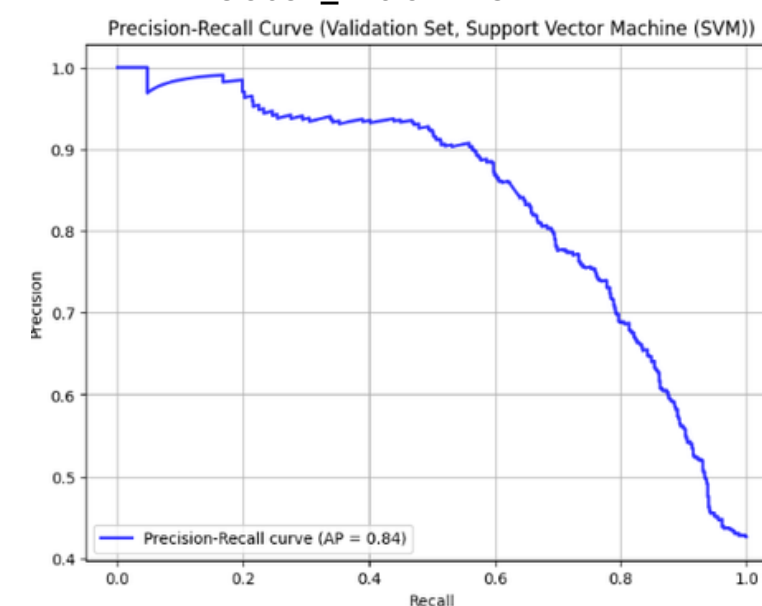
Universal Sentence Encoder + Logistic Regression



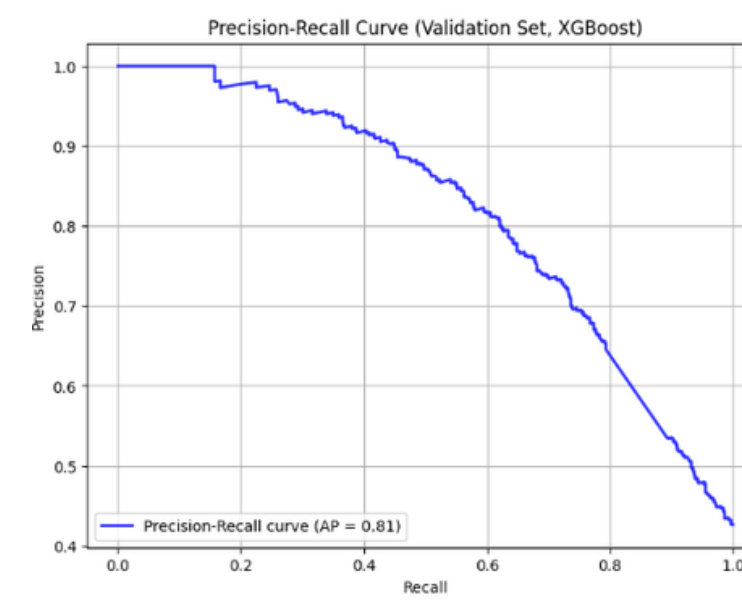
regresion_logistica



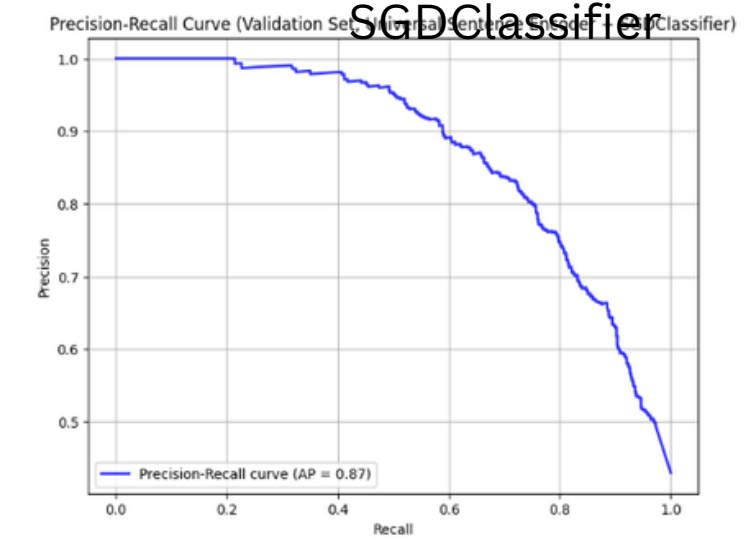
vector_machine



XGBoost

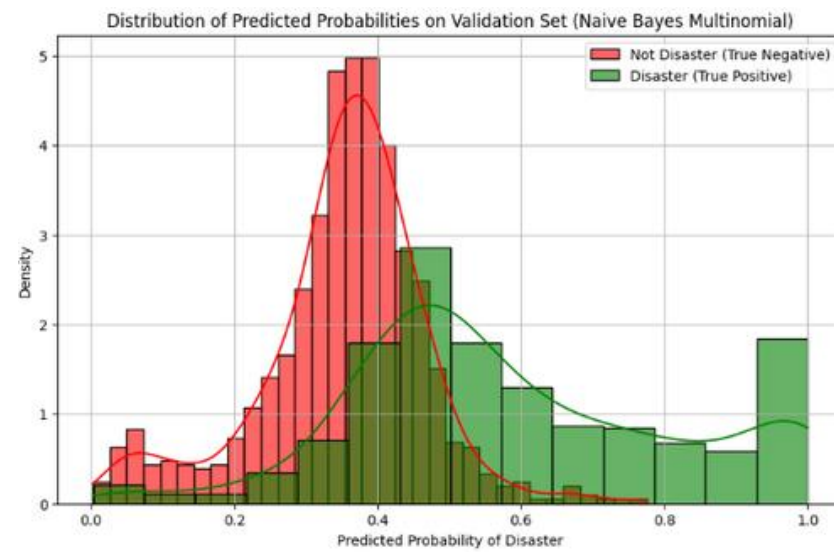


Universal Sentence Encoder + SGDClassifier

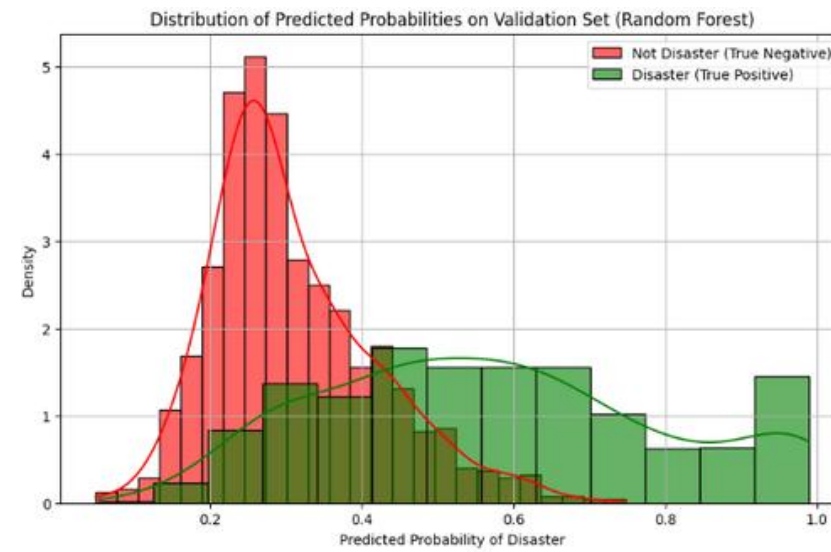


Histograma: Falsos positivos y falsos negativos

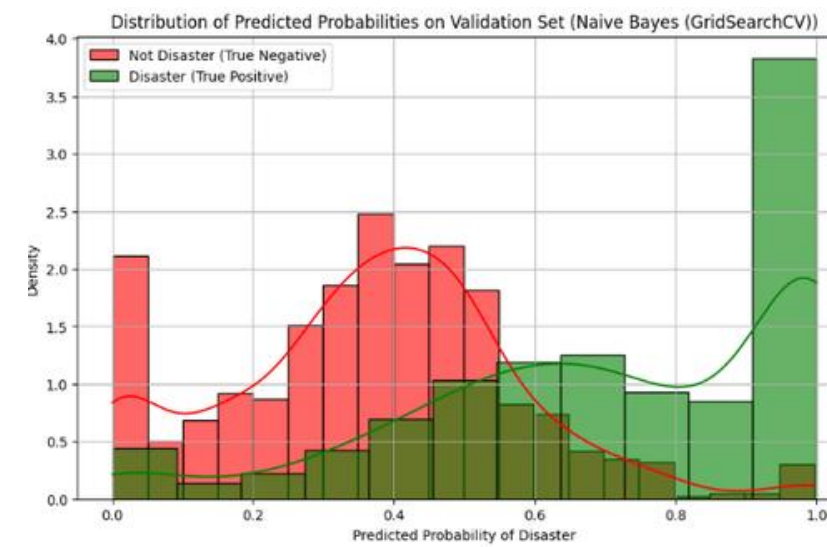
naive_bayes



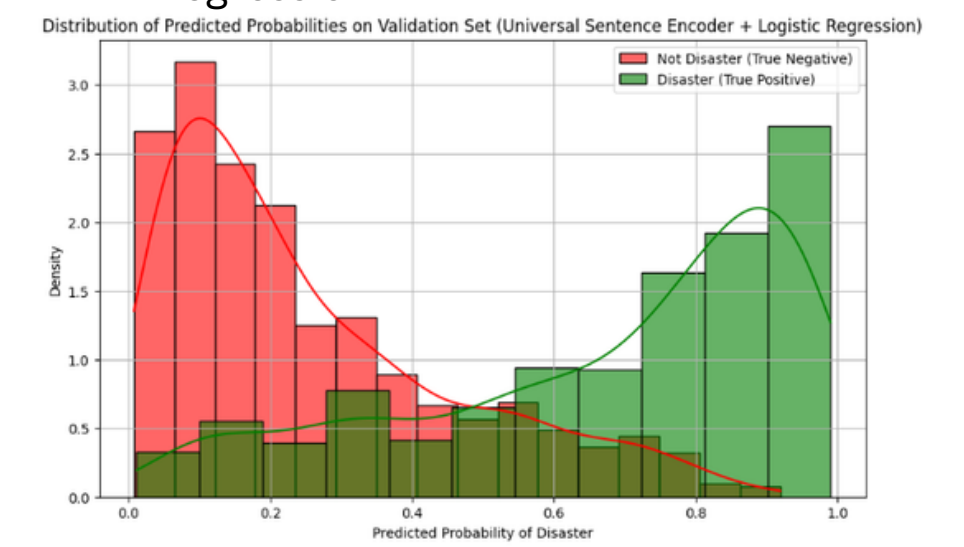
random_forest



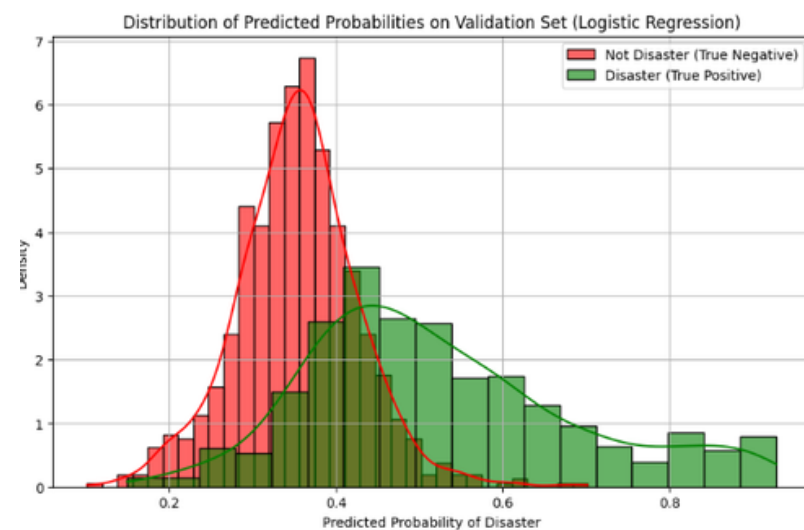
naive_bayes_gridsearch



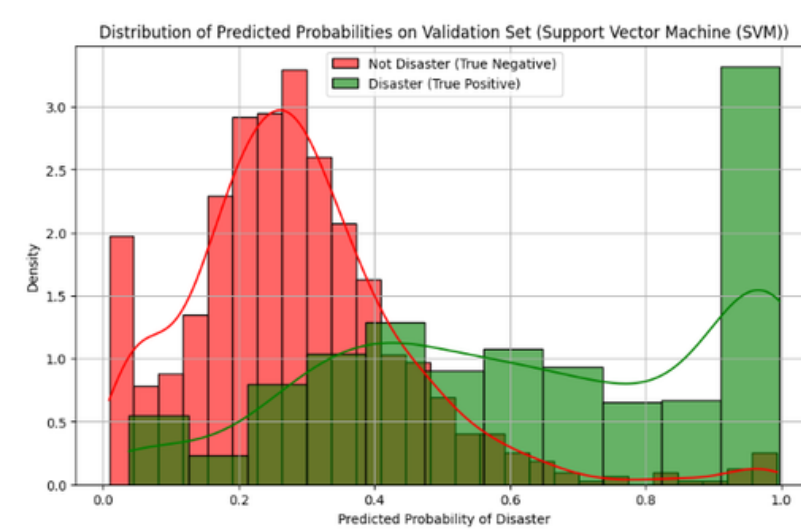
Universal Sentence
Encoder + Logistic
Regression



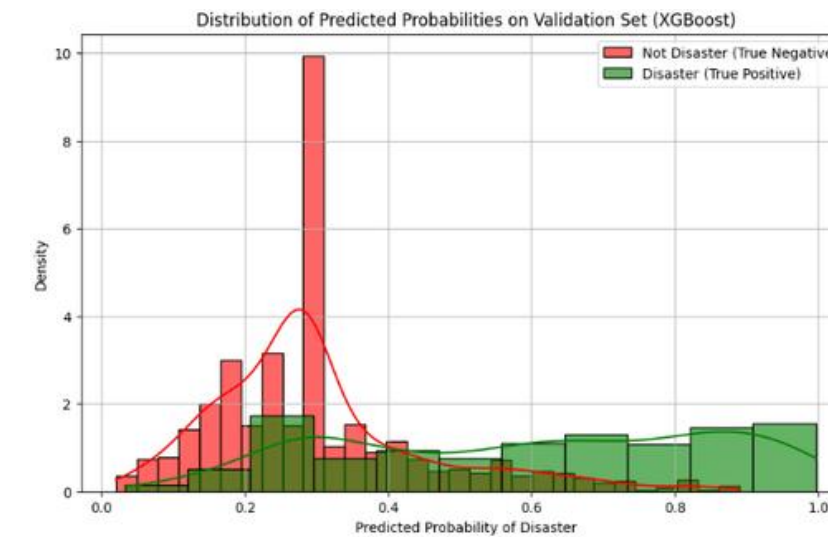
regresion_logistica



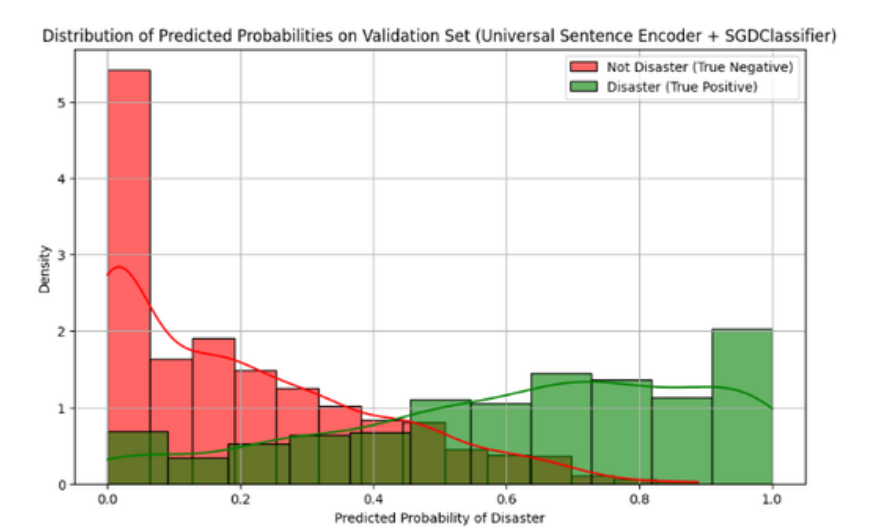
vector_machine



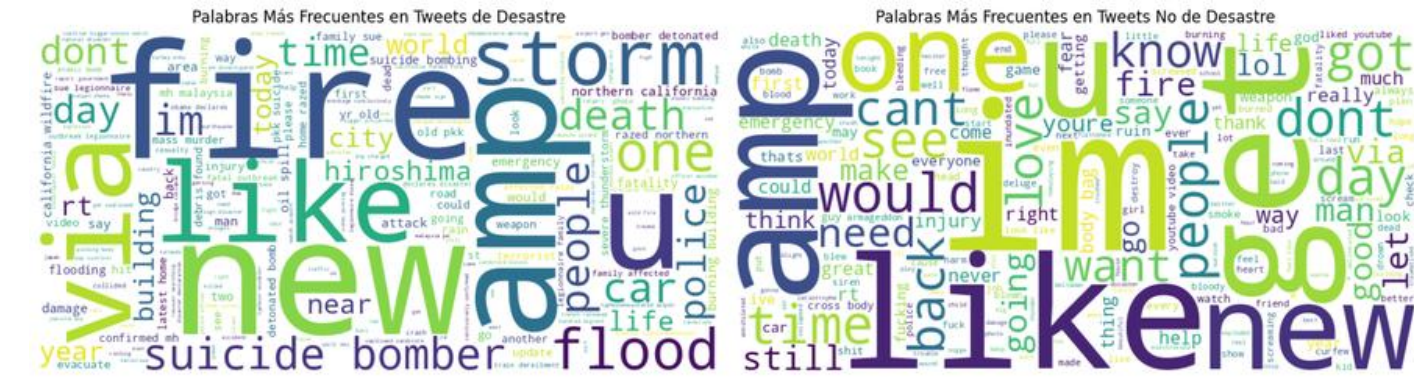
XGBoost



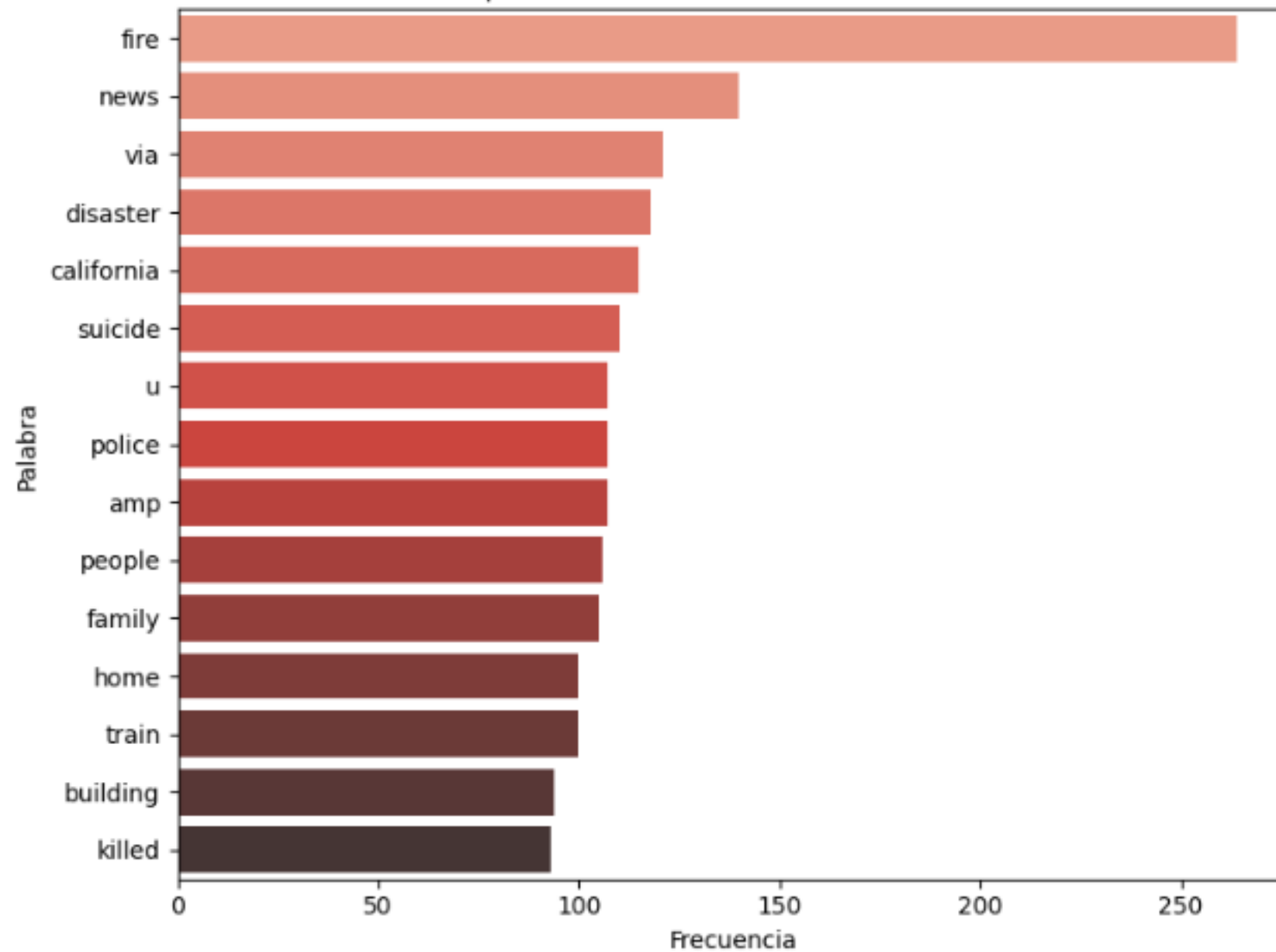
Universal Sentence
Encoder + SGDClassifier



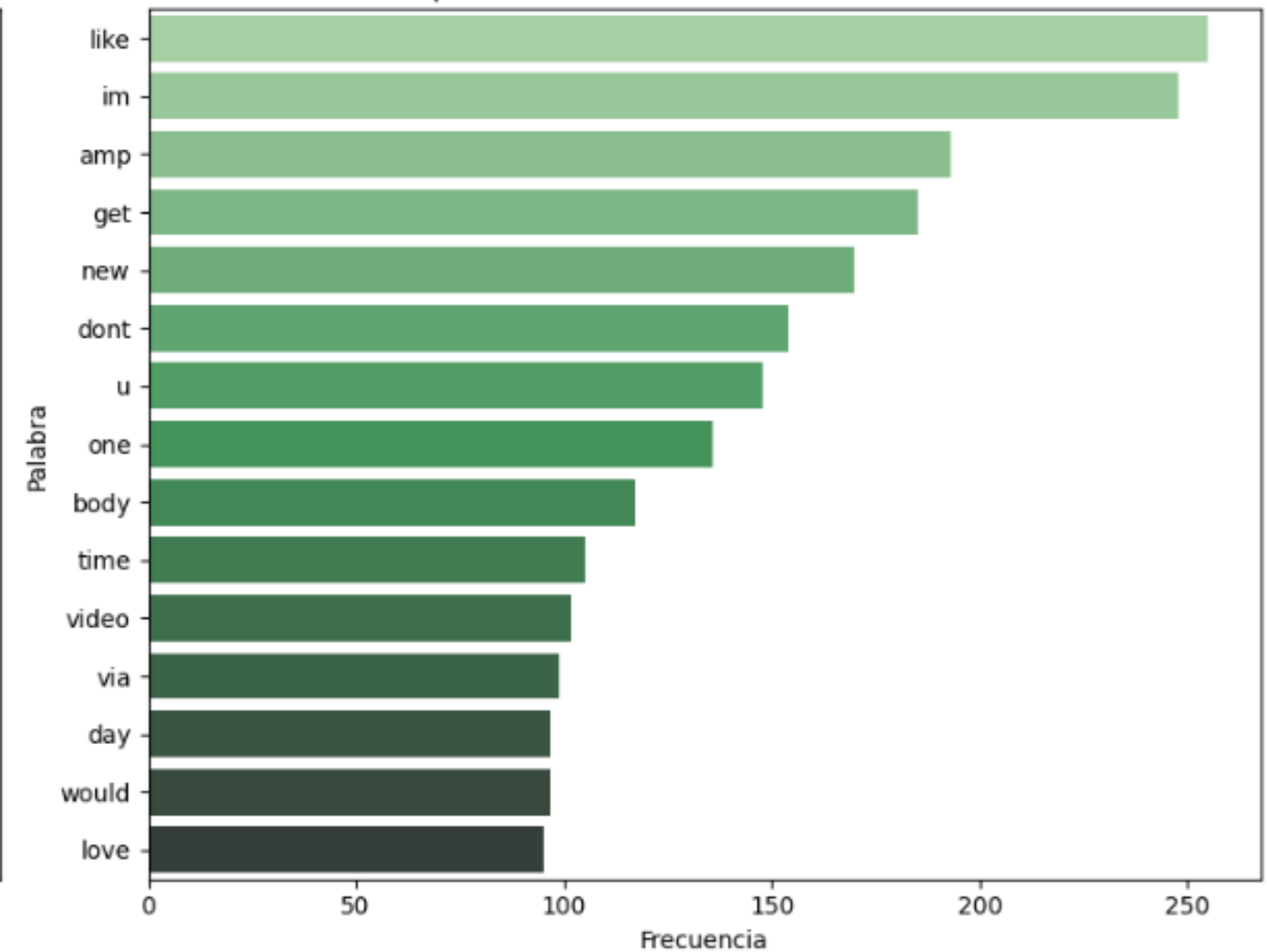
Palabras mas frecuentes



Top 15 Palabras en Tweets de Desastre



Top 15 Palabras en Tweets No de Desastre



Evaluación en Kaggle

Procesamiento del Lenguaje Natural con Tweets de Desastres			Enviar predicción	...
Visión general			Datos	Código
Modelos			Discusión	Tabla de clasificación
Reglas			Equipo	Presentaciones
Presentación y descripción			Puntuación pública ⓘ	
✓	submission_XGBoost.csv	Completo · Hace 44 segundos	0.76984	
✓	submission_vector_machine.csv	Completo · hace 1m	0.79528	
✓	submission_regresion_logistica.csv	Completo · hace 2m	0.79681	
✓	submission_naive_bayes_gridsearch.csv	Completo · hace 2m	0.79865	
✓	submission_random_forest.csv	Completo · hace 3m	0.79007	
✓⌚	submission.csv	Completo · hace 2 meses	0.79007	

(Ranking 858 de 956 participantes)

Conclusión

Este modelo implementa modelo Naive Bayes Multinomial para clasificar los tweets, utilizando grades conjuntos de datos de entrenamiento y validación teniendo en cuenta que estos datos deben ser limpiados y optimizados para su efectivo aprendizaje. Los resultados obtenidos muestran que este modelo de IA presenta alta precisión en la clasificación de este tipo de tweets.



! Muchas Gracias i

