

cedia



research
lab

by cedia

**Redacción y diseño de
los casos de uso de las
herramientas HPC, Indata,
GitHub y Mattermost**

—
CASO DE
USO 1

cedia

*“Conectando ideas,
transformando
sociedades”*





Almacena y comparte
datos de investigación.



Usa computación
de alto rendimiento.



Colabora en proyectos de
investigación de manera
efectiva.

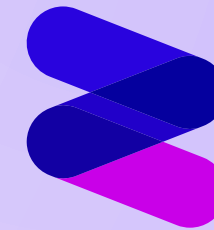


Envía archivos
grandes de forma
segura.

*Mejorará la
reproducibilidad
y eficiencia en
tu investigación*



*Una Plataforma que revolucionará
la Investigación Científica en Ecuador*



**research
lab**
by cedia

Redacción y diseño de los casos
de uso de las herramientas HPC,
Indata, GitHub y Mattermost

CASO DE USO 1

CONTENIDO

CASO DE USO 1

1.1

Introducción

pp — 08

1.2



Uso de la
herramienta
HPC

pp — 10

1.3



Uso de la
herramienta
Indata

pp — 20

1.4



Uso de la
herramienta
Gitlab

pp — 46

1.5



Uso de la
herramienta
Mattermost

pp — 60

1.6



Uso de la
herramienta
Filesender

pp — 72

CASO DE USO 1

Modelo de aprendizaje automático para la detección de ransomware.

1.1

INTRODUCCIÓN

El ransomware es un tipo de malware que puede presentarse como un archivo ejecutable (.exe), troyano, o a través de URLs maliciosas y técnicas como el phishing. Este software malicioso encripta los datos de la víctima y exige un rescate para su recuperación, lo que lo convierte en una seria amenaza, ya que puede extorsionar a usuarios, interrumpir operaciones y comprometer el acceso a información personal o empresarial. Para enfrentar este desafío en el ámbito de la seguridad informática, se recomienda el uso de técnicas de aprendizaje automático. Estas técnicas permiten analizar patrones de comportamiento, identificar firmas distintivas y anticipar tácticas futuras, proporcionando una solución efectiva para la detección del ransomware.

Para abordar esta problemática, se propone investigar un modelo de entrenamiento o algoritmo que maximice la efectividad en la detección de ataques de ransomware utilizando algoritmos de aprendizaje automático. El proceso incluirá la evaluación de la precisión de los modelos, la comparación entre ellos, la identificación de archivos benignos y maliciosos. Todo esto se llevará a cabo sobre un conjunto de datos específico de ataques de ransomware.

1.2

USO DE LA HERRAMIENTA HPC



HPC

Antes de iniciar la investigación, se realiza una revisión sistemática de la literatura, centrada en la selección de fuentes primarias y secundarias. El objetivo es identificar los métodos y técnicas de Machine Learning más efectivos para la detección y mitigación de ataques de ransomware, así como determinar las características esenciales para su identificación y los atributos más eficaces para este propósito. Con los datos recopilados, se puede seleccionar la herramienta HPC para acceder a Rstudio Server y generar gráficos que reflejen los resultados obtenidos en la revisión sistemática de la literatura.

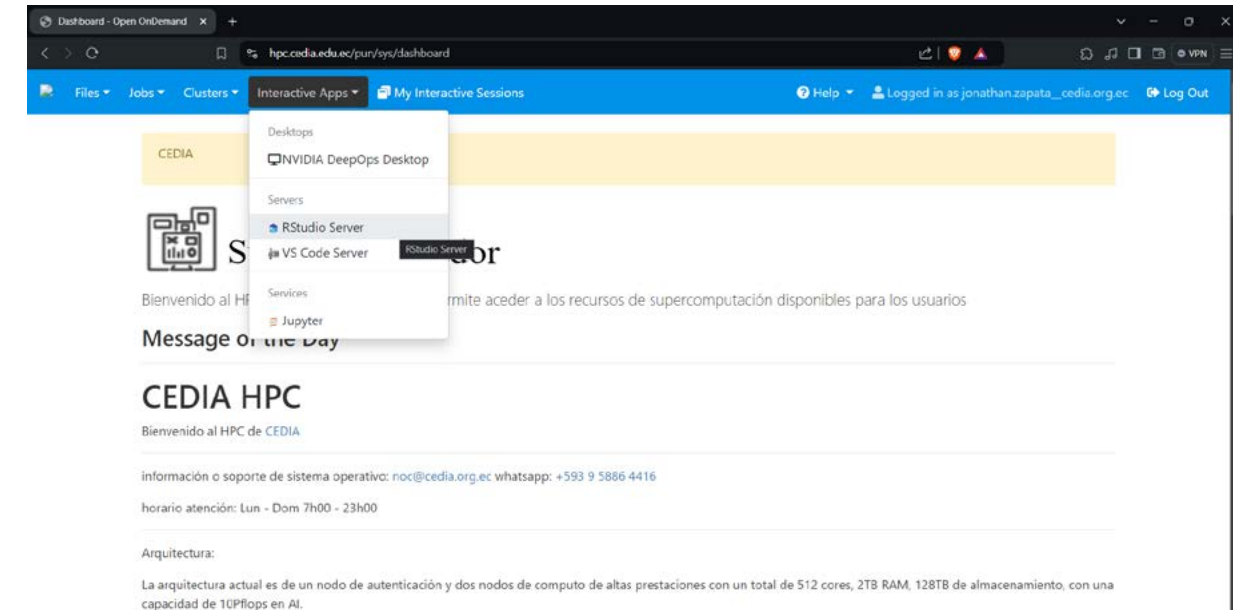


figura 1

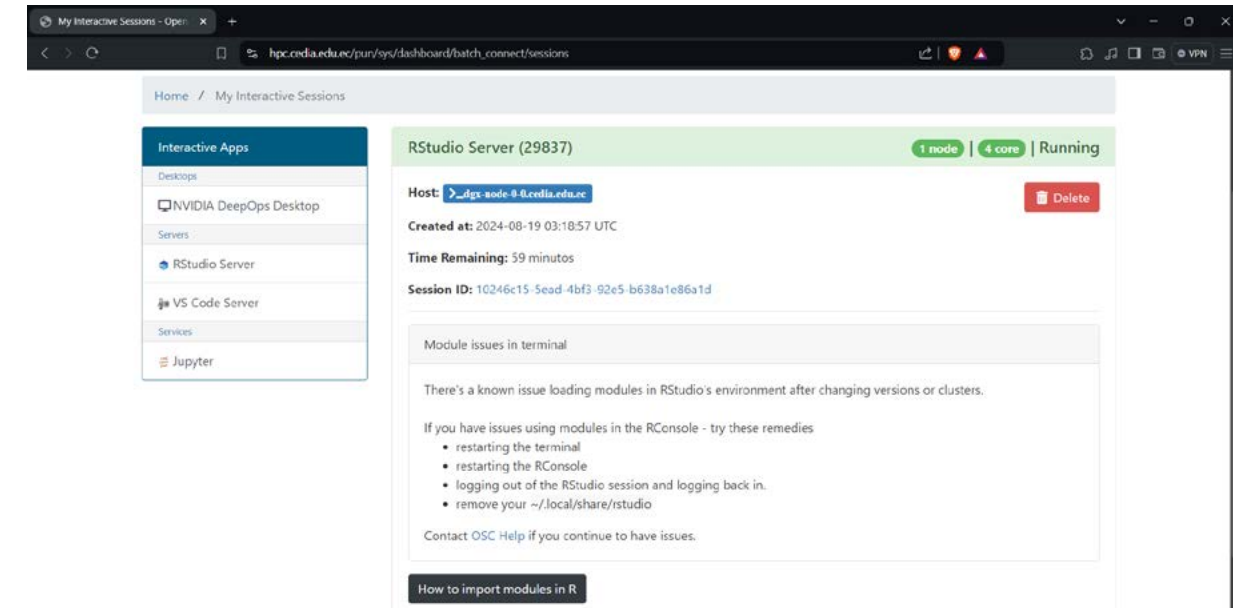


figura 2

En este caso, gracias a la revisión de la literatura, que incluyó el análisis de varios artículos de investigación en distintas base de datos de journals como World Wide Science, Scopus o Sciecn Direct, se identificaron los algoritmos de entrenamiento más utilizados y su frecuencia de uso:

- **Random Forest:** 26 veces
- **SVM:** 15 veces
- **Others:** 16 veces
- **XGBoost:** 5 veces
- **AdaBoost:** 8 veces
- **Gradient Boosting:** 7 veces
- **KNN:** 7 veces
- **Decision Tree:** 15 veces
- **Logistic Regression:** 6 veces
- **Naïve Bayes:** 7 veces
- **Neural Networks:** 2 veces
- **Long Short-term Memory:** 6 veces.

Por ende, vamos a ejecutar el un nuevo Script que nos ayude a representar y graficar de mejor manera estos datos.

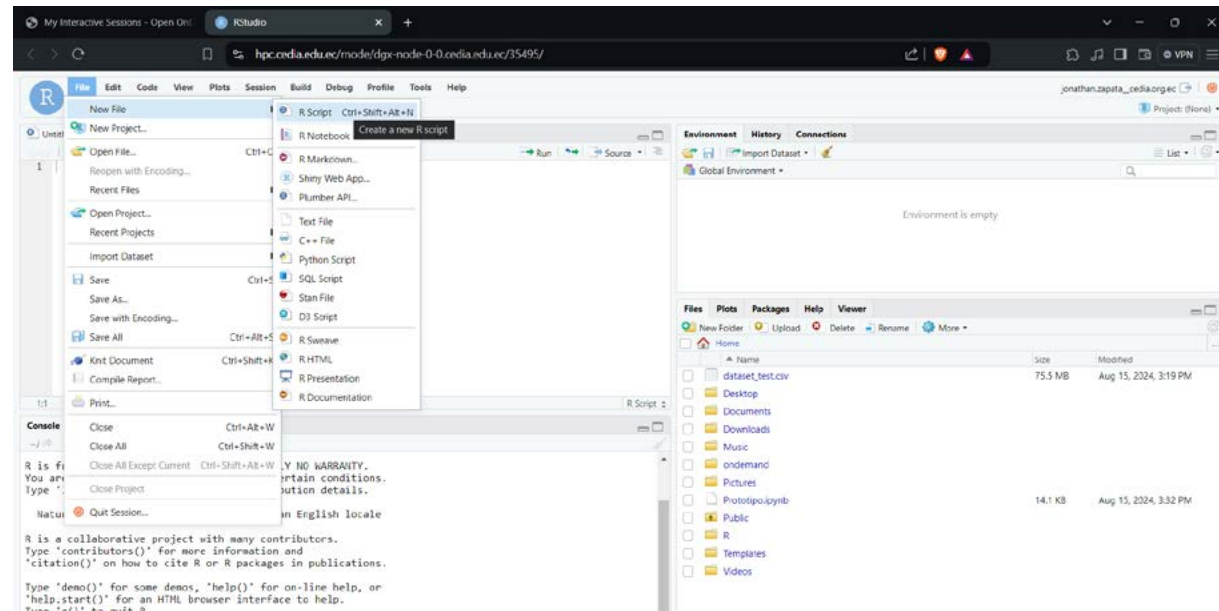


figura 3

Se ejecuto el siguiente Script en R

```
# Install and load the ggplot2 library
install.packages ("ggplot2")
library(ggplot2)

# Manual data
Algorithm <- c("Random Forest", "SVM", "Others", "XGBoost", "AdaBoost",
              "Gradient Boosting", "KNN", "Decision Tree", "Logistic Regression",
              "Naïve Bayes", "Neural Networks", "Long Short-term Memory")
values <- c(26, 15, 16, 5, 8, 7, 7, 15, 6, 7, 2, 6)

# Create a data frame
data <- data.frame(Algorithm, values)

# Calculate percentages
data$percentage <- round((data$values / sum(data$values)) * 100, 1)

# Create the scatter plot with percentages
ggplot(data, aes(x = Algorithm, y = percentage)) +
  geom_point(color = "blue", size = 3) +
  geom_text(aes(label = paste0(percentage, "%")), vjust = -0.5, size = 3.5) +
  labs(
    x = "Algorithm",
    y = "Percentage") +
  scale_y_continuous(expand = expansion(mult = c(0, 0.1))) +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1, size = 12), # Increased size
    axis.text.y = element_text(size = 8),
    axis.title.x = element_text(size = 10),
    axis.title.y = element_text(size = 10),
    plot.title = element_text(size = 12, hjust = 0.5)
```


Y tenemos como resultado el siguiente grafico

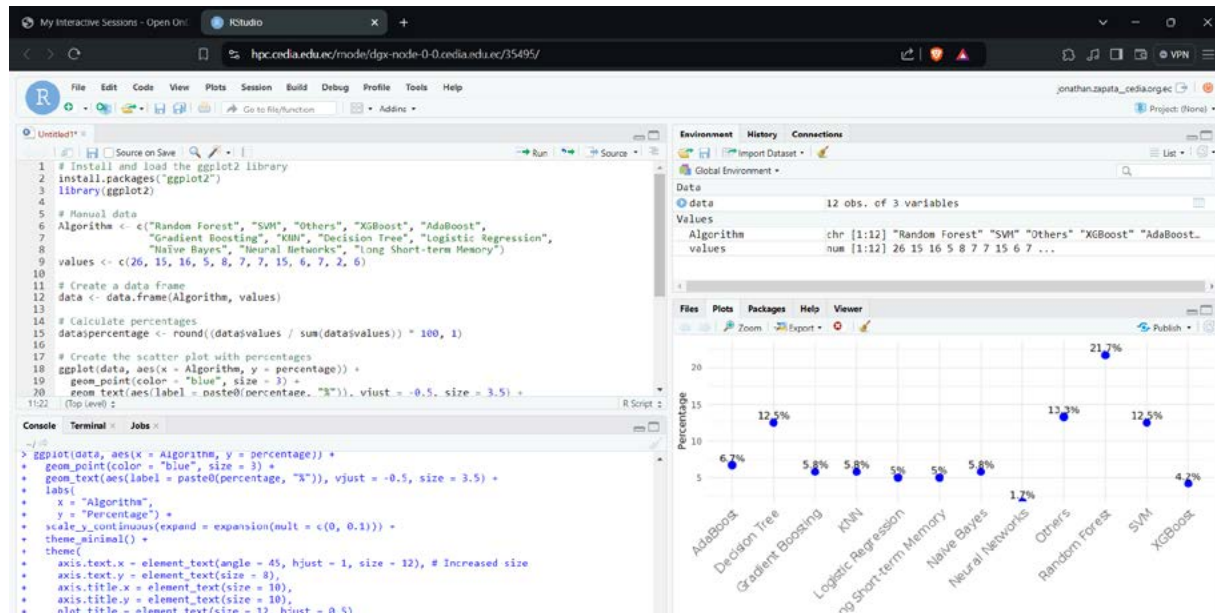


figura 4

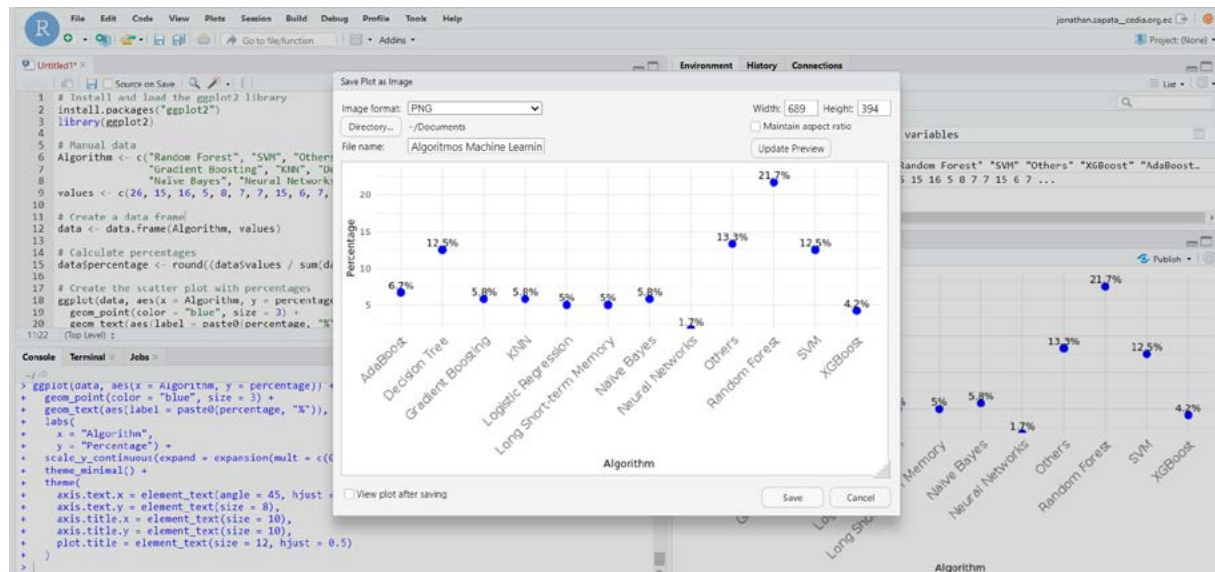


figura 5

Otro ejemplo relevante es la identificación de las técnicas o modelos con mayor frecuencia de uso en la detección de ransomware. Los resultados de la revisión sistemática fueron:

- **Dynamic Analysis:** 12
- **Static Analysis:** 18
- **Hybrid Analysis:** 1
- **Network Traffic:** 5
- **Digital DNA Sequencing:** 1
- **Supervised Learning:** 1

Se ejecutó un nuevo script para mejorar el diseño y la interpretación de los datos. Los resultados obtenidos fueron los siguientes:

```
# Instalar y cargar la librería ggplot2
install.packages("ggplot2")
library(ggplot2)

# Crear un data frame con los datos
datos <- data.frame(
  Techniques = c("Dynamic Analysis", "Static Analysis", "Hybrid Analysis", "Network
Traffic", "Digital DNA Sequencing", "Supervised Learning"),
  Quantity = c( 12, 18, 1, 5, 1, 1)
)

# Crear el gráfico de barras con colores Springer y labels en blanco
ggplot(datos, aes(x = Techniques, y = Quantity, fill = Techniques)) +
  geom_bar(stat = "identity") +

  scale_fill_manual(values = c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442",
"#0072B2", "#D55E00")) + # Colores Springer
  scale_y_continuous(limits = c(0, max(datos$Quantity) + 5)) + # Aumentar la escala del
eje Y
  theme_minimal() +
  theme(
    axis.text.x = element_blank(), # Eliminar el texto del eje X
    axis.title.x = element_blank(), # Eliminar la etiqueta del eje X
    axis.text.y = element_text(size = 12), # Aumentar el tamaño del texto del eje Y
    axis.title.y = element_blank(), # Eliminar la etiqueta del eje Y
    plot.title = element_text(size = 14, hjust = 0.5) # Centrar y aumentar el tamaño del
título del gráfico
  ) +
  labs()
```

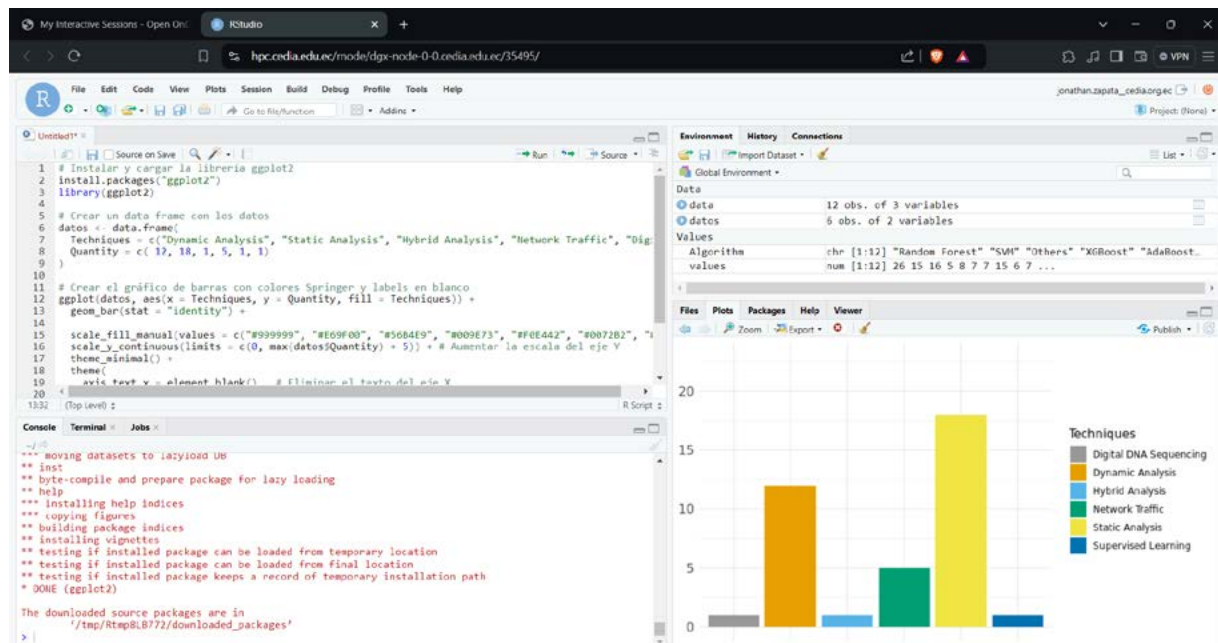


figura 6

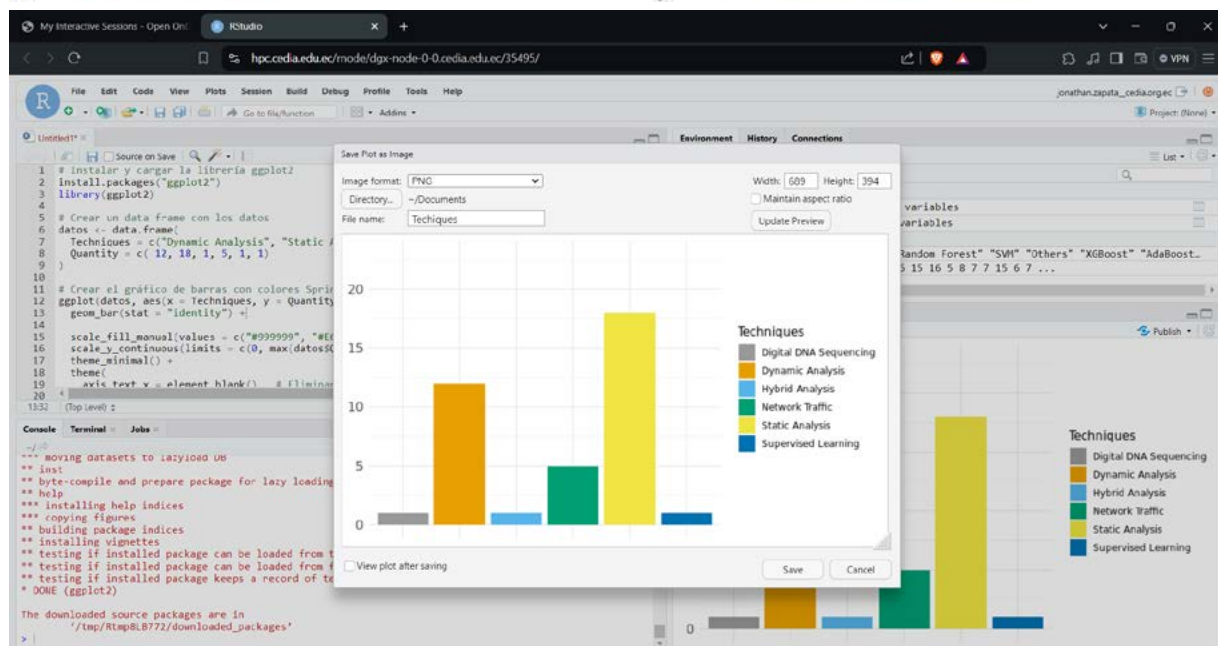


figura 7

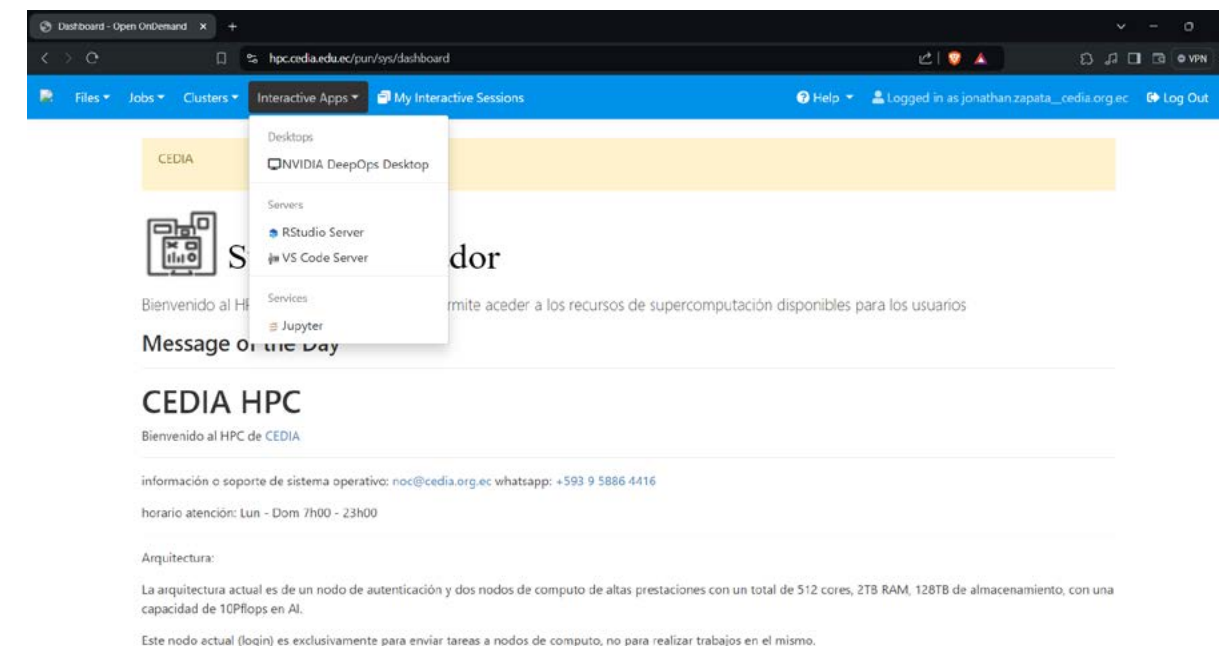


figura 8



Configuramos las opciones al momento de crear Jupyter en este caso se va a procesar mas de 200.000 datos por ese motivo se realiza las configuraciones

Jupyter - Open OnDemand

hpc.cedia.edu.ec/pur/sys/dashboard/batch_connect/sys/jupyter/session_contexts/new

Files Jobs Clusters Interactive Apps My Interactive Sessions Help Logged in as jonathan.zapata@cedia.org.ec Log Out

CEDIA

Home / My Interactive Sessions / Jupyter

Interactive Apps

- Desktops
- NVIDIA DeepOps Desktop
- Servers
- RStudio Server
- VS Code Server
- Services
- Jupyter**

Jupyter version: bc52c1d

This app will launch a customizable Jupyter server on our cluster and automatically present its interface on this webpage. You are free to create your own instances in Conda/Singularity etc on our clusters.

Jupyter Instance

Last Image...

Which Jupyter image to run

Commands to initiate Jupyter

Use JupyterLab instead of Jupyter Notebook?

JupyterLab is the next generation of Jupyter, and is completely compatible with existing Jupyter Notebooks. Note this requires JupyterLab to be installed.

figura 9

Jupyter - Open OnDemand

hpc.cedia.edu.ec/pur/sys/dashboard/batch_connect/sys/jupyter/session_contexts

Partition

cpu-dev

Each queue have different resources limits.

Number of hours

10

Number of hours for which the Jupyter instance will run for

Number of CPU cores

8

Number of cpu cores available for Jupyter instance

Total Memory to allocate

8320

Total memory for Jupyter instance in megabytes

Number of GPUs

0

Number of GPUs requested

I would like to receive an email when the session starts

Launch

figura 10

1.3

USO DE LA HERRAMIENTA INDATA



Ya completado el proceso, vamos a realizar el ciclo de vida de Machine learning, que consiste en adquisición, preprocesamiento de datos, entrenamiento del modelo y evaluación:

Adquisición de datos

En este caso, utilizaremos un dataset que contiene características de archivos PE (Portable Executable) para el entrenamiento del modelo. El dataset incluye aproximadamente 52 características.

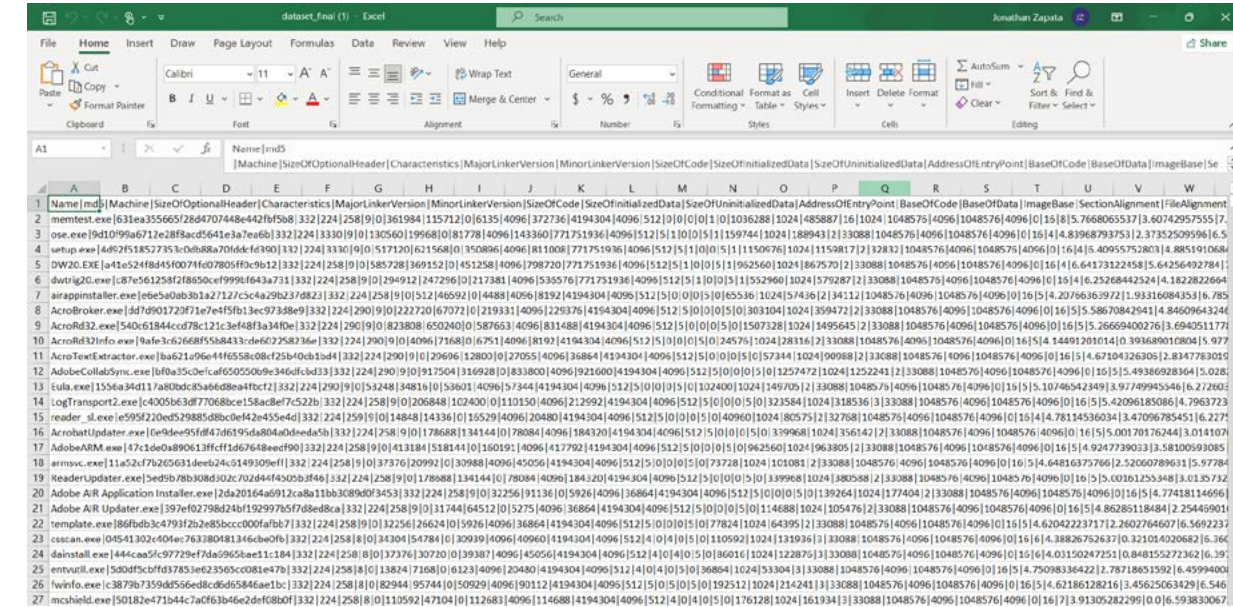


figura 11

Incluimos los resultados dentro del dataset en la aplicación InData para que otros investigadores puedan acceder a ellos de modo que puedan validar y comprobar la eficiencia el modelo de entrenamiento de aprendizaje automático.

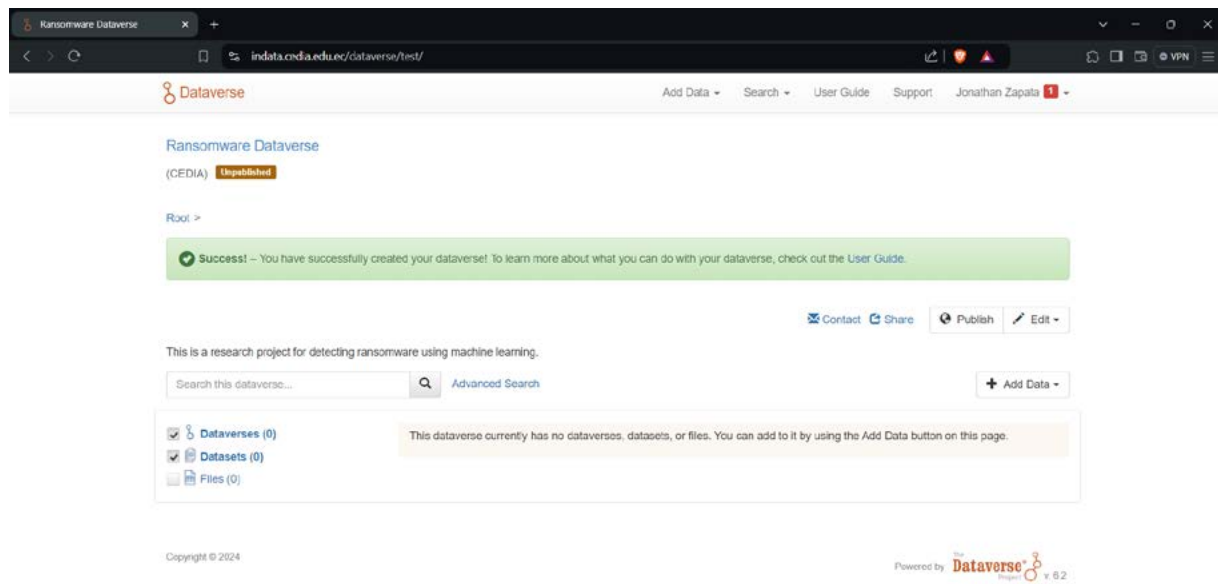
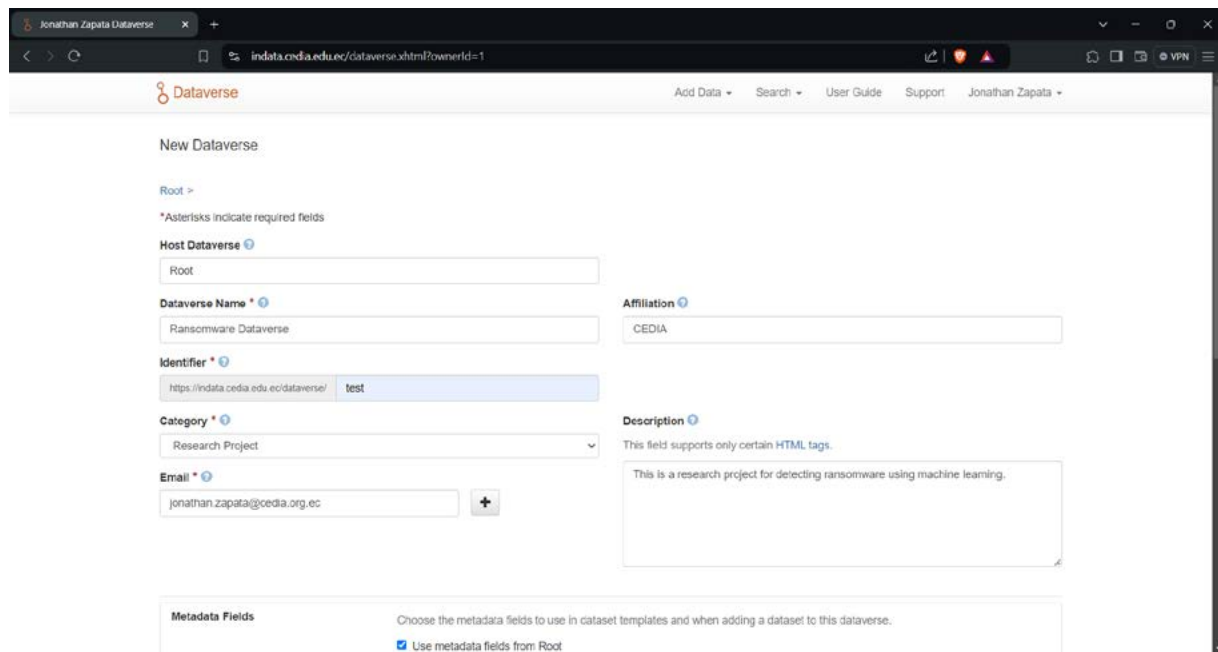


figura 12

Editamos los accesos, para que mas investigadores puedan acceder a nuestro dataset

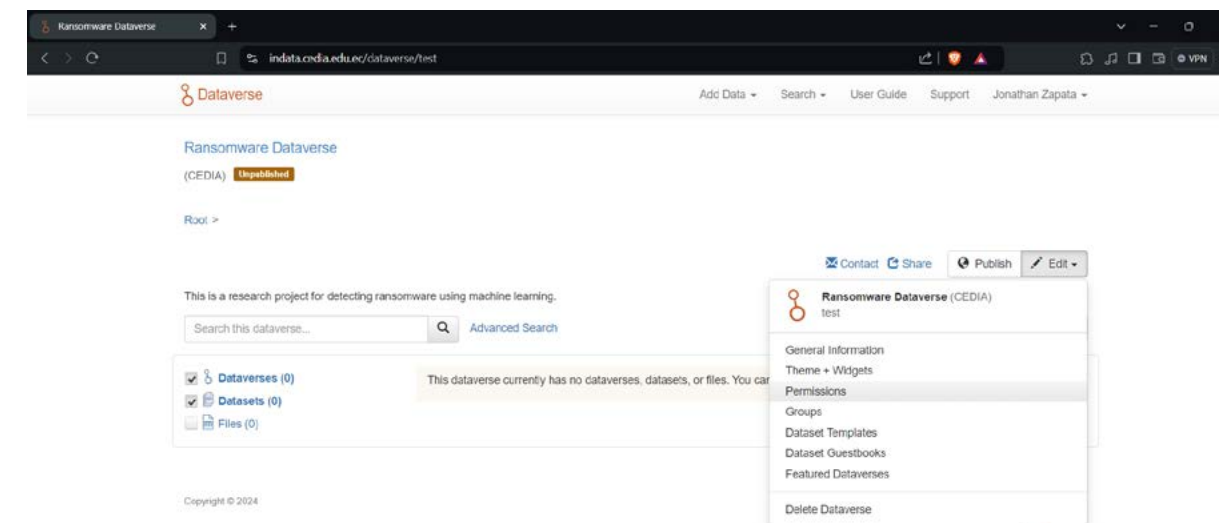


figura 14

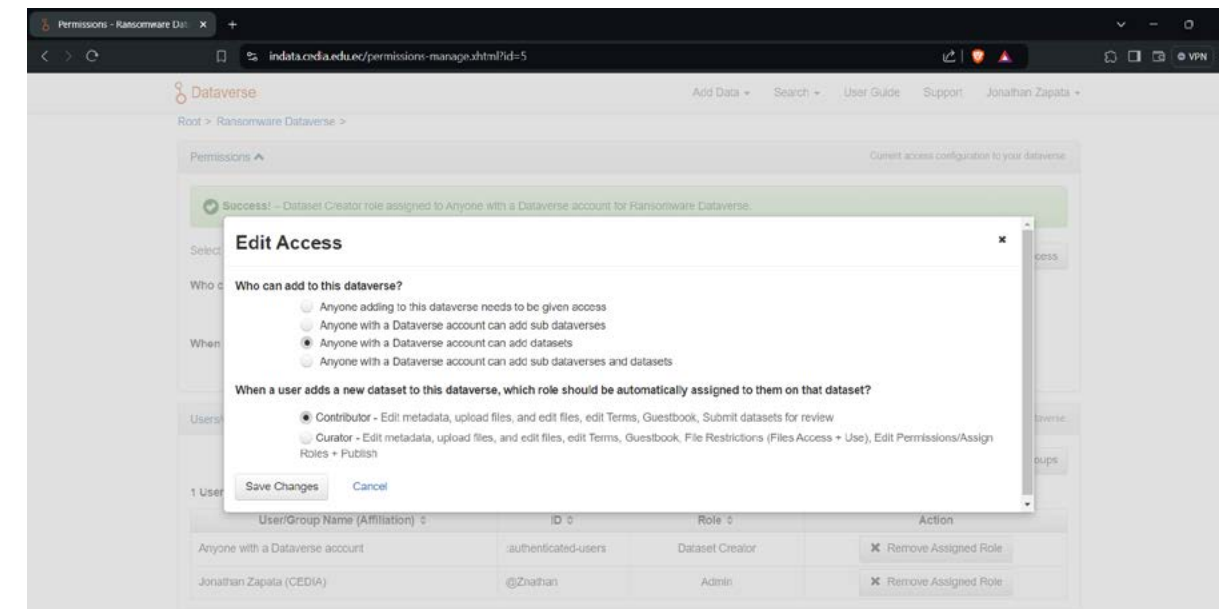


figura 15

Y finalmente publicamos nuestro repositorio de datos en dataverse

Figure 16 shows the 'Add New Dataset' form in Dataverse. The form includes fields for Title, Author, Point of Contact, and Description. The Title is 'Replication Data for: Test'. The Author is 'Zapata, Jonathan' from 'CEDIA'. The Point of Contact is also 'Zapata, Jonathan' from 'CEDIA'. The Description is 'This is a dataset that allows me to include all ransomware samples and benign files to detect and train the machine learning model.'

figura 16

Figure 17 shows the 'Related Publication' section of the form. It contains two citation entries. The first entry is for 'J. Zapata, Machine learning ransomware, 2024, repositorio de GitHub. [Online]. Available: https://github.com/JonathanZapata1/MachineLearningRansomware.git'. The second entry is for 'Masum, M., Shahriar, H., Haddad, H., Faruk, M. J., Valero, M., Khan, M. A., Rahman, M. A., & Adnan, M. I. (2022). Ransomware Classification and Detection With Machine Learning Algorithms. In 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 1246-1252)'. The Identifier Type for the second entry is 'dcl' and the Identifier is '10.1109/CCWC54503.2022.9720869'.

figura 17

Figure 18 shows the '1 File' section of the form. The File Name is 'dataset_model.csv'. The File Path is empty. The Description is 'Add file description...'. There are 'Save Dataset' and 'Cancel' buttons at the bottom.

figura 18

Y finalmente creado el dataset dentro de nuestro repositorio o queda listo para publicar el Dataverse.

Figure 19 shows the dataset page for 'Replication Data for: Test'. The page displays the dataset title, author, and a success message: 'Success! - This dataset has been created.' There are also warning messages: 'Info - This draft version needs to be published. When ready for sharing, please publish it so that others can see these changes.' and 'Dataset Locked - The tabular data files uploaded are being processed and converted into the archival format.' The dataset is currently in 'Draft' status.

figura 19

Para que nuestro Dataverse puedan verlo otros investigadores que buscan mediante un repositorio de Indata CEDIA lo publicamos para que sea publico y de libre acceso

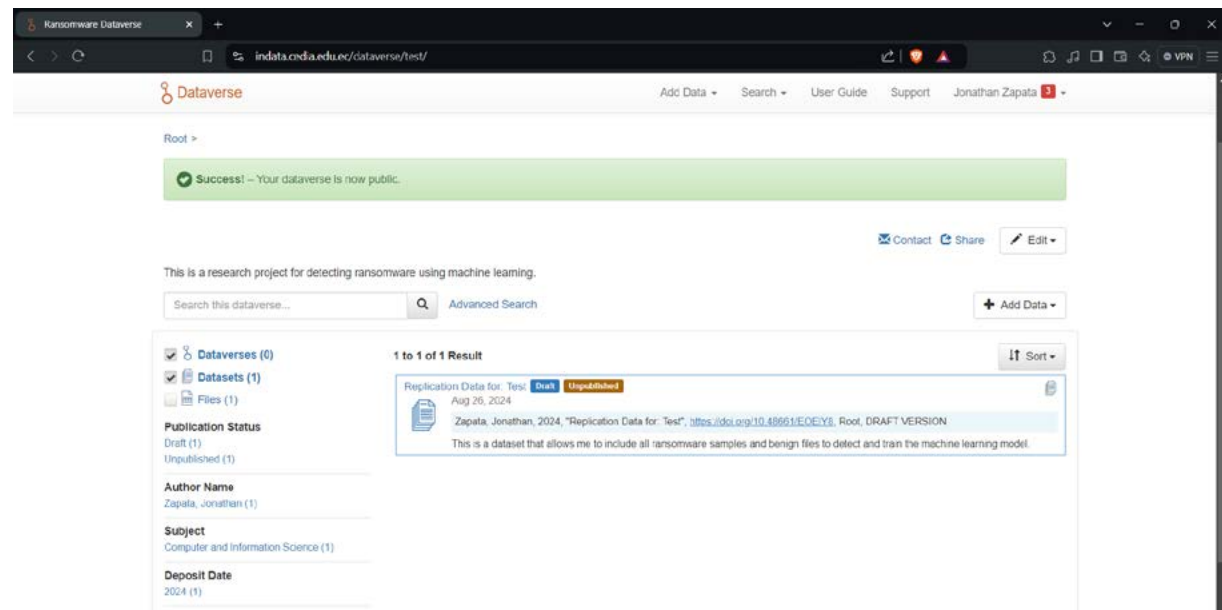


figura 20

Una vez que el archivo ha sido subido, podemos publicar nuestro dataset y compartirlo a través de diversos canales de comunicación, como Facebook, Twitter y LinkedIn.

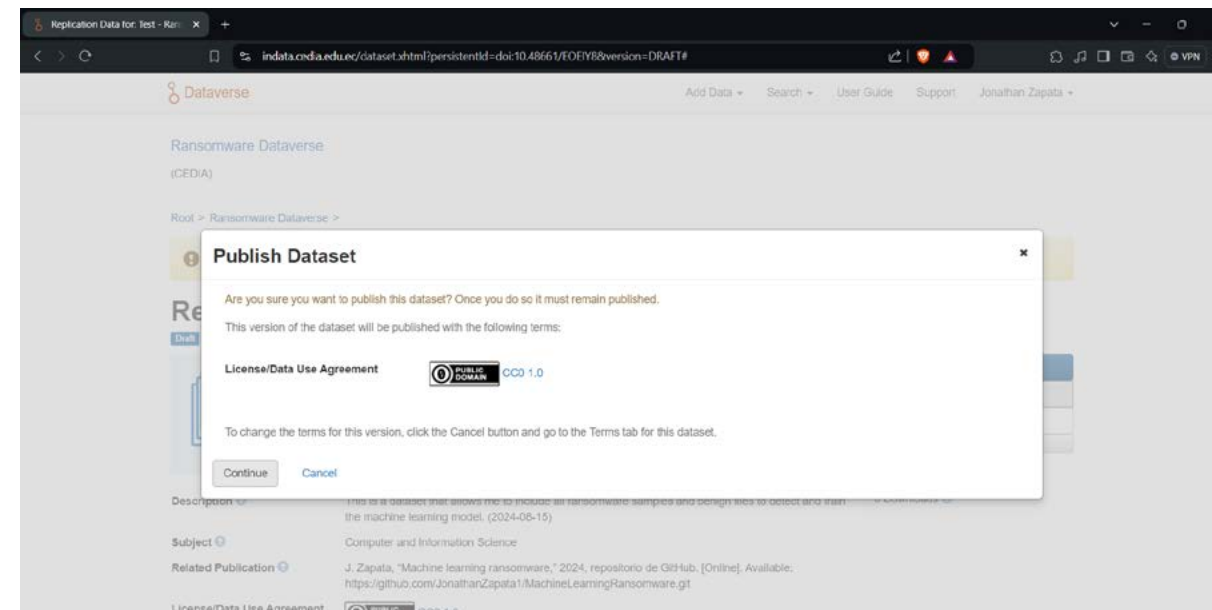


figura 21



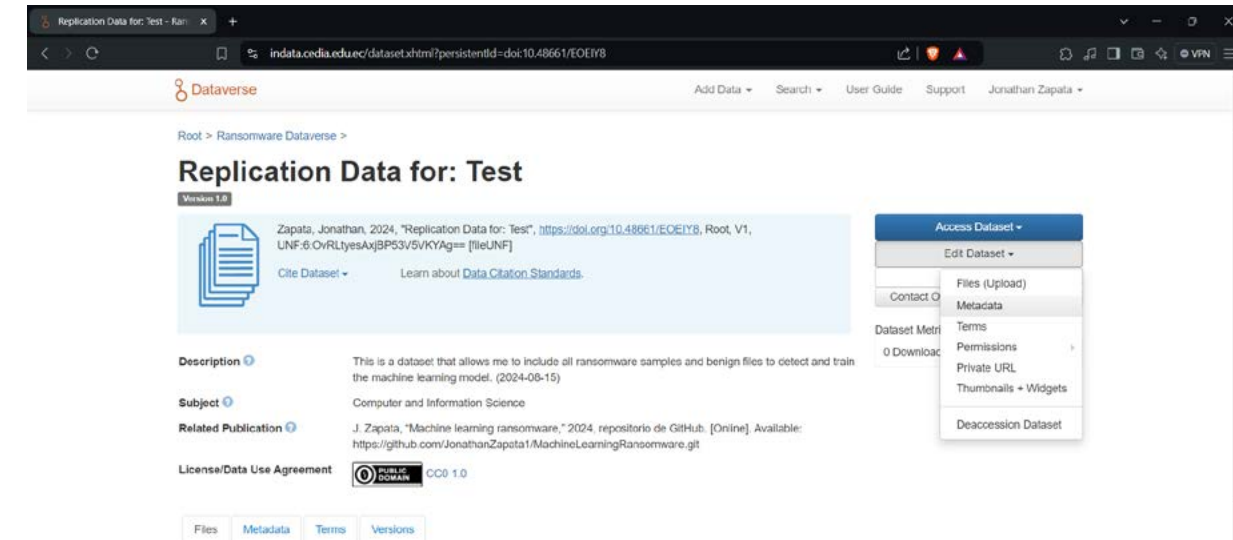
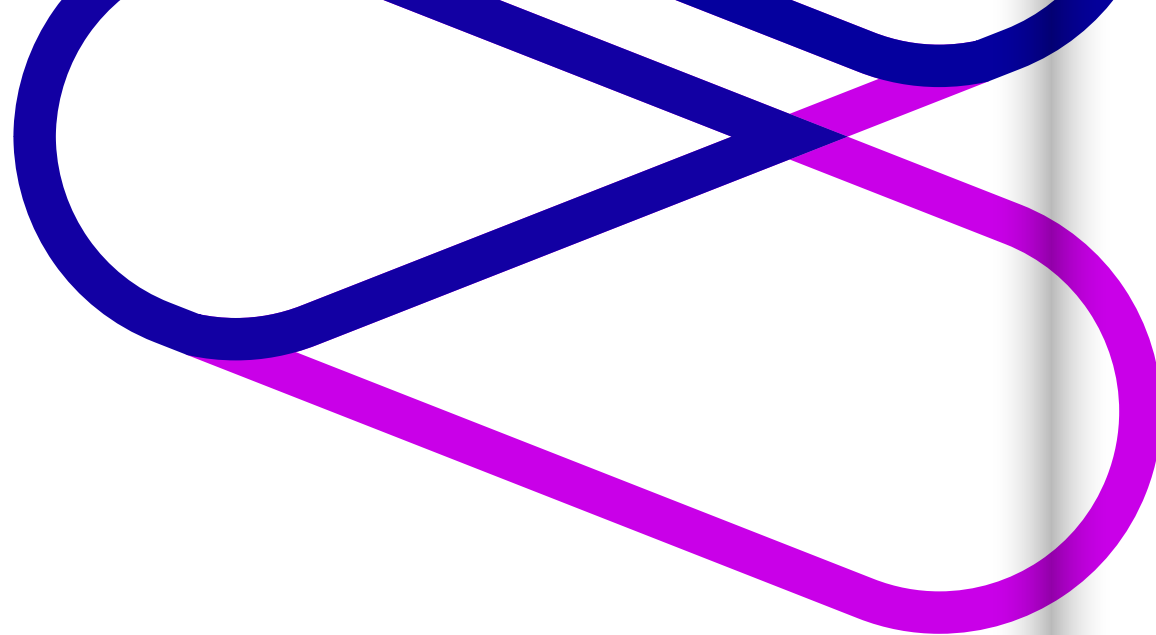


figura 23

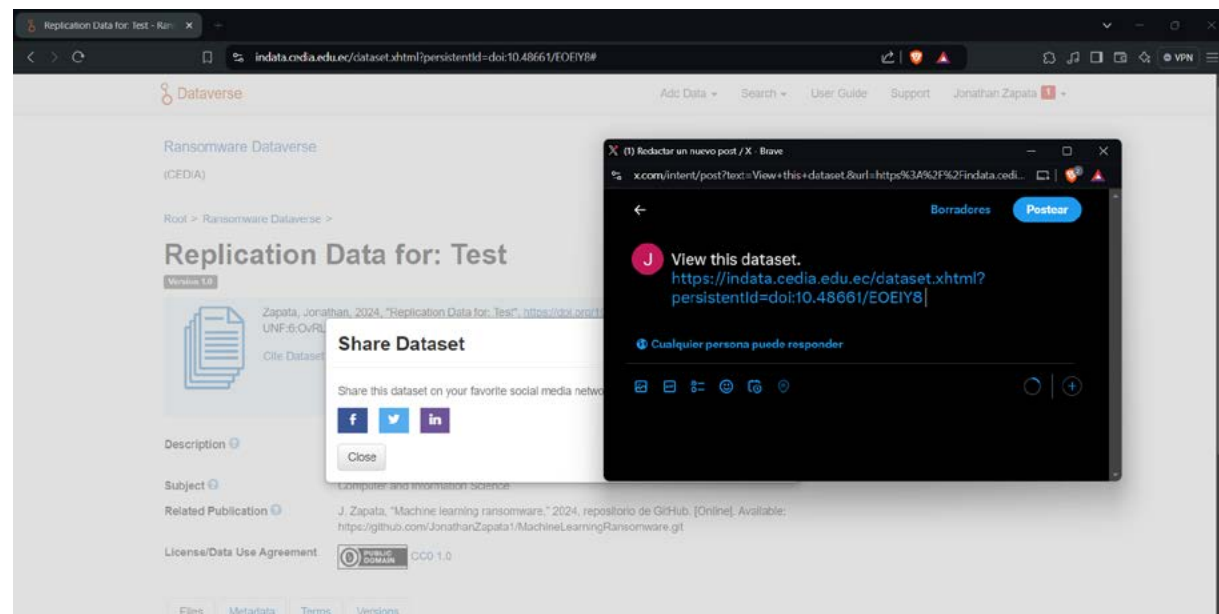


figura 22

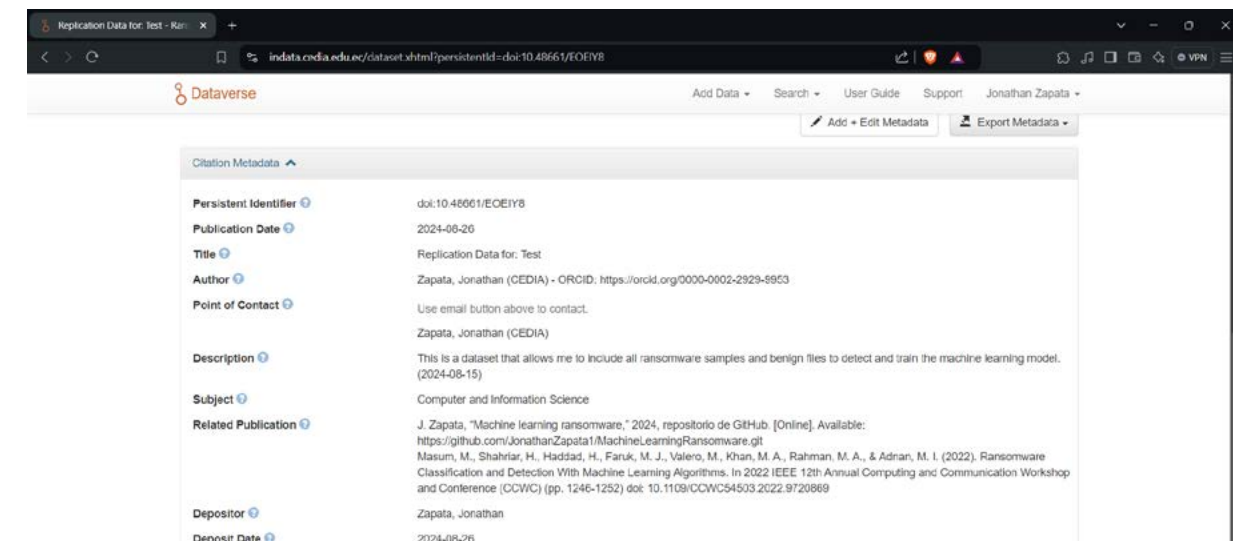


figura 24

Del mismo tenemos la opción de editar, eliminar y agregar el dataset del mismo modo los metadatos que acompañan a este:

Finalmente, nuestro Dataverse es público y accesible dentro de InDATA CEDIA, sin necesidad de realizar un login.

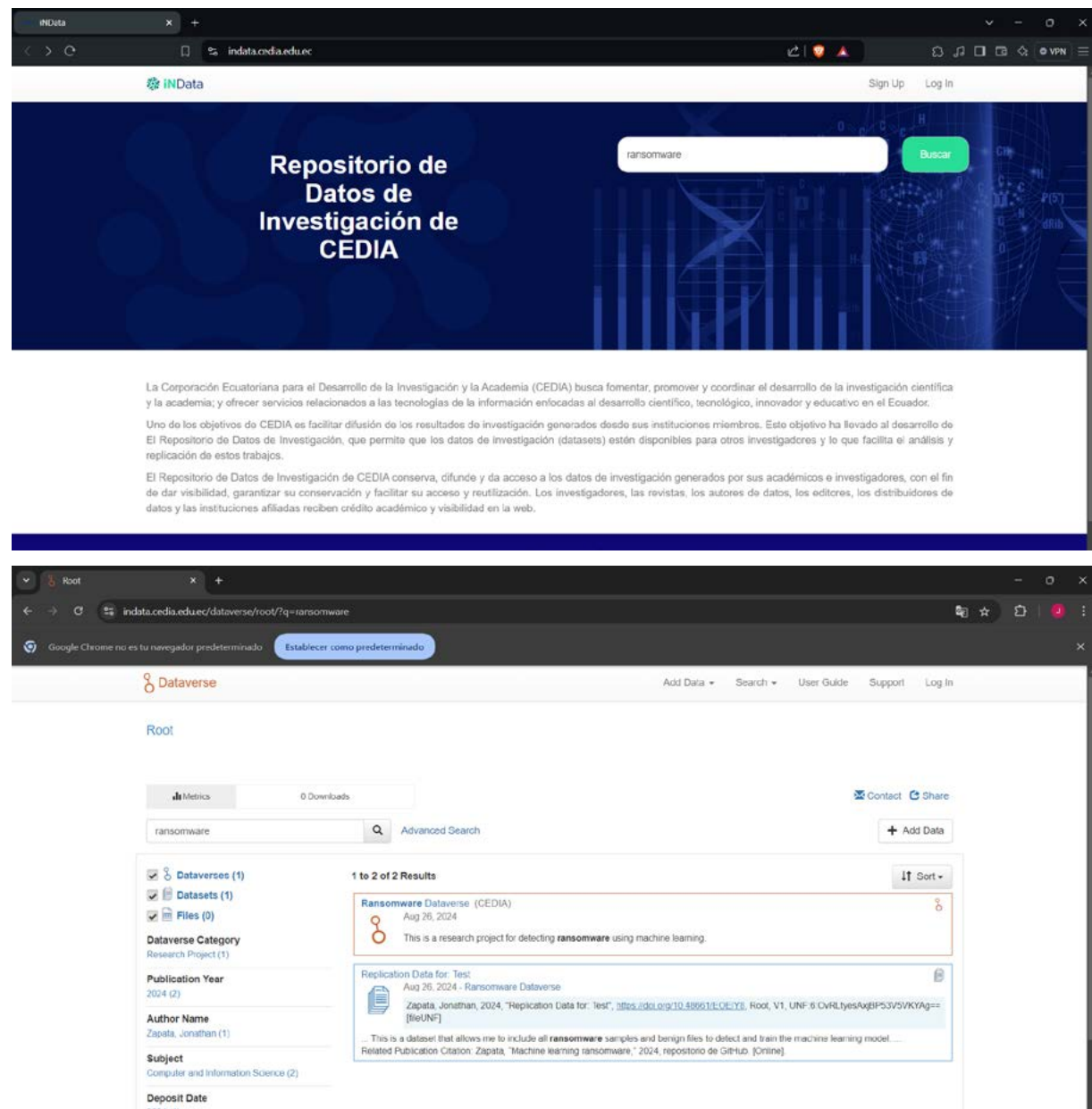


figura 25



figura 26

Y descargar el Dataset

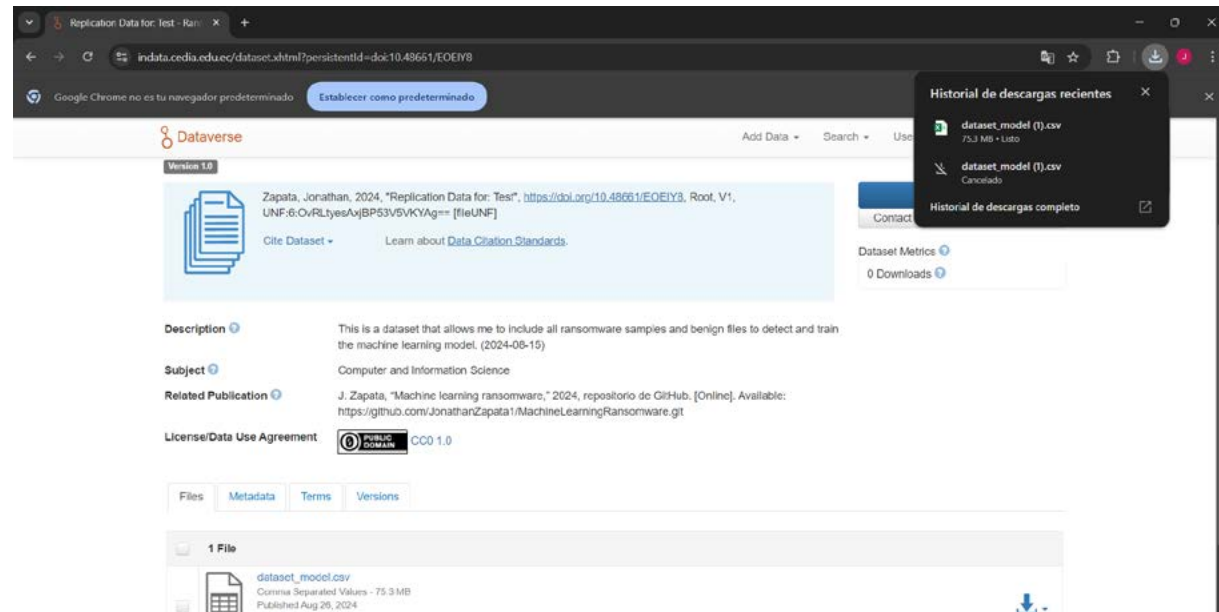
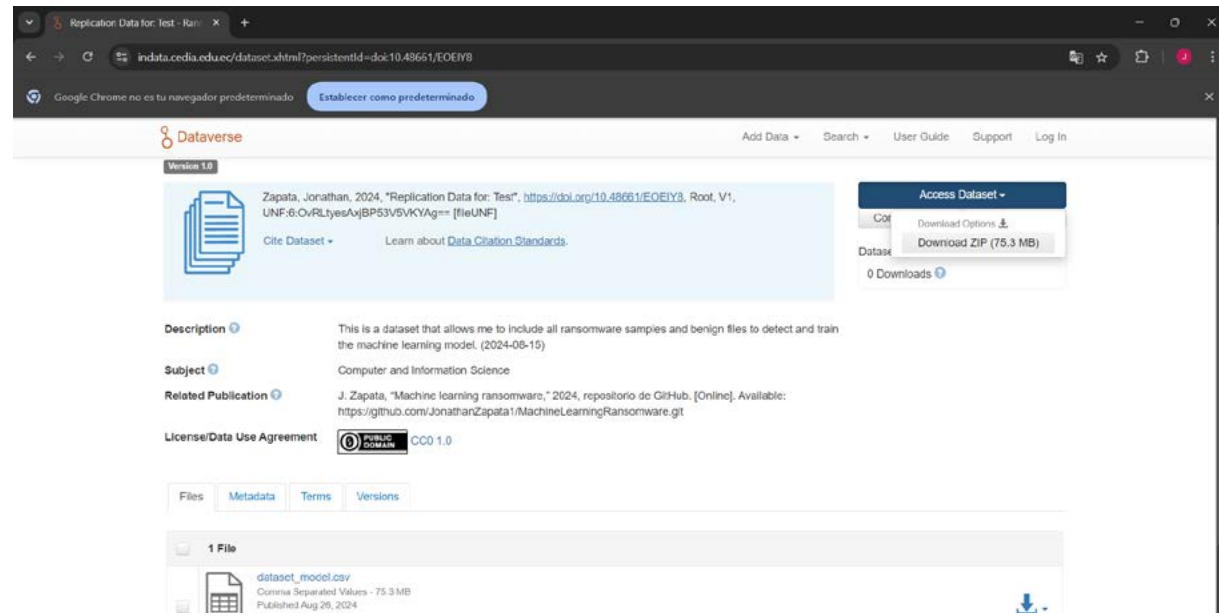


figura 27

figura 28

Preprocesamiento

Durante el preprocesamiento, aplicamos técnicas de selección de modelos y utilizamos herramientas en Python, junto con librerías especializadas, para limpiar los archivos del dataset o CSV. Esto incluye la eliminación de duplicados y el manejo de celdas vacías. Utilizamos **Jupyter de HPC** para realizar estas tareas y subimos el archivo CSV al entorno.

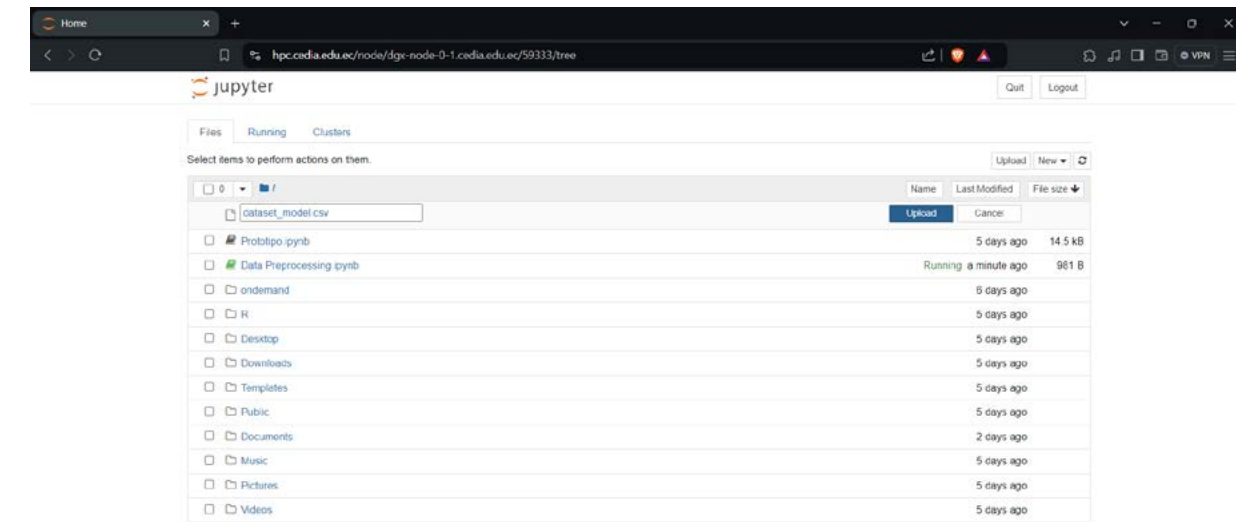
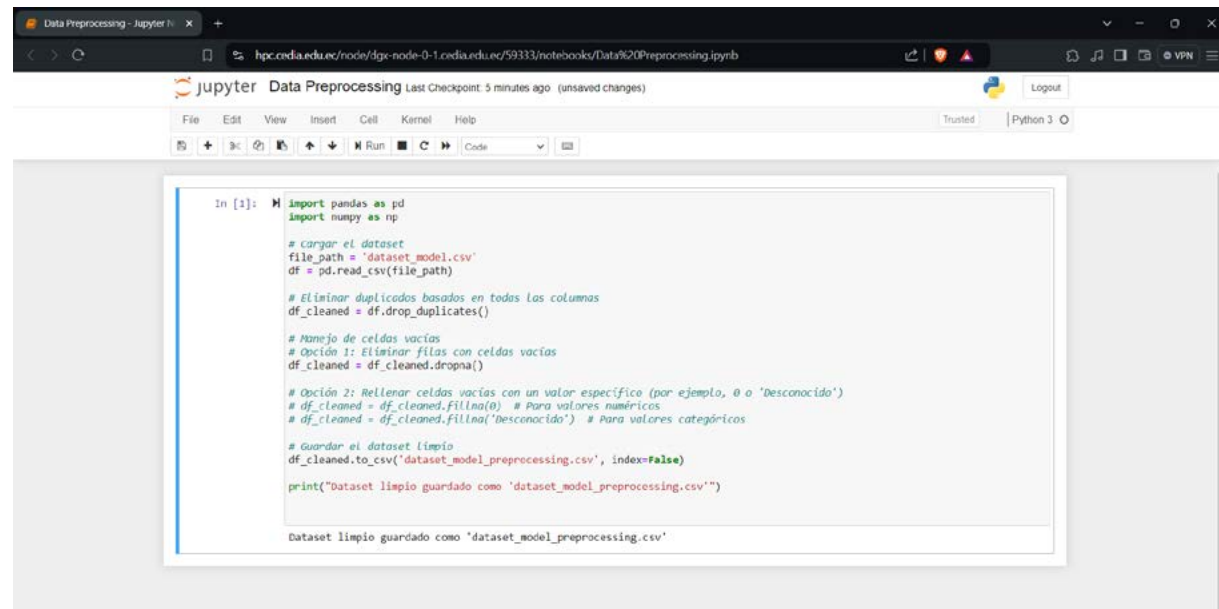


figura 29

Y ejecutamos el siguiente algoritmo.



```
In [1]: import pandas as pd
import numpy as np

# cargar el dataset
file_path = 'dataset_model.csv'
df = pd.read_csv(file_path)

# Eliminar duplicados basados en todas las columnas
df_cleaned = df.drop_duplicates()

# Manejo de celdas vacías
# Opción 1: Eliminar filas con celdas vacías
df_cleaned = df_cleaned.dropna()

# Opción 2: Rellenar celdas vacías con un valor específico (por ejemplo, 0 o 'Desconocido')
# df_cleaned = df_cleaned.fillna(0) # Para valores numéricos
# df_cleaned = df_cleaned.fillna('Desconocido') # Para valores categóricos

# Guardar el dataset limpio
df_cleaned.to_csv('dataset_model_preprocessing.csv', index=False)
print("Dataset limpio guardado como 'dataset_model_preprocessing.csv'")
```

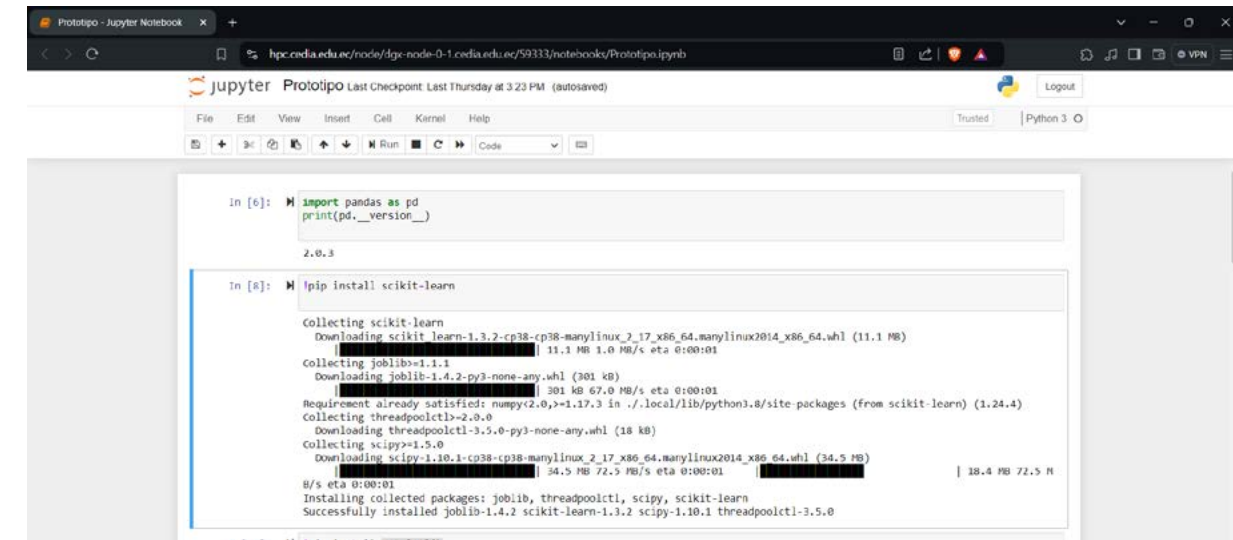
Dataset limpio guardado como 'dataset_model_preprocessing.csv'

figura 30

NOTA: Aunque el tiempo de ejecución del clúster en Jupyter finalice, tanto los datasets como los archivos .py relacionados con las ejecuciones en Python se guardarán en tu carpeta personal.

Entrenamiento del modelo

Antes de continuar con el entrenamiento del modelo es la ejecución e instalación de librerías que se van a usar en ese caso scikit-learn, matplotlib.



```
In [6]: import pandas as pd
print(pd.__version__)

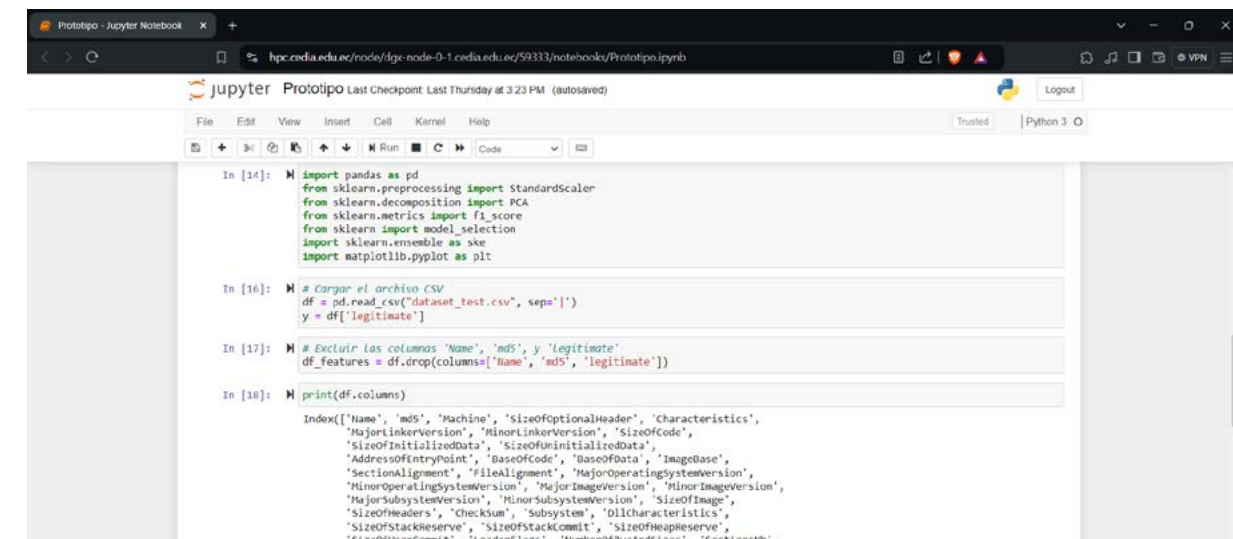
2.0.3

In [8]: !pip install scikit-learn

Collecting scikit-learn
  Downloading scikit_learn-1.3.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.1 MB)
    | 11.1 MB 1.0 MB/s eta 0:00:01
Collecting joblib<=1.1.1
  Downloading joblib-1.4.2-py3-none-any.whl (301 kB)
    | 301 kB 67.0 MB/s eta 0:00:01
Requirement already satisfied: numpy<2.0,>=1.17.3 in ./local/lib/python3.8/site-packages (from scikit-learn) (1.24.4)
Collecting threadpoolctl<=2.0.0
  Downloading threadpoolctl-3.5.0-py3-none-any.whl (18 kB)
Collecting scipy<=1.5.0
  Downloading scipy-1.10.1-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (34.5 MB)
    | 34.5 MB 72.5 MB/s eta 0:00:01
Installing collected packages: joblib, threadpoolctl, scipy, scikit-learn
Successfully installed joblib-1.4.2 scikit-learn-1.3.2 scipy-1.10.1 threadpoolctl-3.5.0
```

figura 31

Después, comienza la codificación y la creación del modelo a entrenar, empezando con el algoritmo Random Forest



```
In [14]: import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import f1_score
from sklearn import model_selection
import sklearn.ensemble as ske
import matplotlib.pyplot as plt

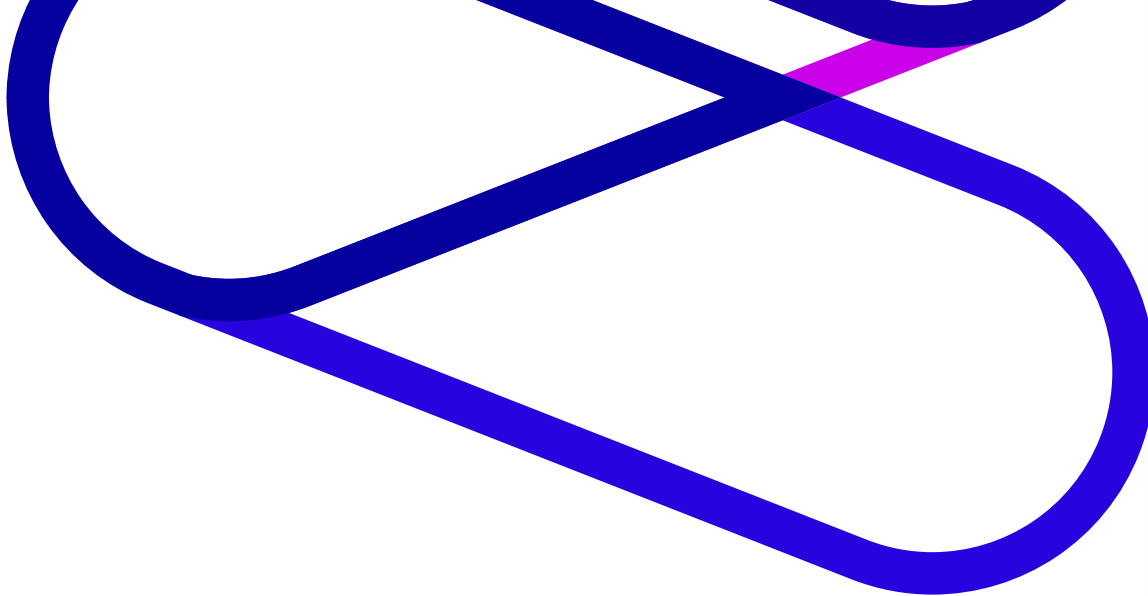
In [16]: # Cargar el archivo CSV
df = pd.read_csv("dataset_test.csv", sep='|')
y = df['legitimate']

In [17]: # Excluir las columnas 'Name', 'md5', y 'legitimate'
df_features = df.drop(columns=['Name', 'md5', 'legitimate'])

In [18]: # print(df.columns)

Index(['Name', 'md5', 'Machine', 'SizeOfOptionalHeader', 'Characteristics',
'MajorLinkerVersion', 'MinorLinkerVersion', 'SizeOfCode',
'SizeOfInitializedData', 'SizeOfUninitializedData',
'AddressOfEntryPoint', 'BaseOfCode', 'BaseOfData', 'ImageBase',
'SectionAlignment', 'FileAlignment', 'MajorOperatingSystemVersion',
'MinorOperatingSystemVersion', 'MajorImageVersion', 'MinorImageVersion',
'MajorSubsystemVersion', 'MinorSubsystemVersion', 'SizeOfImage',
'SizeOfHeaders', 'Checksum', 'Subsystem', 'DllCharacteristics',
'SizeOfStackReserve', 'SizeOfStackCommit', 'SizeOfHeapReserve',
'SizeOfHeapCommit', 'LoaderFlags', 'NumberOfRvaAndSizes', 'SectionNb'],
dtype='object')
```

figura 32



```
hpc.cedia.edu.ec/node/dgx-node-0-1.cedia.edu.ec/59333/notebooks/Prototipo.ipynb
jupyter Prototipo Last Checkpoint Last Thursday at 3:23 PM (autosaved)

In [26]: # Estandarización de los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_features)
print(X_scaled.shape)

# Aplicar PCA
pca = PCA(0.95)
X_pca = pca.fit_transform(X_scaled)
print(X_pca.shape)
print(pca.explained_variance_ratio_)

# Dividir datos en datos de entrenamiento y de prueba
X_train, X_test, y_train, y_test = model_selection.train_test_split(X_pca, y, test_size=0.2, random_state=2)

# Imprimir el número de muestras de entrenamiento y prueba
print("\n[*] Ejemplos de entrenamiento: ", len(X_train))
print("\n[*] Muestras de prueba: ", len(X_test))

(217658, 54)
(217658, 36)
[0.09139714 0.08362185 0.05821202 0.05007173 0.0418178 0.04042469
 0.03903565 0.03756709 0.03696948 0.03467353 0.03298574 0.02792588
 0.02494339 0.021862 0.02051288 0.01945013 0.01890382 0.01854946
 0.01851761 0.01849512 0.01845422 0.01841119 0.01824991 0.01753057
 0.01692272 0.01579469 0.01504638 0.01404467 0.01311339 0.01293752
 0.01234813 0.0118361 0.01031608 0.00924180 0.0092627 0.00690005]

[*] Ejemplos de entrenamiento: 174126
[*] Muestras de prueba: 43532
```

figura 33

El conjunto de datos está en un 80% para entrenamiento y un 20% para prueba. Esta división permitió entrenar el modelo y evaluar qué variables podrían ajustarse o qué métodos adicionales podrían explorarse para mejorar la calidad del modelo en el proceso de aprendizaje automático.

Y seguidamente los algoritmos restantes:

- Decision Tree

```
hpc.cedia.edu.ec/node/dgx-node-0-2.cedia.edu.ec/29930/notebooks/Prototipo.ipynb
jupyter Prototipo Last Checkpoint 2 minutes ago (unsaved changes)

In [26]: # Importar librerías
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score, f1_score

In [27]: # Cargar el archivo CSV
df = pd.read_csv("dataset_model_preprocessing.csv", sep="|")
# Eliminar las columnas no necesarias
df = df.drop(columns=["Name", "md5"])

# Separar las características y la etiqueta
X = df.drop(columns=["legitimate"])
y = df["legitimate"]

In [30]: # Dividir el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Inicializar el clasificador de Árbol de Decisión
dt_classifier = DecisionTreeClassifier(random_state=42)

# Entrenar el clasificador
dt_classifier.fit(X_train, y_train)

# Hacer predicciones en el conjunto de prueba
y_pred = dt_classifier.predict(X_test)
```

figura 34

- AdaBoost

```
hpc.cedia.edu.ec/node/dgx-node-0-2.cedia.edu.ec/29930/notebooks/Prototipo.ipynb
jupyter Prototipo Last Checkpoint 4 minutes ago (unsaved changes)

In [31]: # Importar librerías
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score, f1_score

In [27]: # Cargar el archivo CSV
df = pd.read_csv("dataset_model_preprocessing.csv", sep="|")
# Eliminar las columnas no necesarias
df = df.drop(columns=["Name", "md5"])

# Separar las características y la etiqueta
X = df.drop(columns=["legitimate"])
y = df["legitimate"]

In [32]: # Dividir el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Inicializar el clasificador base para AdaBoost (DecisionTreeClassifier)
base_classifier = DecisionTreeClassifier(max_depth=1, random_state=42)

# Inicializar el clasificador AdaBoost
ada_classifier = AdaBoostClassifier(base_classifier, n_estimators=50, random_state=42)

# Entrenar el clasificador
ada_classifier.fit(X_train, y_train)
```

figura 35



- Logistic Regression

```

In [35]: # Importar librerías
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import f1_score
from sklearn import model_selection
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression

In [33]: # Cargar el archivo CSV
df = pd.read_csv("dataset_model_preprocessing.csv", sep='|')
# Excluir las columnas "Name", "md5", y "legitimate"
df_features = df.drop(columns=["Name", "md5", "legitimate"])

# Estandarización de los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_features)
print(X_scaled.shape)

(217656, 54)

In [37]: # Entrenar el algoritmo de Regresión Logística en el conjunto de datos de entrenamiento
clf_lr = LogisticRegression(random_state=42, max_iter=1000)
clf_lr.fit(X_train, y_train)

# Predecir las clases del conjunto de pruebas
y_pred_lr = clf_lr.predict(X_test)

```

figura 37

- K-Nearest Neighbors (KNN)

```

from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression

In [33]: # Cargar el archivo CSV
df = pd.read_csv("dataset_model_preprocessing.csv", sep='|')
# Excluir las columnas "Name", "md5", y "legitimate"
df_features = df.drop(columns=["Name", "md5", "legitimate"])

# Estandarización de los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_features)
print(X_scaled.shape)

(217656, 54)

In [36]: # Dividir datos en datos de entrenamiento y de prueba
X_train, X_test, y_train, y_test = model_selection.train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Imprimir el número de muestras de entrenamiento y prueba
print("\n[*] Ejemplos de entrenamiento: ", len(X_train))
print("\n[*] Muestras de prueba: ", len(X_test))

# Entrenar el algoritmo de k-NN en el conjunto de datos de entrenamiento
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

```

figura 36

- Naïve Bayes

```

In [38]: # Importar librerías
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import f1_score
from sklearn import model_selection
from sklearn.naive_bayes import GaussianNB

In [39]: # Cargar el archivo CSV
df = pd.read_csv("dataset_model_preprocessing.csv", sep='|')
# Excluir las columnas "Name", "md5", y "legitimate"
df_features = df.drop(columns=["Name", "md5", "legitimate"])

# Estandarización de los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_features)
print(X_scaled.shape)

(217656, 54)

In [40]: # Dividir datos en datos de entrenamiento y de prueba
X_train, X_test, y_train, y_test = model_selection.train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Imprimir el número de muestras de entrenamiento y prueba
print("\n[*] Ejemplos de entrenamiento: ", len(X_train))
print("\n[*] Muestras de prueba: ", len(X_test))

# Entrenar el algoritmo de Naïve Bayes en el conjunto de datos de entrenamiento

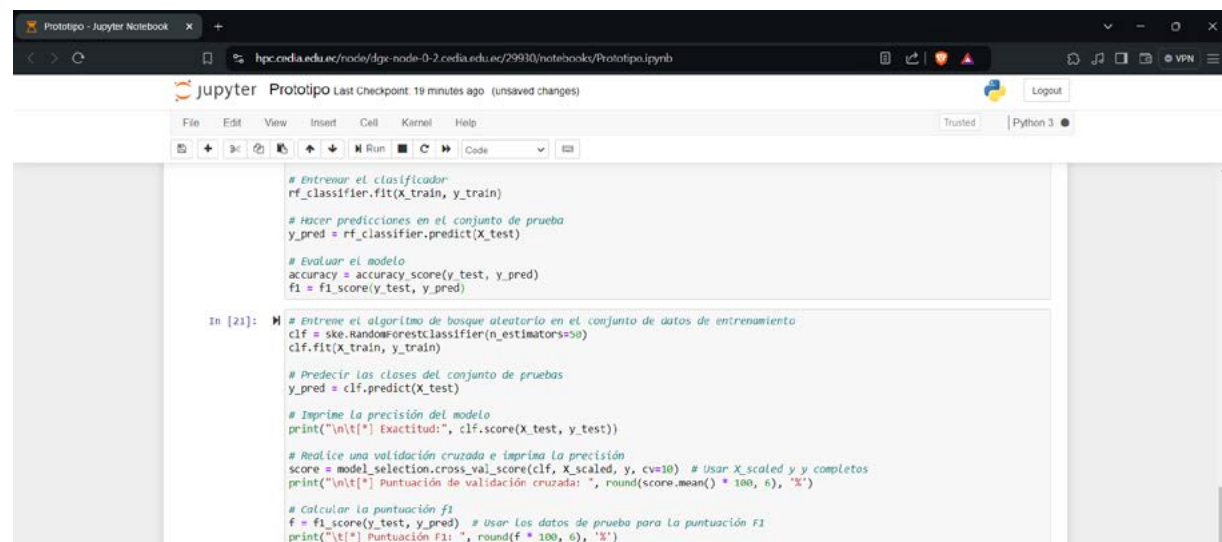
```

figura 38

Evaluación

Finalmente, al verificar y evaluar cada uno de los algoritmos, llegamos a los modelos de calificación utilizando métricas como la puntuación F1, la exactitud y la validación cruzada.

- Random Forest



```
# Entrenar el clasificador
rf_classifier.fit(X_train, y_train)

# Hacer predicciones en el conjunto de prueba
y_pred = rf_classifier.predict(X_test)

# Evaluar el modelo
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

In [21]: # Entrene el algoritmo de bosque aleatorio en el conjunto de datos de entrenamiento
clf = sklearn.ensemble.RandomForestClassifier(n_estimators=50)
clf.fit(X_train, y_train)

# Predecir las clases del conjunto de pruebas
y_pred = clf.predict(X_test)

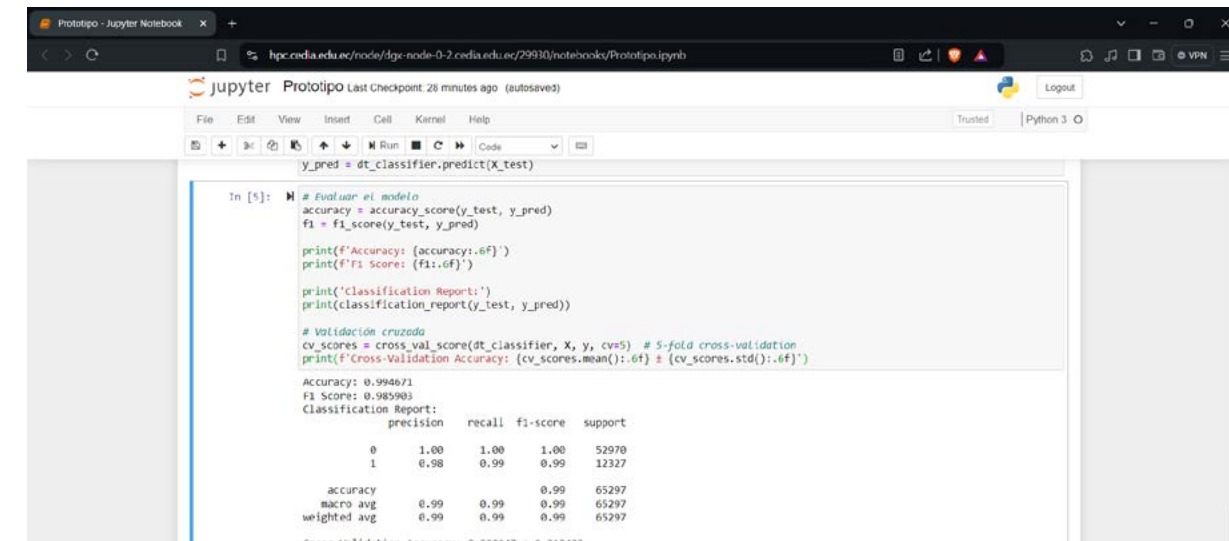
# Imprime la precisión del modelo
print("\n[*] Exactitud: ", clf.score(X_test, y_test))

# Realice una validación cruzada e imprima la precisión
score = model_selection.cross_val_score(clf, X_scaled, y, cv=10) # Usar X_scaled y y completos
print("\n[*] Puntuación de validación cruzada: ", round(score.mean() * 100, 6), "%")

# Calcular la puntuación f1
f = f1_score(y_test, y_pred) # Usar los datos de prueba para la puntuación f1
print("\n[*] Puntuación F1: ", round(f * 100, 6), "%")
```

figura 39

- Decision Tree



```
y_pred = dt_classifier.predict(X_test)

In [5]: # Evaluar el modelo
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f'Accuracy: {accuracy:.6f}')
print(f'F1 Score: {f1:.6f}')

print('Classification Report:')
print(classification_report(y_test, y_pred))

# Validación cruzada
cv_scores = cross_val_score(dt_classifier, X, y, cv=5) # 5-fold cross-validation
print(f'Cross-Validation Accuracy: {cv_scores.mean():.6f} ± {cv_scores.std():.6f}')

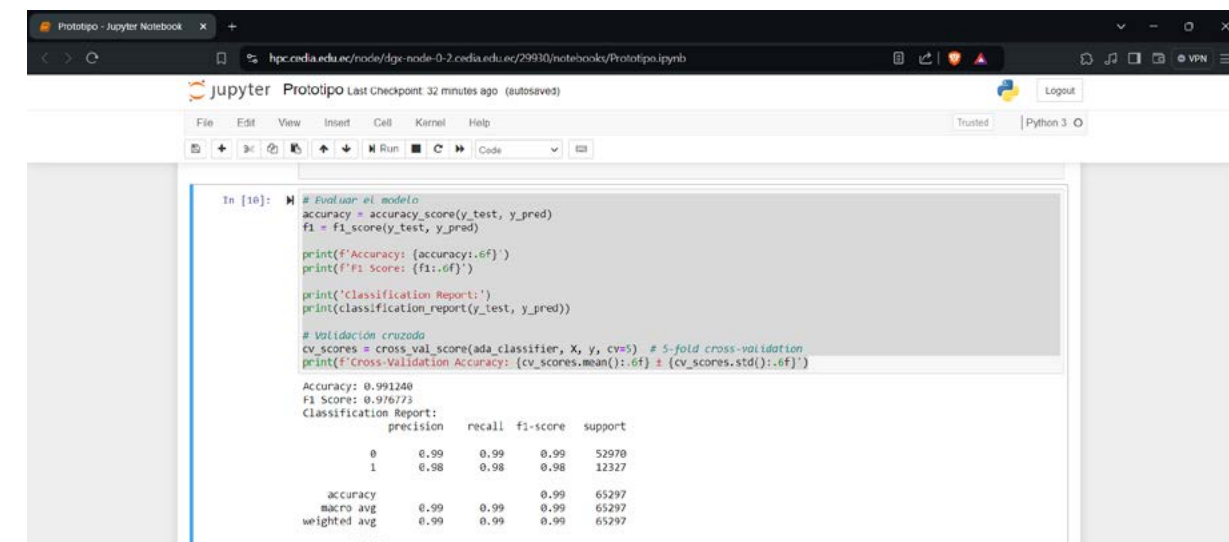
Accuracy: 0.994671
F1 Score: 0.985903
Classification Report:
              precision    recall  f1-score   support

     0       1.00      1.00      1.00     52970
     1       0.98      0.99      0.99     12327

 accuracy          0.99          0.99          0.99          65297
 macro avg         0.99          0.99          0.99          65297
 weighted avg      0.99          0.99          0.99          65297
```

figura 40

- AdaBoost



```
In [10]: # Evaluar el modelo
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f'Accuracy: {accuracy:.6f}')
print(f'F1 Score: {f1:.6f}')

print('Classification Report:')
print(classification_report(y_test, y_pred))

# Validación cruzada
cv_scores = cross_val_score(ada_classifier, X, y, cv=5) # 5-fold cross-validation
print(f'Cross-Validation Accuracy: {cv_scores.mean():.6f} ± {cv_scores.std():.6f}')

Accuracy: 0.991240
F1 Score: 0.976773
Classification Report:
              precision    recall  f1-score   support

     0       0.99      0.99      0.99     52970
     1       0.98      0.98      0.98     12327

 accuracy          0.99          0.99          0.99          65297
 macro avg         0.99          0.99          0.99          65297
 weighted avg      0.99          0.99          0.99          65297
```

figura 41

- K-Nearest Neighbors (KNN)

```

# Entrenar el algoritmo de k-NN en el conjunto de datos de entrenamiento
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Predecir las clases del conjunto de pruebas
y_pred_knn = knn.predict(X_test)

[*] Ejemplos de entrenamiento: 174124
[*] Muestras de prueba: 43532

In [14]: # Imprimir la precisión del modelo k-NN
print("\n[*] Exactitud (k-NN):", knn.score(X_test, y_test))

# Realizar una validación cruzada e imprimir la precisión para k-NN
score_knn = model_selection.cross_val_score(knn, X_scaled, y, cv=10) # Usar X_scaled y y completos
print("\n[*] Puntuación de validación cruzada (k-NN): ", round(score_knn.mean() * 100, 6), "%")

# Calcular la puntuación F1 para k-NN
f_knn = f1_score(y_test, y_pred_knn) # Usar los datos de prueba para la puntuación F1
print("\n[*] Puntuación F1 (k-NN): ", round(f_knn * 100, 6), "%")

[*] Exactitud (k-NN): 0.99283285867867
[*] Puntuación de validación cruzada (k-NN): 98.800412 %
[*] Puntuación F1 (k-NN): 98.12292 %

```

figura 42

- Logistic Regression

```

# Entrenar el algoritmo de Regresión Logística en el conjunto de datos de entrenamiento
clf_lr = LogisticRegression(random_state=42, max_iter=1000)
clf_lr.fit(X_train, y_train)

# Predecir las clases del conjunto de pruebas
y_pred_lr = clf_lr.predict(X_test)

[*] Ejemplos de entrenamiento: 174124
[*] Muestras de prueba: 43532

In [7]: # Imprimir la precisión del modelo de Regresión Logística
print("\n[*] Exactitud (Logistic Regression):", clf_lr.score(X_test, y_test))

# Realizar una validación cruzada e imprimir la precisión para Regresión Logística
score_lr = model_selection.cross_val_score(clf_lr, X_scaled, y, cv=10) # Usar X_scaled y y completos
print("\n[*] Puntuación de validación cruzada (Logistic Regression): ", round(score_lr.mean() * 100, 6), "%")

# Calcular la puntuación F1 para Regresión Logística
f_lr = f1_score(y_test, y_pred_lr) # Usar los datos de prueba para la puntuación F1
print("\n[*] Puntuación F1 (Logistic Regression): ", round(f_lr * 100, 6), "%")

[*] Exactitud (Logistic Regression): 0.9832077552145548
[*] Puntuación de validación cruzada (Logistic Regression): 98.227031 %
[*] Puntuación F1 (Logistic Regression): 95.534787 %

```

figura 43

- Naïve Bayes

```

# Entrenar el algoritmo de Naïve Bayes en el conjunto de datos de entrenamiento
clf = GaussianNB()
clf.fit(X_train, y_train)

# Predecir las clases del conjunto de pruebas
y_pred = clf.predict(X_test)

[*] Ejemplos de entrenamiento: 174124
[*] Muestras de prueba: 43532

In [11]: # Imprimir la precisión del modelo
print("\n[*] Exactitud:", clf.score(X_test, y_test))

# Realizar una validación cruzada e imprimir la precisión
score = model_selection.cross_val_score(clf, X_scaled, y, cv=10) # Usar X_scaled y y completos
print("\n[*] Puntuación de validación cruzada: ", round(score.mean() * 100, 6), "%")

# Calcular la puntuación F1
f = f1_score(y_test, y_pred) # Usar los datos de prueba para la puntuación F1
print("\n[*] Puntuación F1: ", round(f * 100, 6), "%")

[*] Exactitud: 0.4245842139116053
[*] Puntuación de validación cruzada: 42.74006 %
[*] Puntuación F1: 39.619139 %

```

figura 44

De manera similar, podemos graficar estos datos para validar los modelos de evaluación mediante la puntuación F1, la validación cruzada y la precisión, utilizando diagramas de confusión. Esto permitirá visualizar de forma efectiva el rendimiento de los modelos y facilitará la interpretación de los resultados obtenidos.

```

df = df.drop(columns=['Name', 'msd'])

# Separar las características y la etiqueta
X = df.drop(columns=['legitimato'])
y = df['legitimato']

In [18]: # Dividir el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Inicializar los clasificadores
classifiers = {
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "AdaBoost": AdaBoostClassifier(DecisionTreeClassifier(max_depth=1, random_state=42), n_estimators=50, random_state=42),
    "KNN": KNeighborsClassifier(n_neighbors=5),
    "Logistic Regression": LogisticRegression(random_state=42, max_iter=10000),
    "Naive Bayes": GaussianNB()
}

In [19]: # Entrenar los clasificadores y generar las matrices de confusión
fig, axes = plt.subplots(3, 2, figsize=(12, 10)) # 3 filas, 2 columnas
axes = axes.ravel() # Aplanar la matriz de ejes para un fácil acceso

for idx, (name, clf) in enumerate(classifiers.items()):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    conf_matrix = confusion_matrix(y_test, y_pred)

# Visualizar la matriz de confusión
disp = ConfusionMatrixDisplay(conf_matrix, display_labels=clf.classes_)
disp.plot(ax=axes[idx], values_format='d')

```

figura 45

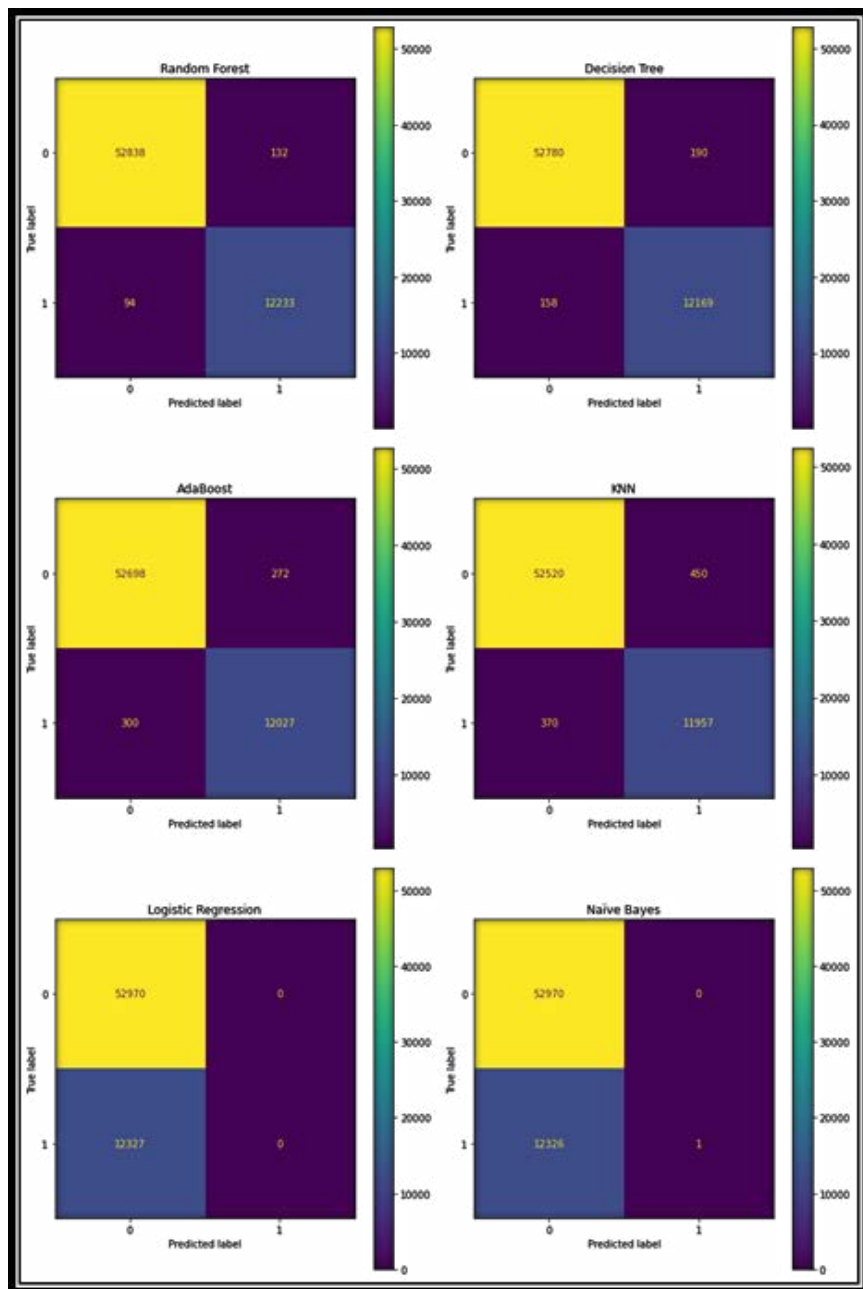


figura 46

Con esto, completamos la investigación sobre el uso de machine learning para la detección de ransomware. Aunque el proceso puede ser arduo, la utilización de supercomputadoras HPC puede optimizar significativamente las tareas de optimización y reducir los tiempos de espera durante la compilación del código. Esto resulta especialmente útil en investigaciones que requieren procesar y preprocesar grandes cantidades de datos, ya sea para predicciones o para la visualización de datos.



1.4

USO DE LA HERRAMIENTA GITLAB

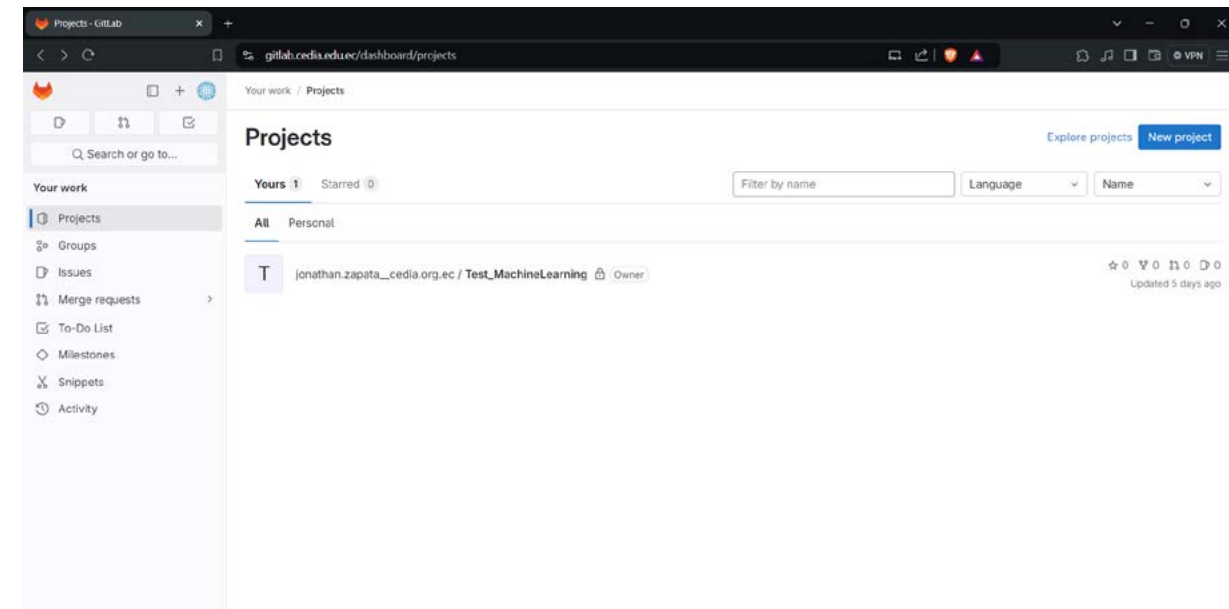


figura 47

Además, al ejecutar estos scripts de desarrollo en Python, es fundamental capturarlos o guardarlos adecuadamente para facilitar la colaboración en equipo y permitir el mantenimiento o las mejoras continuas del código. Por ello, se utiliza la plataforma GitLab para almacenar todo en un único repositorio, asegurando así una gestión eficiente y un acceso centralizado a los scripts.

Para maximizar el uso de **GitLab**, convertimos la investigación en una prueba de concepto, desarrollando una aplicación web. Además, establecimos una conexión directa con GitLab para gestionar el proyecto.

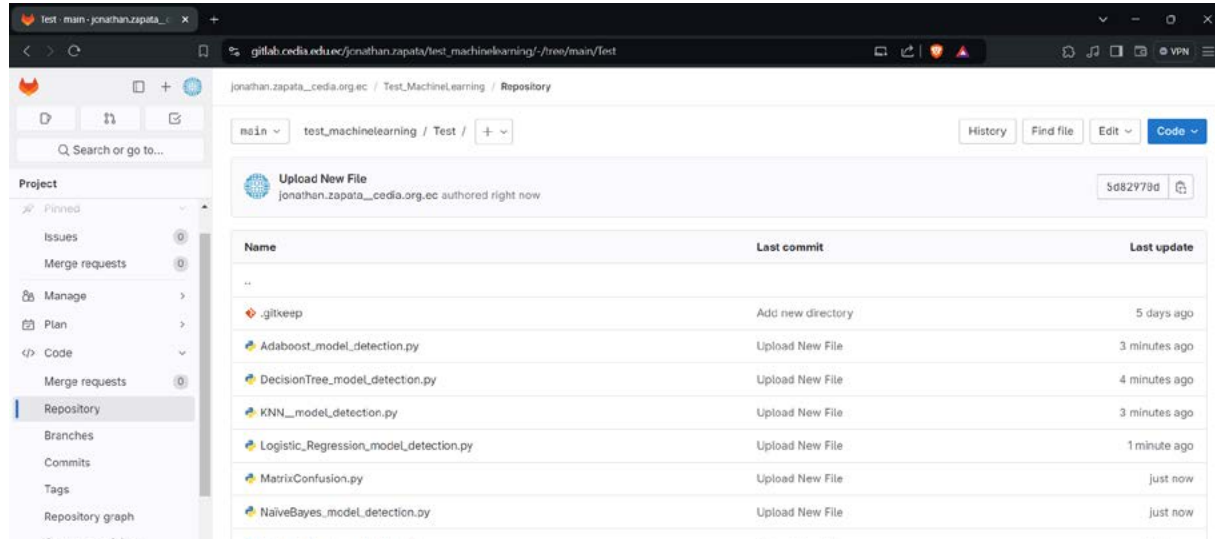


figura 48

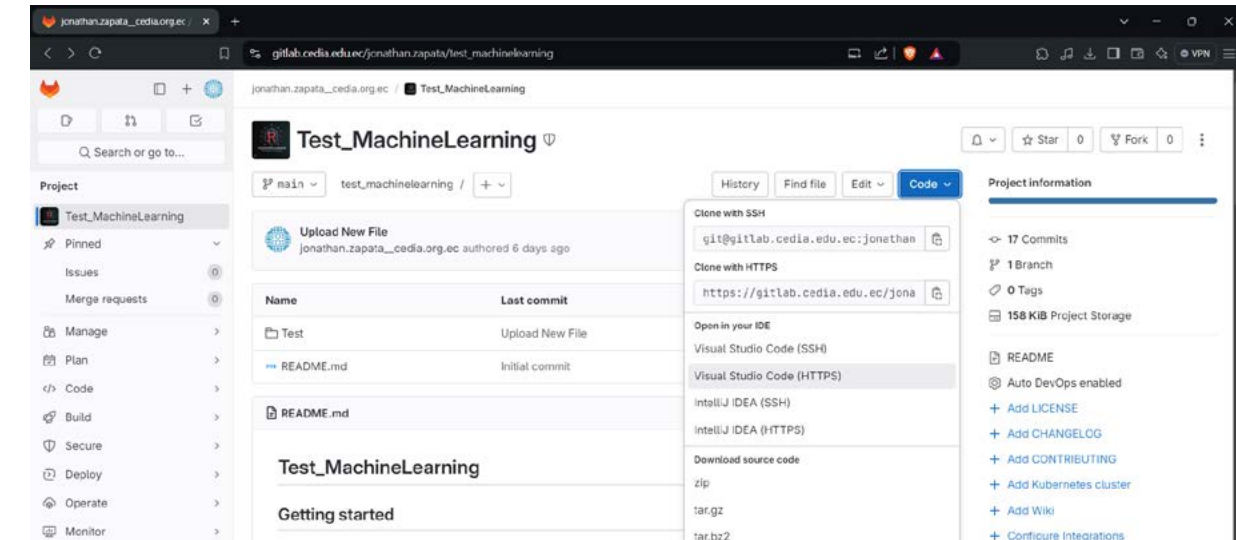


figura 50

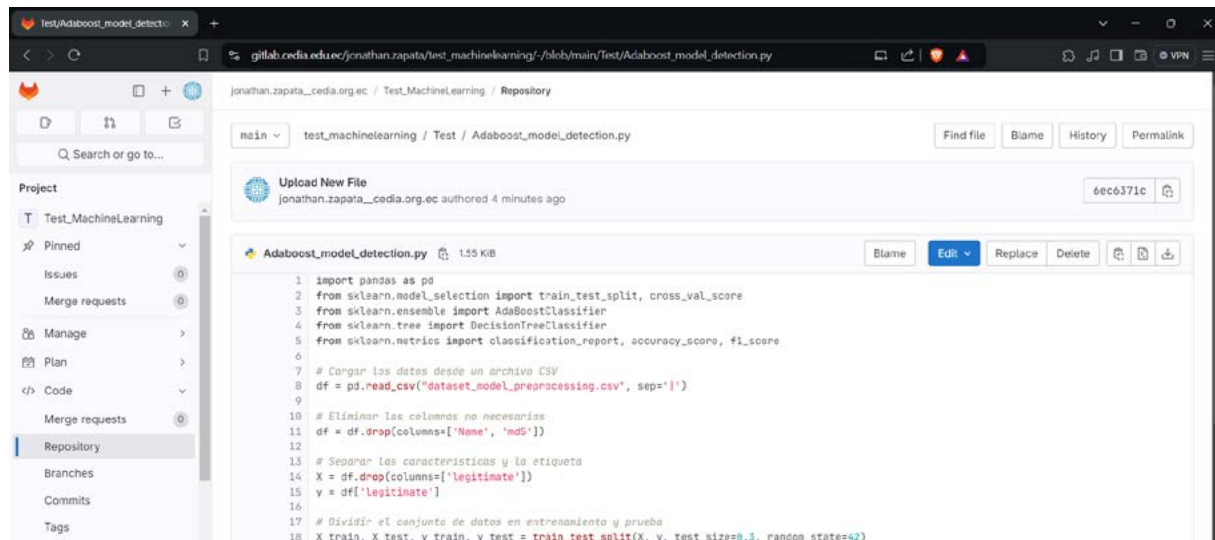


figura 49

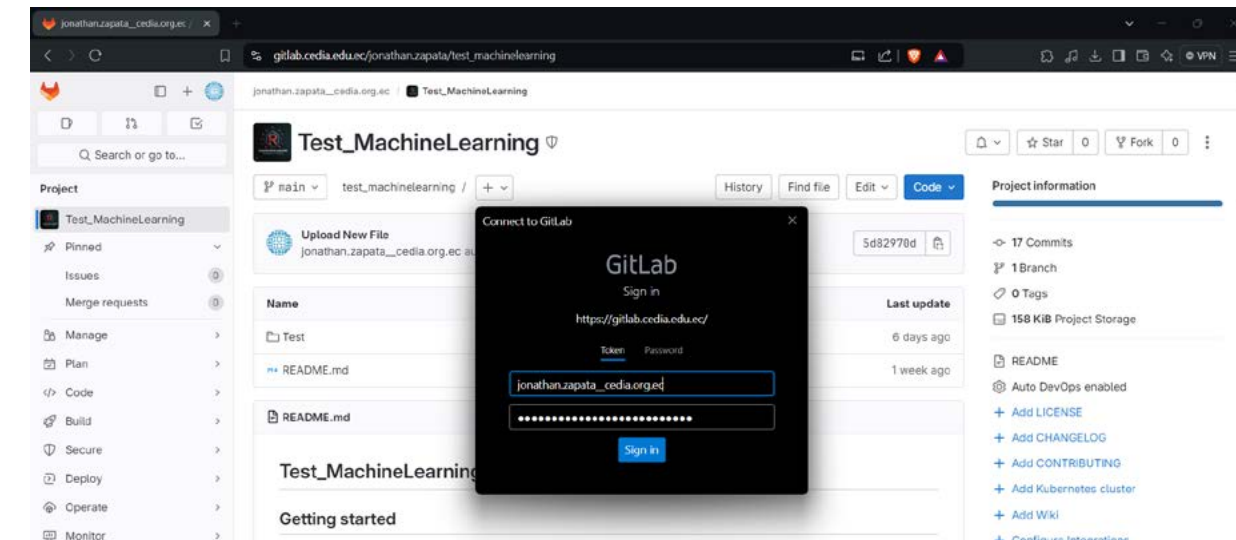


figura 51

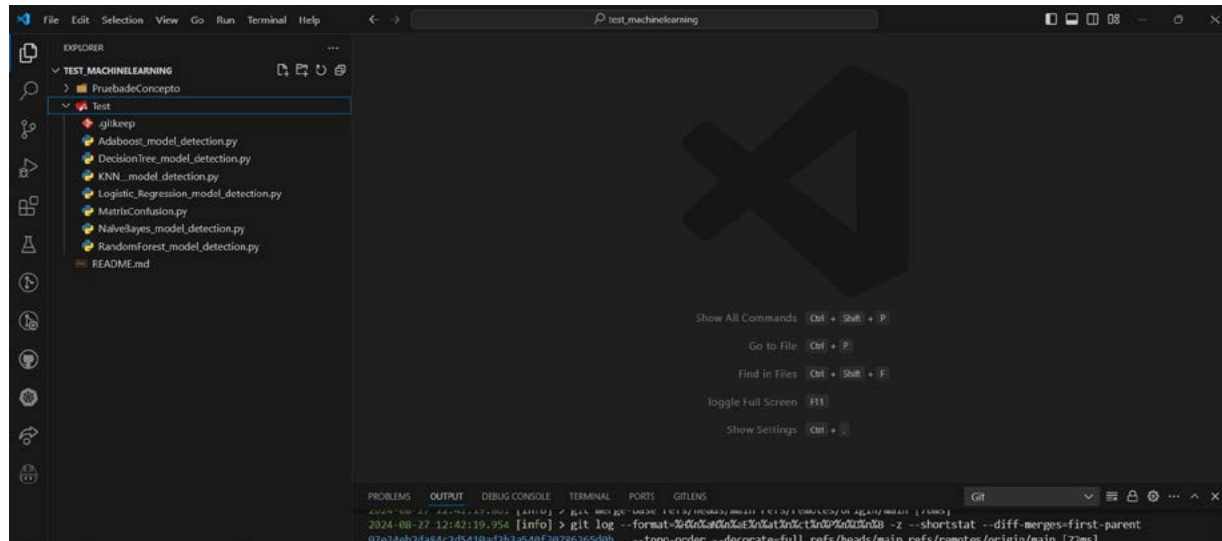


figura 52

Y creamos un pequeño prototipo usando python y Visual Studio CODE

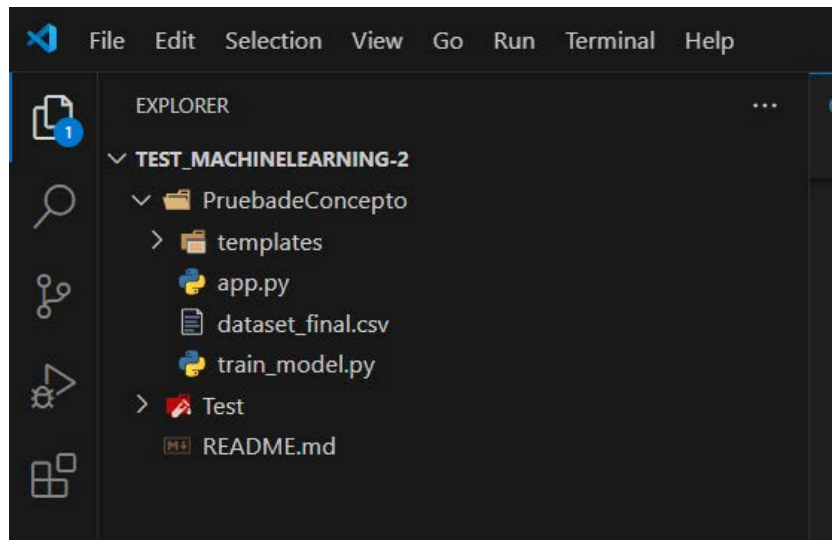


figura 53

Para crear los clasificadores o modelos de entrenamiento de machine learning y generar archivos '.pkl' que almacenan dichos modelos, utilizamos **Jupyter**.

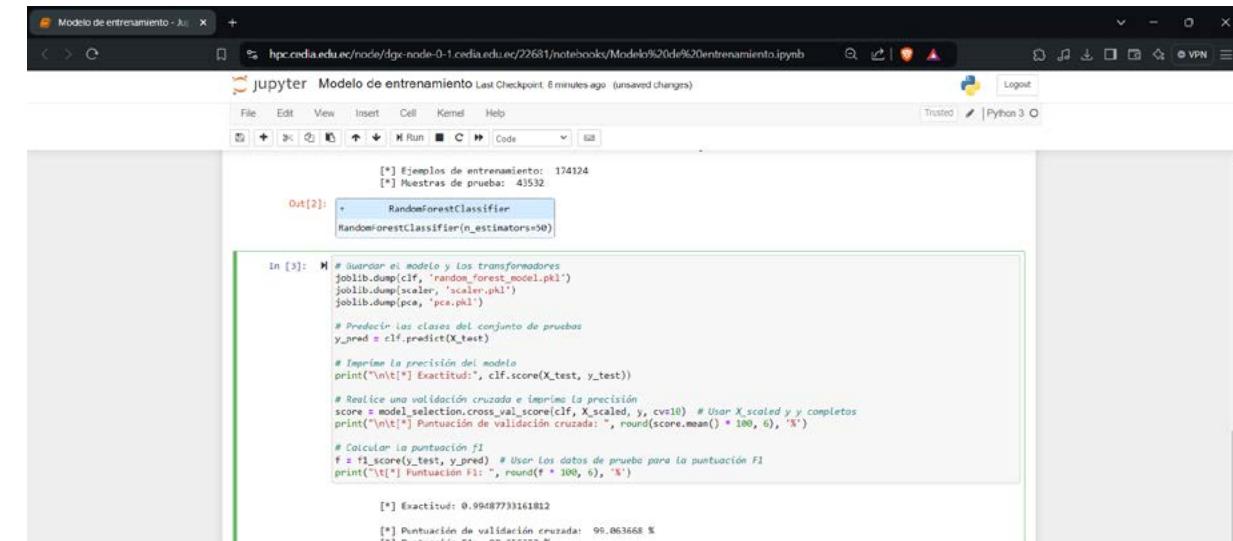


figura 54

Music	13 days ago	
Downloads	13 days ago	
Documents	10 days ago	
Desktop	13 days ago	
Prototipo.ipynb	8 days ago	69 kB
Modelo de entrenamiento.ipynb	Running 3 minutes ago	9.16 kB
Data Preprocessing.ipynb	8 days ago	1.65 kB
scaler.pkl	4 minutes ago	3.46 kB
random_forest_model.pkl	4 minutes ago	9 MB
pca.pkl	4 minutes ago	17.8 kB
dataset_model_preprocessing.csv	8 days ago	79 MB
dataset_model.csv	8 days ago	79 MB

figura 55

Y finalmente se lo puede descargar e implementar al prototipo

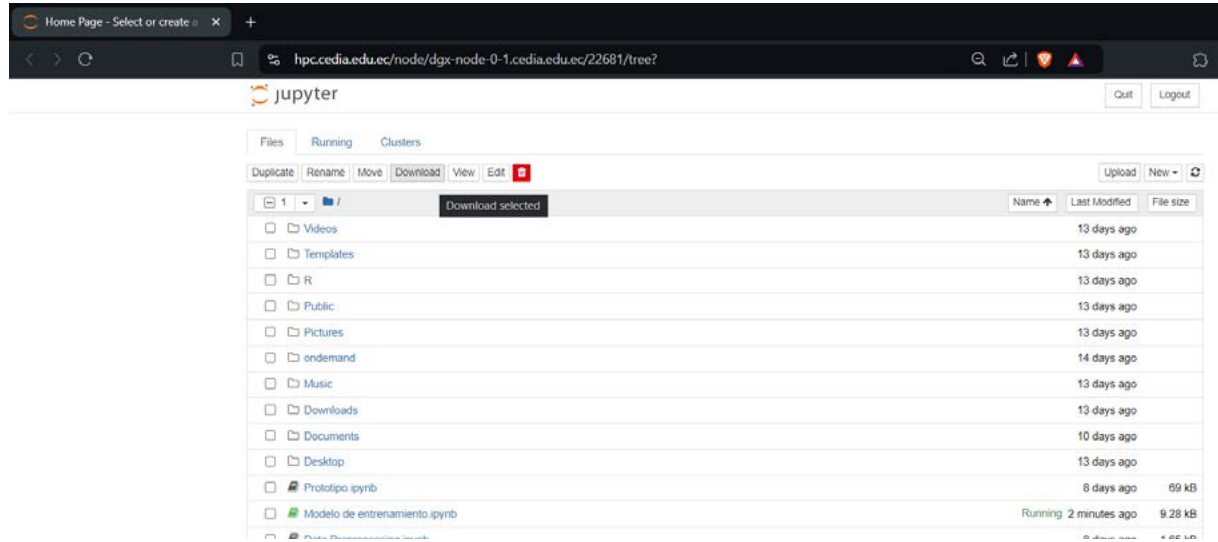


figura 56

Para maximizar el uso de **GitLab**, convertimos la investigación en una prueba de concepto, desarrollando una aplicación web. Además, establecimos una conexión directa con GitLab para gestionar el proyecto.



figura 58

Y ejecutamos el prototipo:

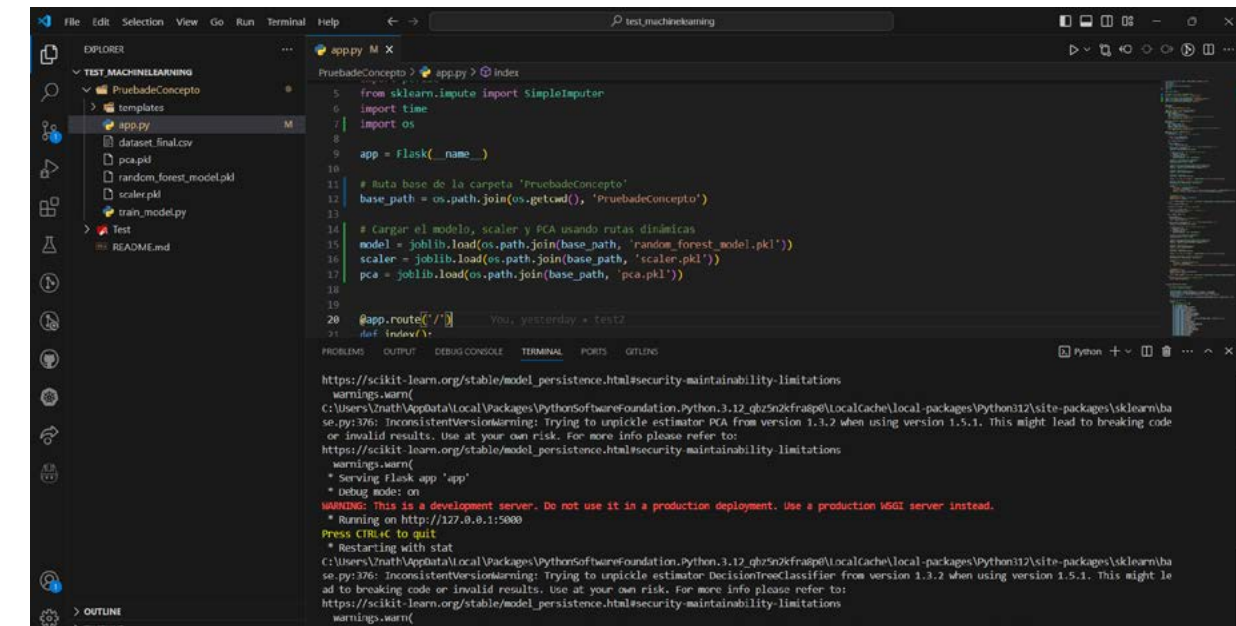


figura 59



figura 60

Para implementar la codificación en la plataforma Git, es fundamental considerar la colaboración y el trabajo en equipo, permitiendo que todos los integrantes del proyecto puedan contribuir y aportar de manera efectiva a la investigación.

Vamos a la **invitación de colaboradores**.

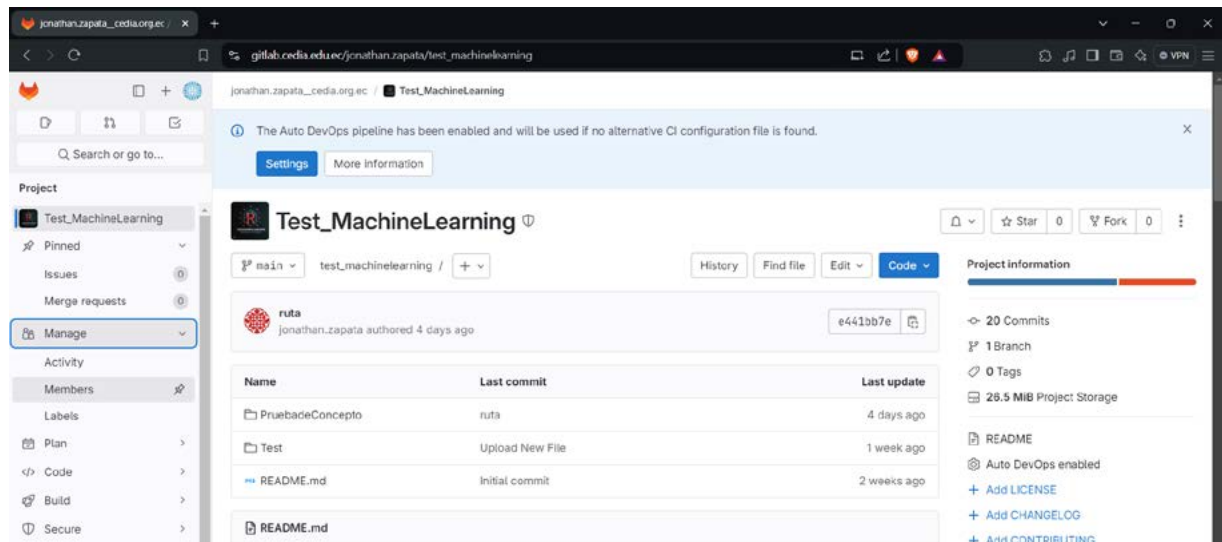


figura 60.1

Nos dirigimos a Manage >Members y finalmente Invite Members . A continuación, invita a todos los colaboradores que participarán en el proyecto.

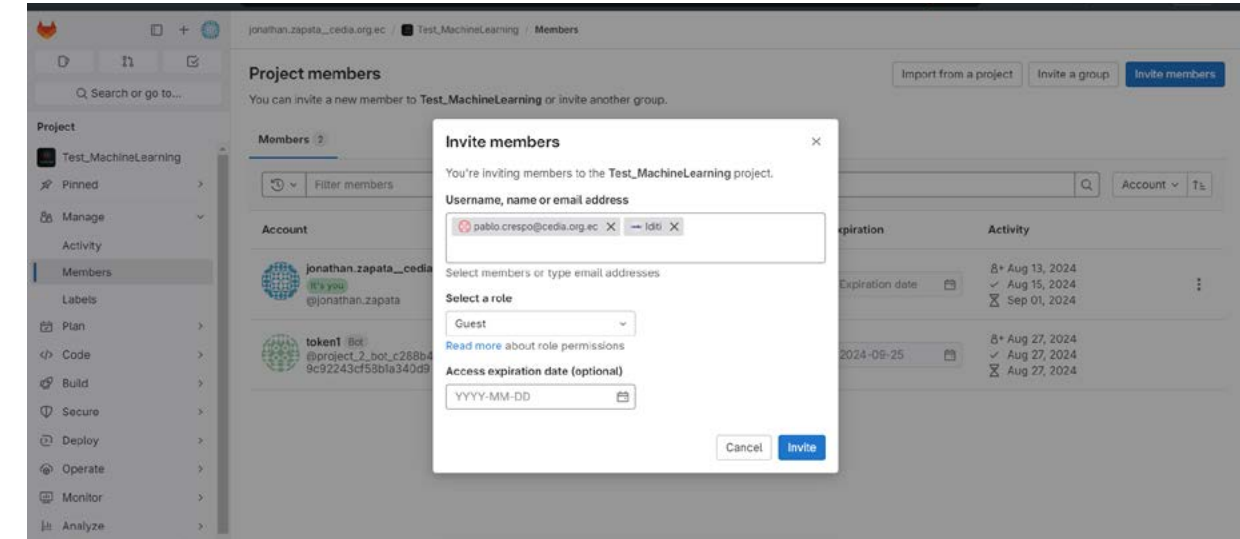


figura 60.2

Y aqui es muy importante el tipo de rol a designar que esta entre Guest, reporter, developer, maintainer y owner.

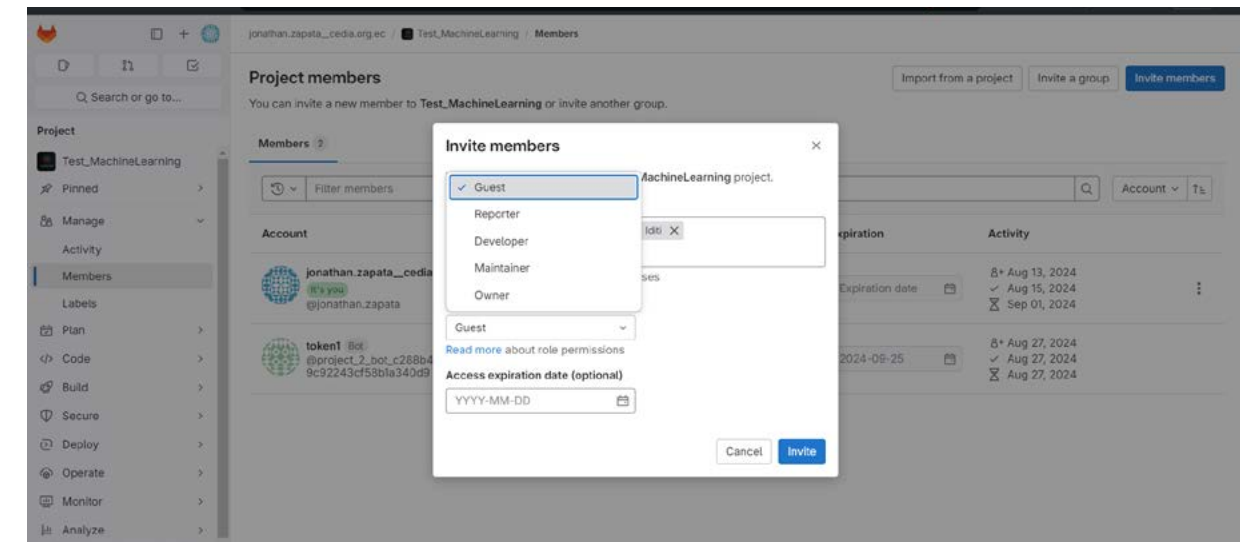
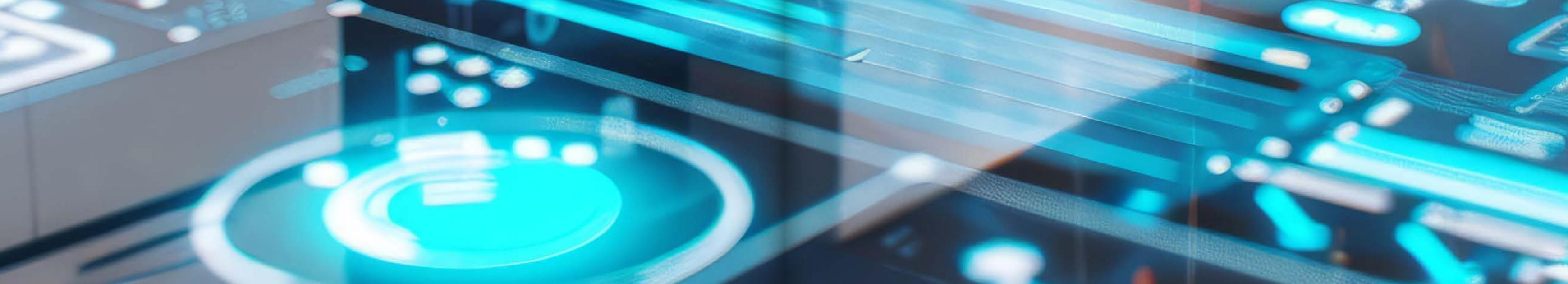


figura 60.3



En este caso, los designaremos como desarrolladores (developers), quienes se encargarán de codificar, modificar o mejorar el código, asegurando que los procesos de la investigación sigan el rumbo adecuado y se mantengan en la dirección correcta.

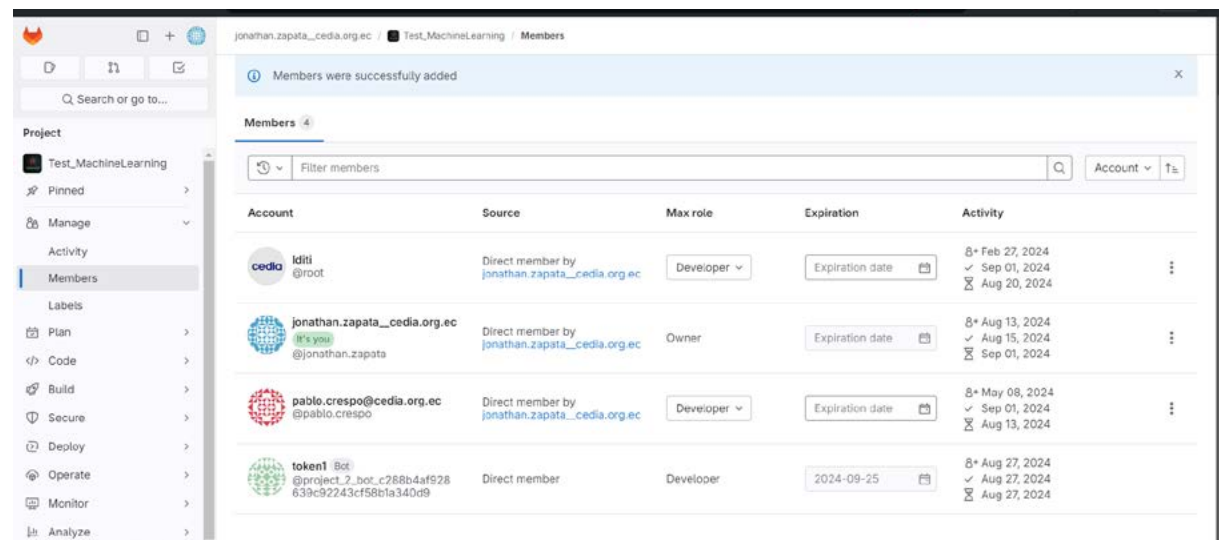


figura 60.4

Ahora procederemos a la asignación de permisos para cada colaborador en la rama principal. En GitHub, la rama principal (branch) representa la línea principal de desarrollo del código. Nos dirigimos a Settings > Repository y configuramos los permisos para que tanto los mantenedores (maintainers) como los desarrolladores (developers) tengan acceso a la rama principal.

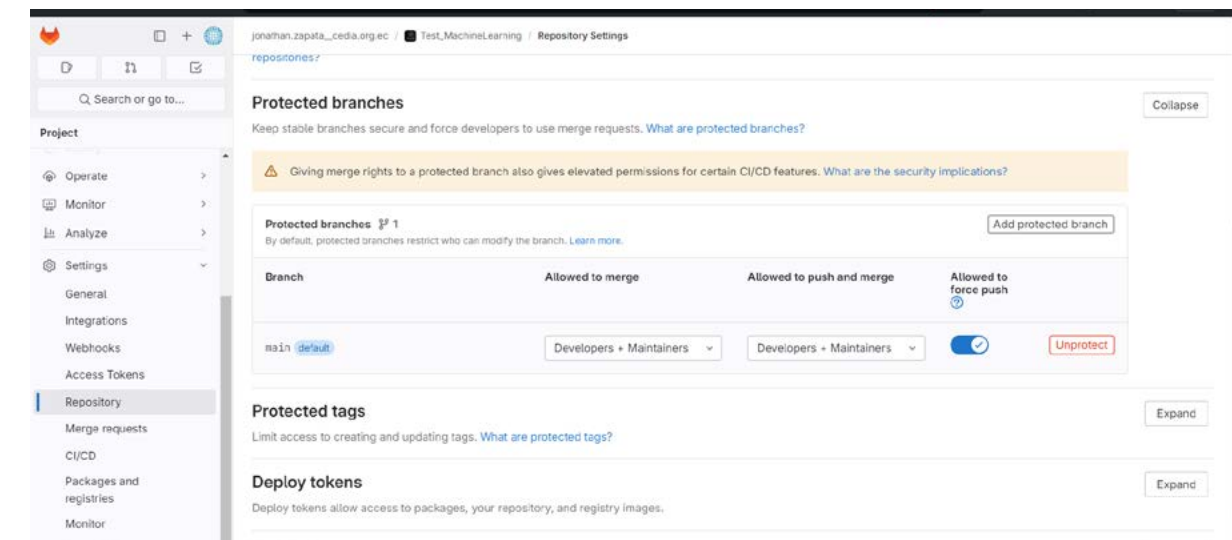


figura 60.5

También es posible crear ramas protegidas, permitiendo que solo el propietario o líder del proyecto tenga acceso para modificarlas. Esto asegura que las ramas críticas del desarrollo estén bajo control, evitando cambios no autorizados.

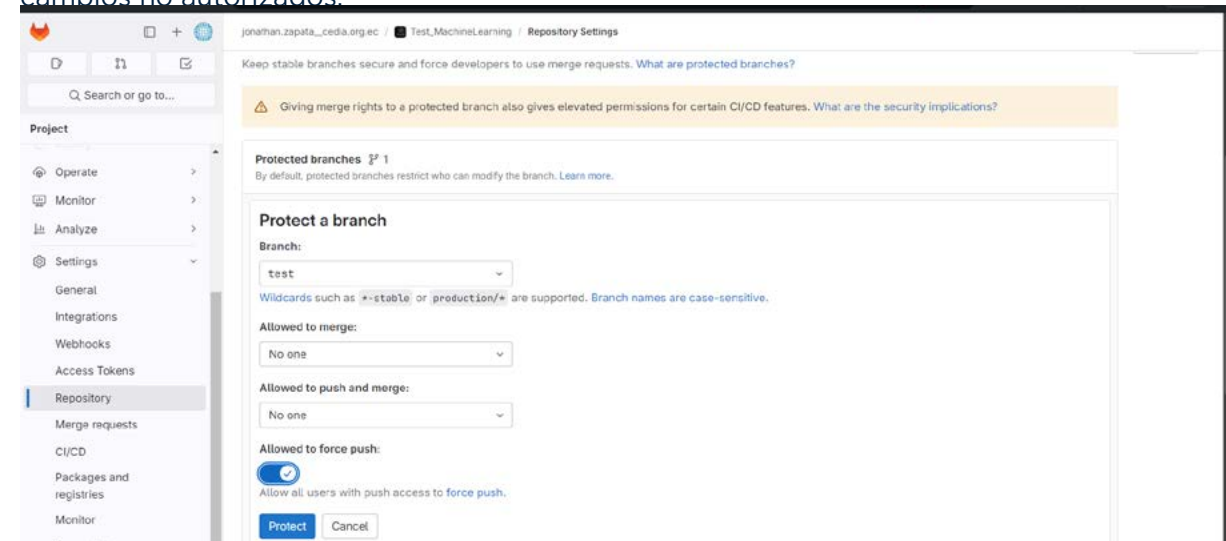


figura 60.6

Finalmente, para la revisión o la designación de un revisor en la investigación, se le puede asignar un token de acceso en caso de que el repositorio sea privado. Este token le permitirá ingresar al repositorio de Git, y se puede generar en la sección de Settings > Access Tokens.

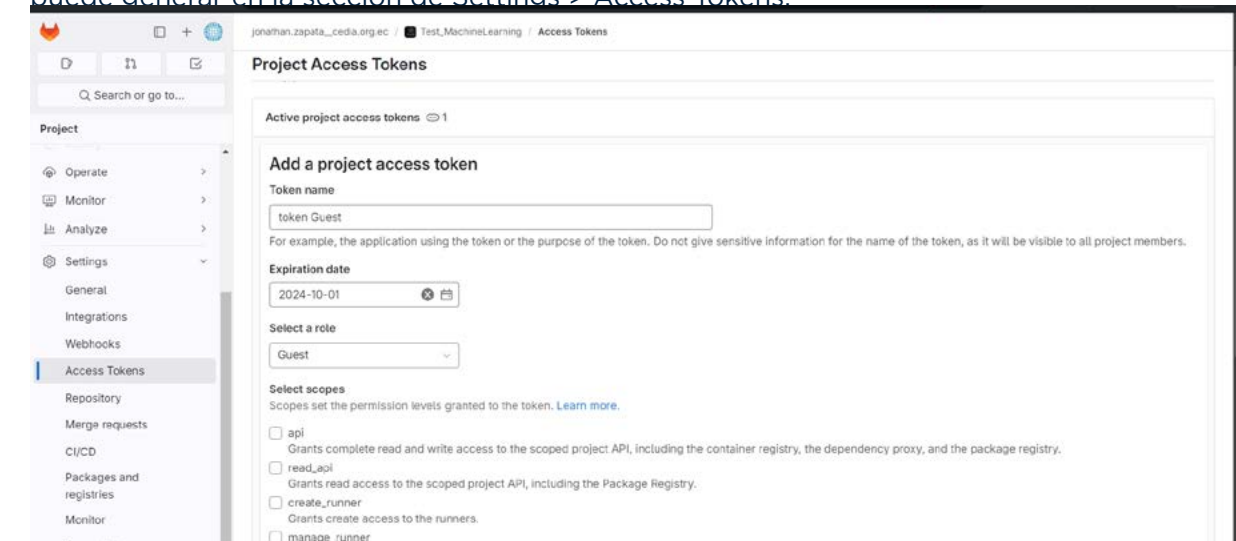


figura 60.8

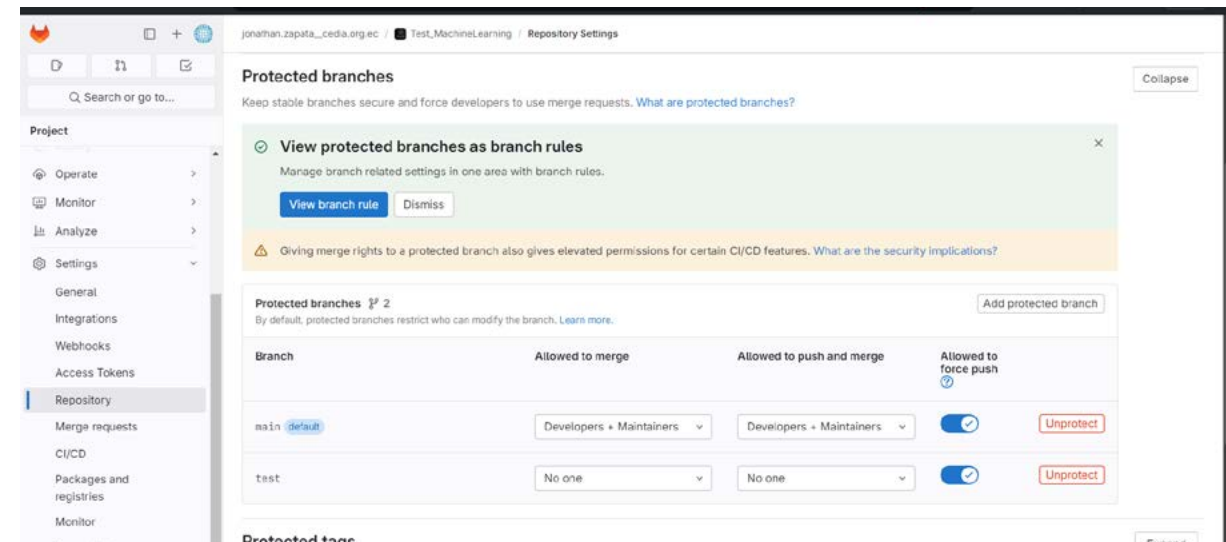


figura 60.7

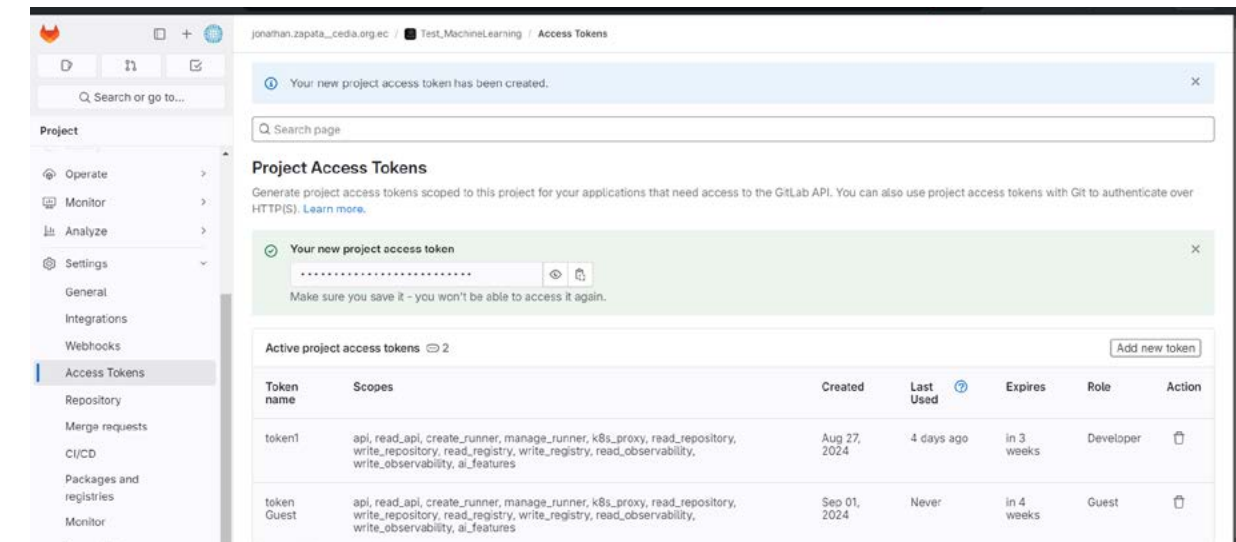


figura 60.9

Una vez se le entrega el token correspondiente, el revisor podrá clonar el repositorio y tener acceso al código.

1.5

USO DE LA HERRAMIENTA MATTERMOST



Finalmente, se utiliza Mattermost como la herramienta principal de comunicación entre los participantes de la investigación. Esta plataforma permite coordinar reuniones para discutir los resultados y el avance del proyecto. Se configura un canal con el nombre del proyecto o una palabra clave relevante, como “Machine Learning Ransomware”, y se invita a los colaboradores involucrados en la investigación para facilitar la colaboración y el intercambio de ideas en tiempo real.

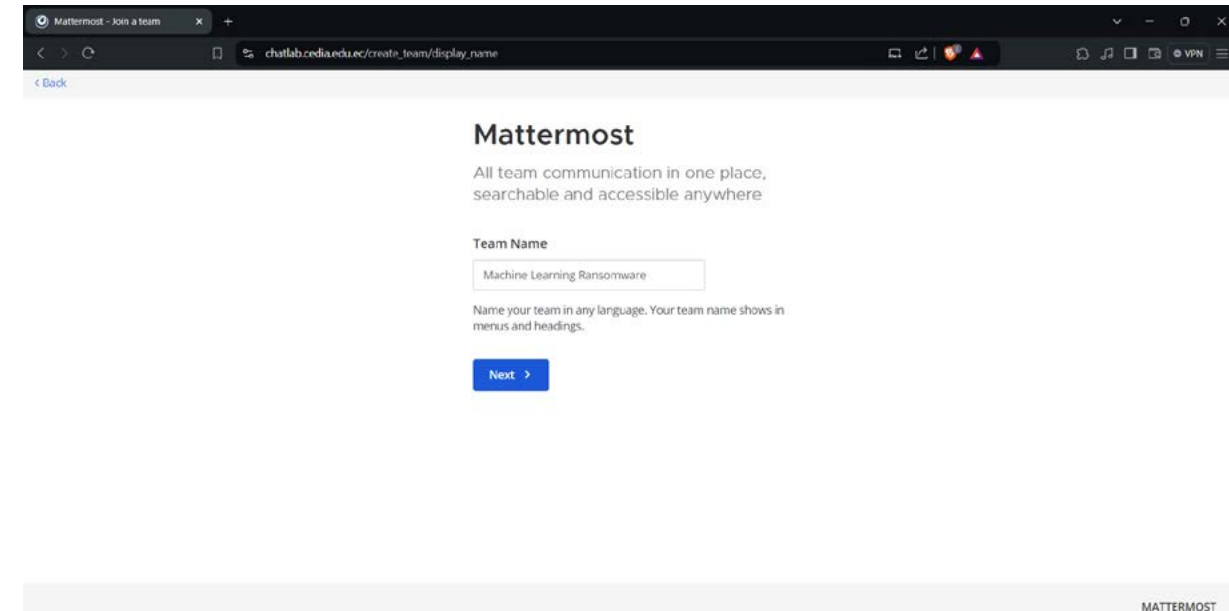


figura 61

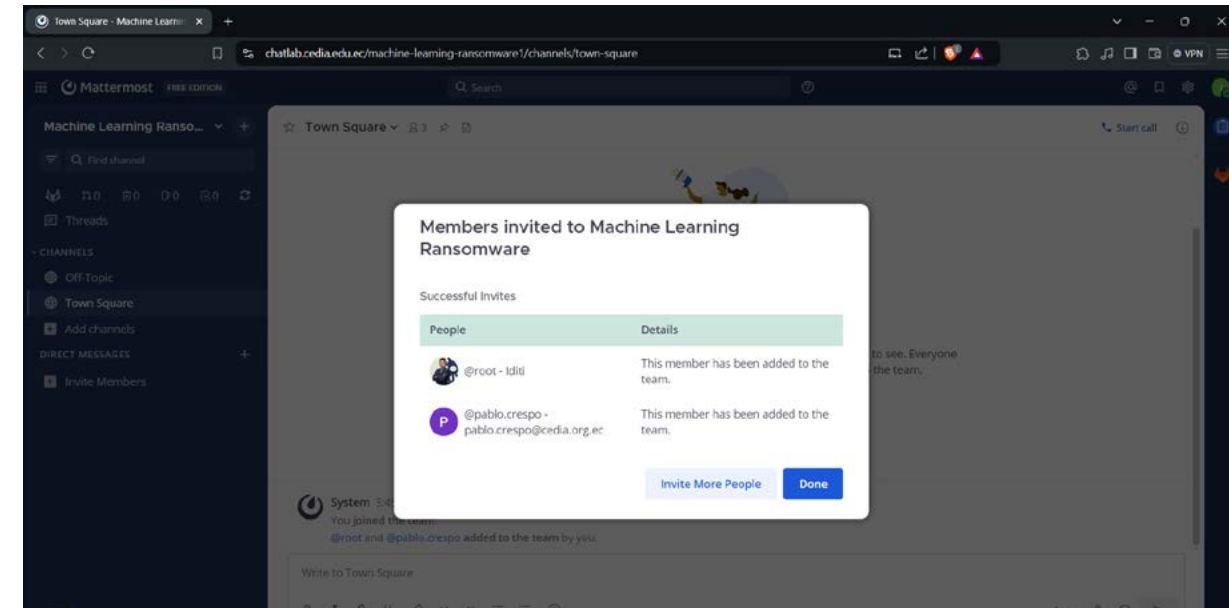


figura 62

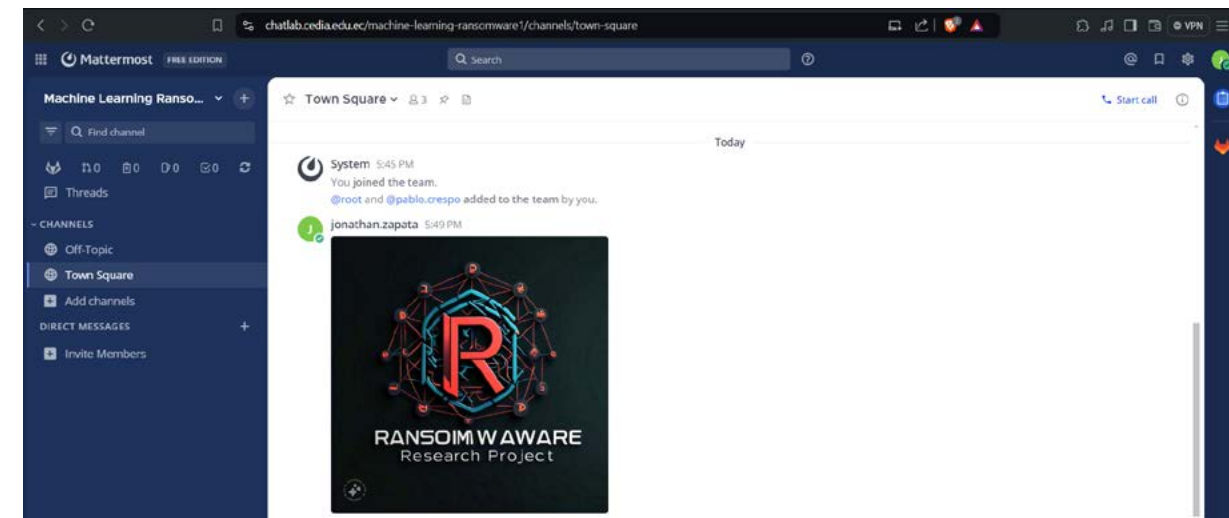


figura 63

Mattermost ofrece una amplia gama de herramientas, como la personalización del perfil, notificaciones, configuración de estado, y la capacidad de identificar y unirse a canales públicos o privados, así como enviar mensajes directos.

Además, es posible compartir código en formato Python, lo que permite a los investigadores visualizar el código que se está utilizando y evaluar posibles cambios o mejoras de manera más efectiva.

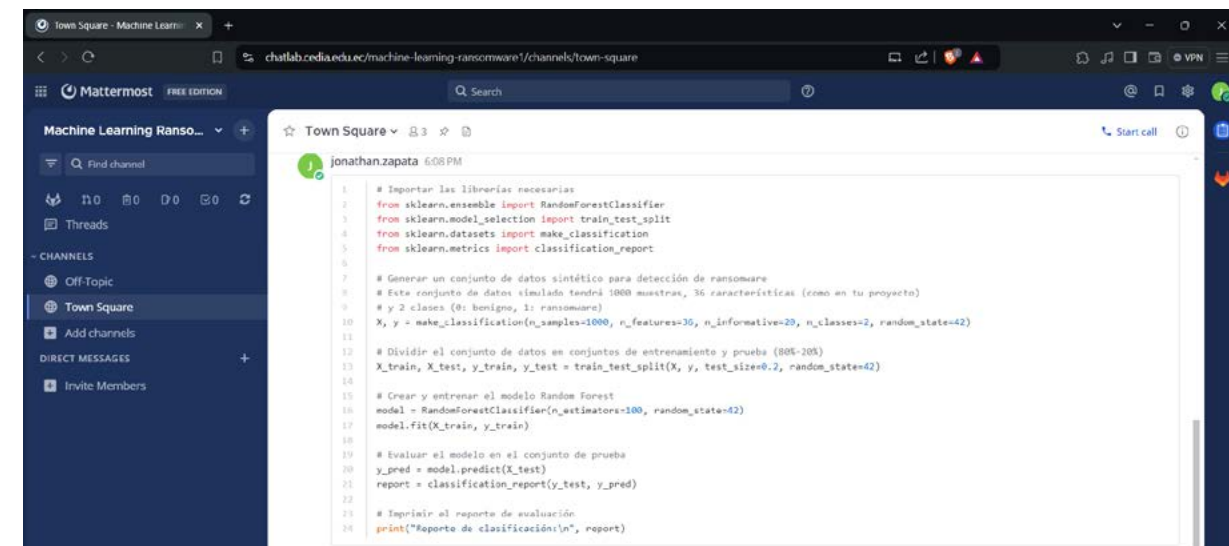


figura 64

Creación de PLAYBOOKS

Colección de procedimientos y directrices documentadas que guían a los equipos en la gestión de tareas o procesos repetitivos

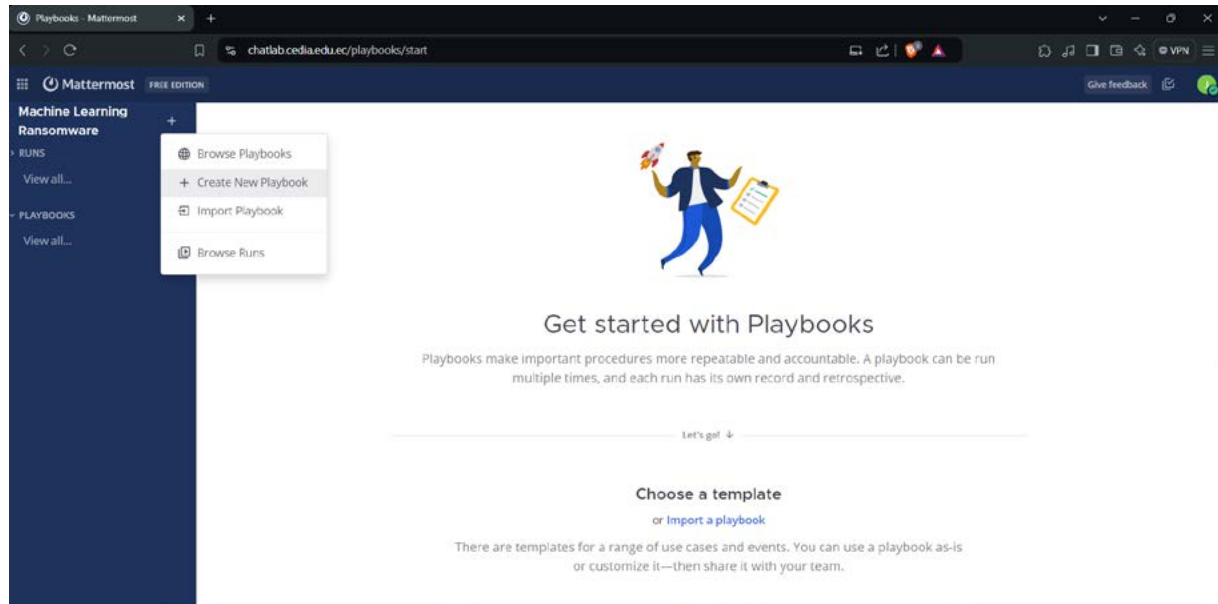


figura 65

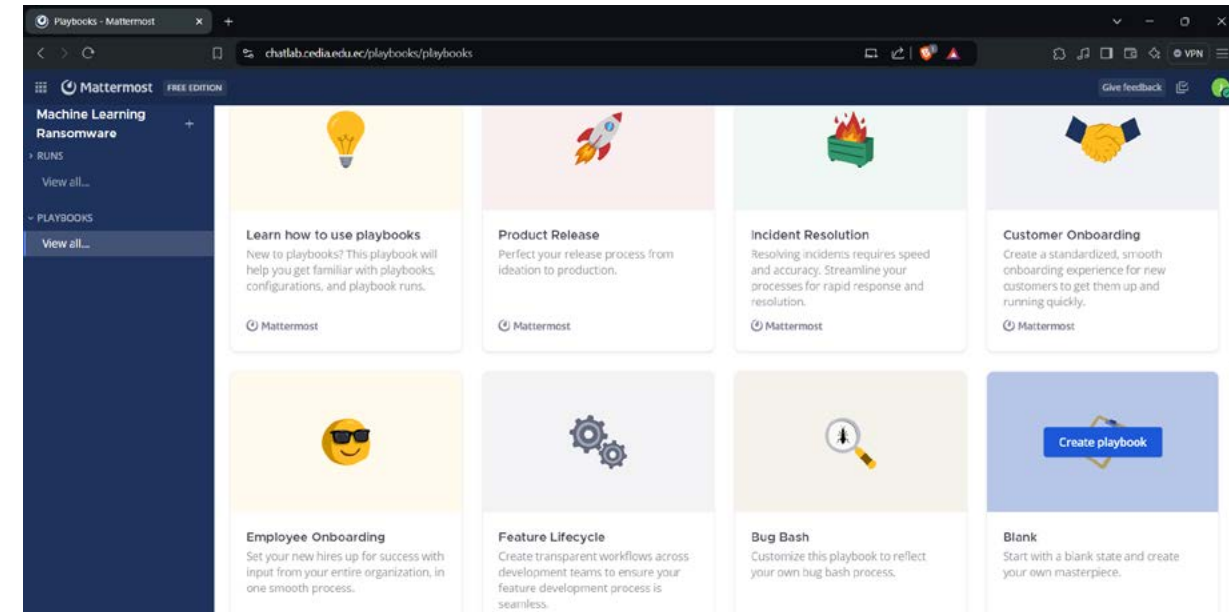


figura 66

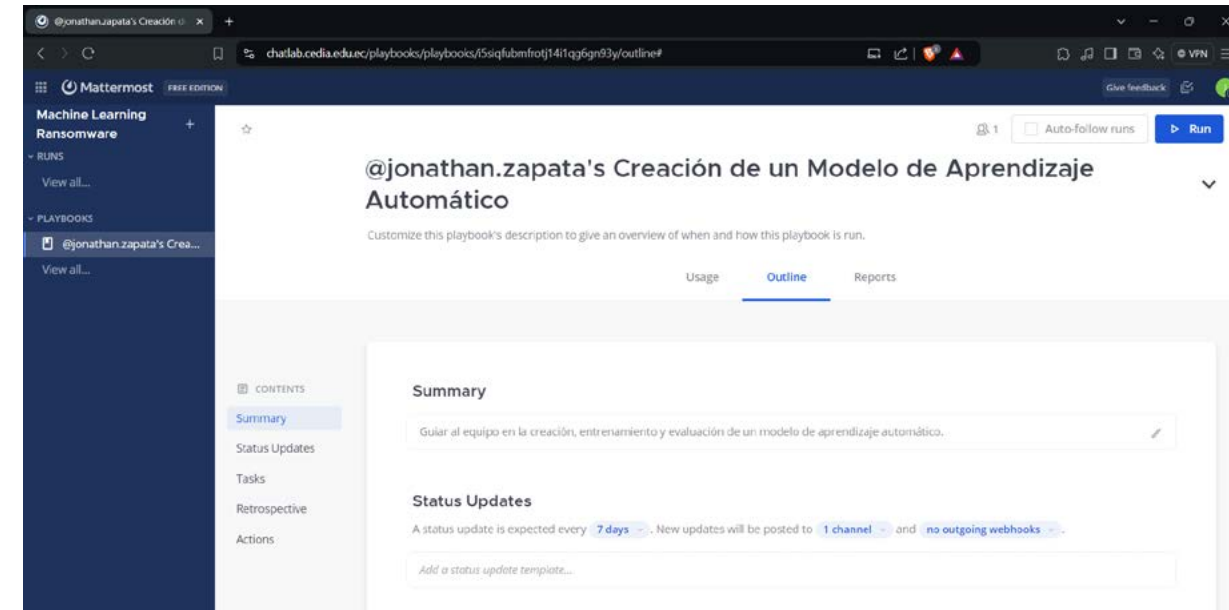


figura 67

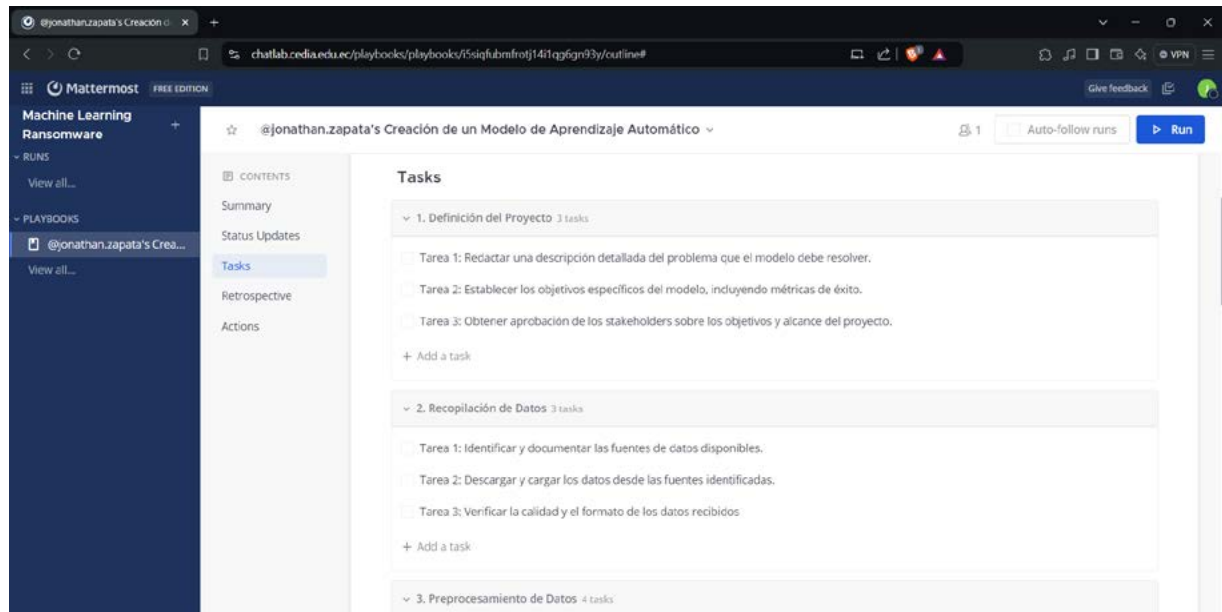


figura 68

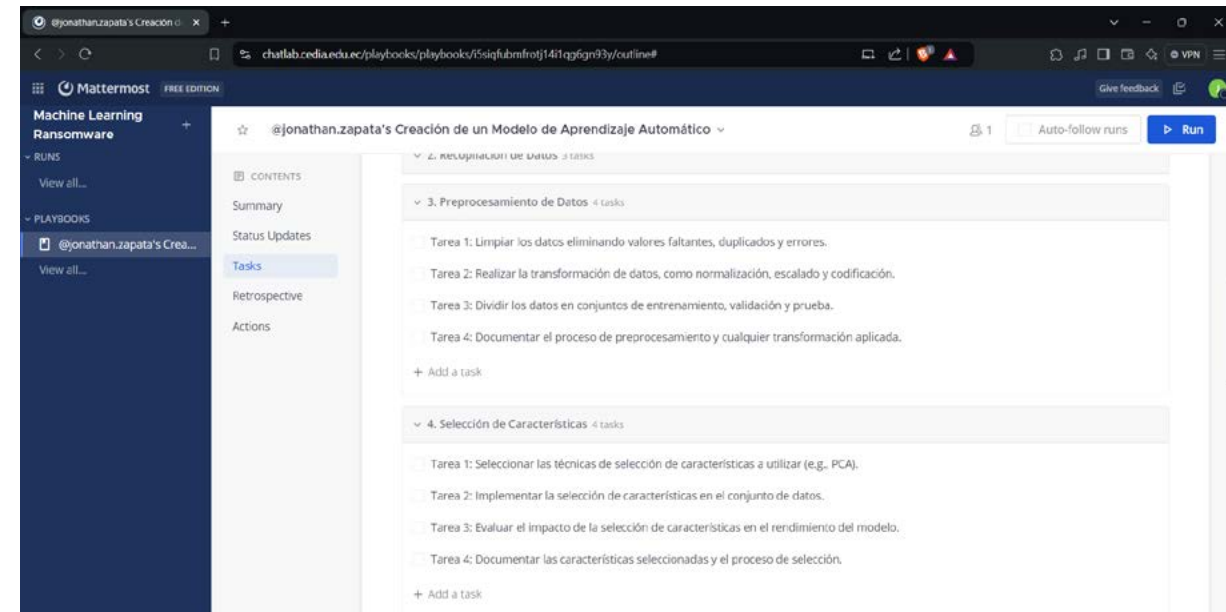


figura 70

Finalmente lo ejecutamos

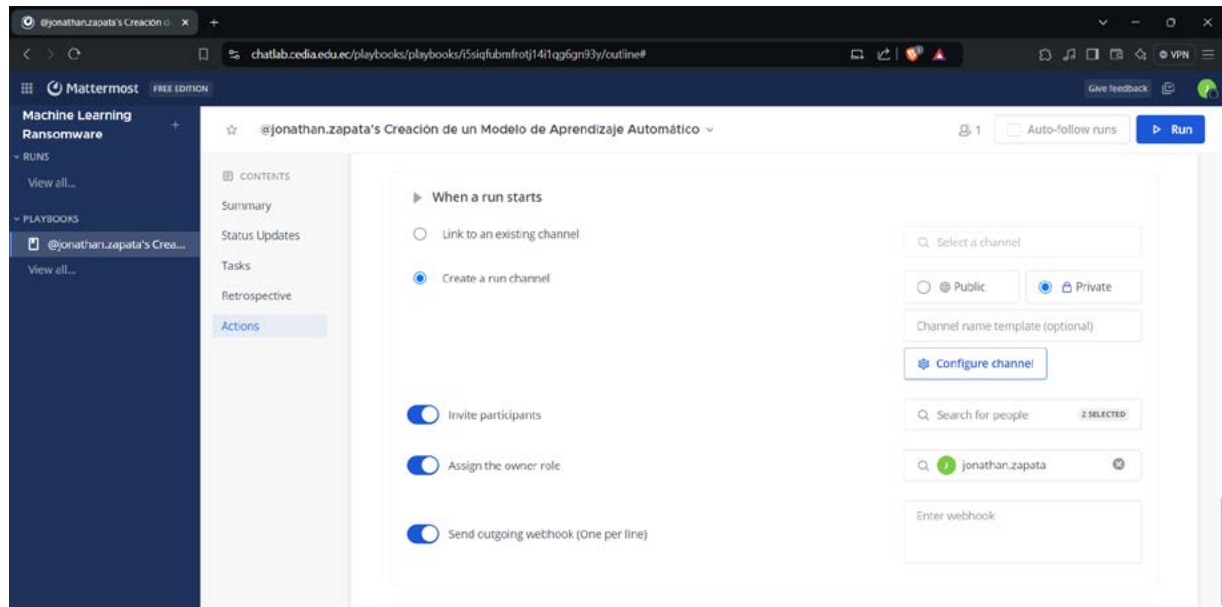


figura 69

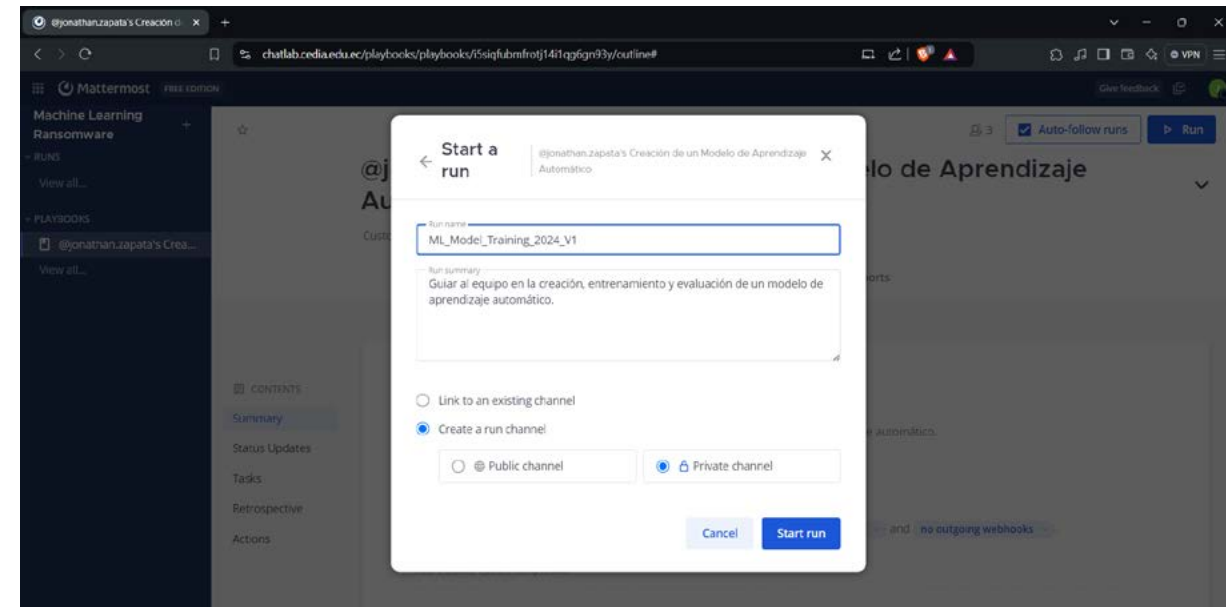


figura 71

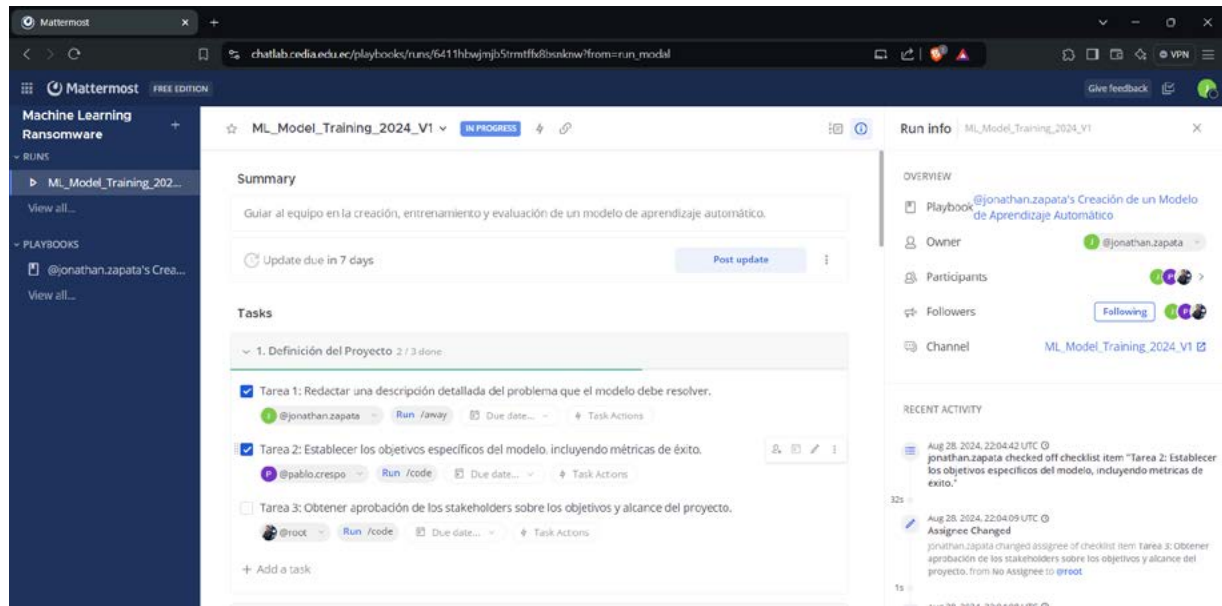


figura 72

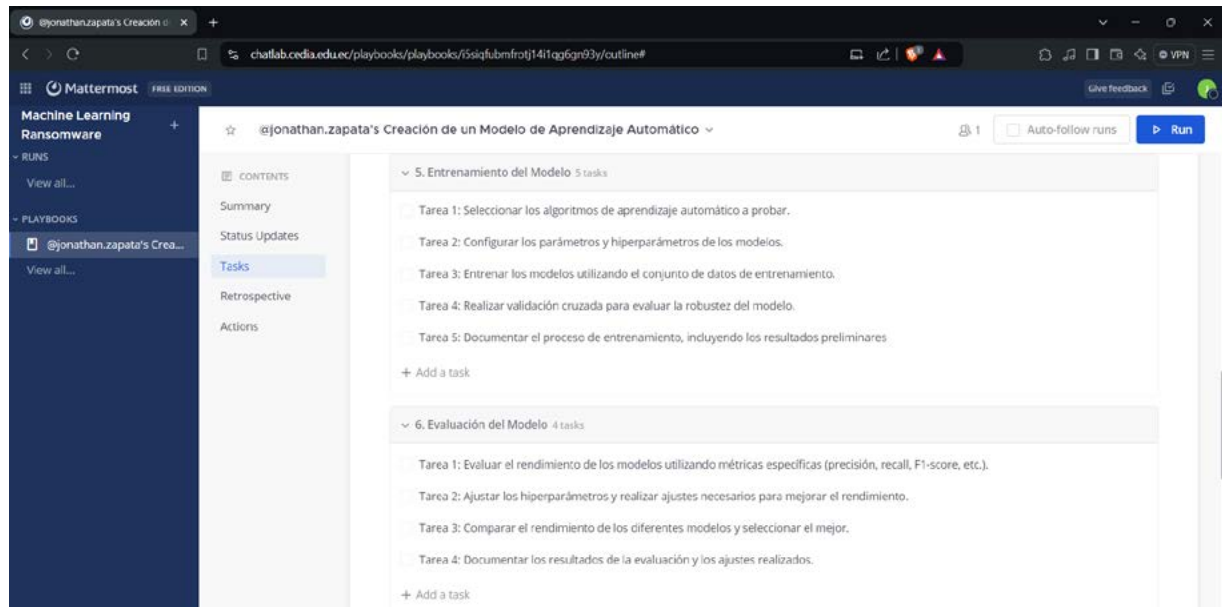


figura 73

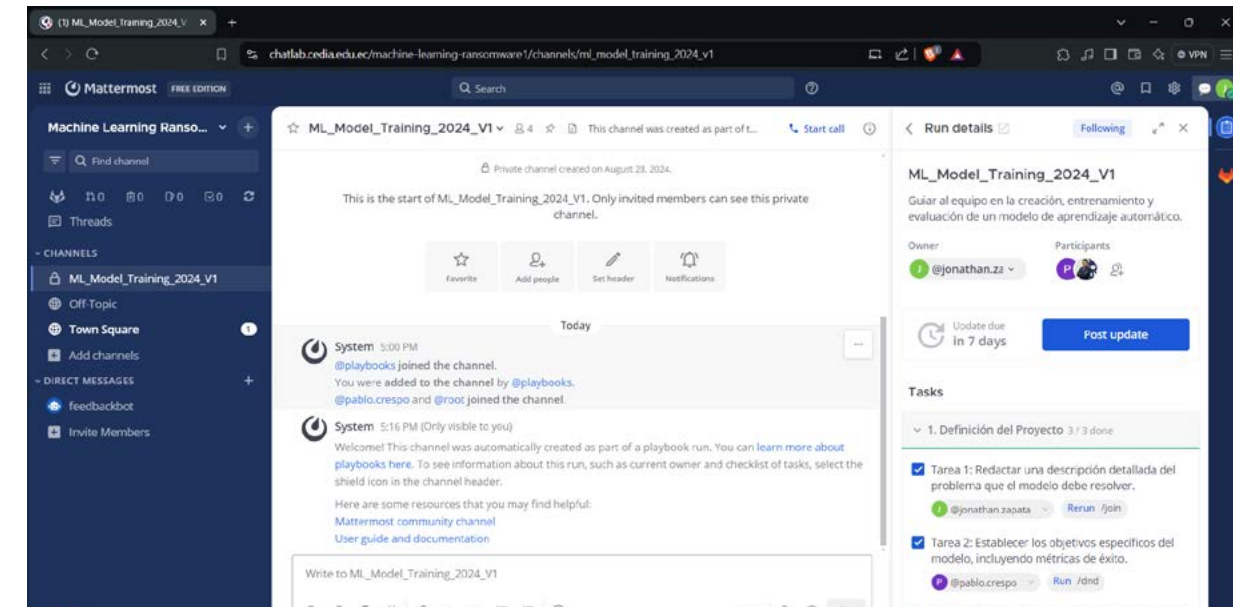


figura 74

Así tenemos un playbook bien elaborado que actúa como una guía integral que organiza y facilita el proceso de desarrollo de un modelo de aprendizaje automático, mejorando la eficiencia, la comunicación y la calidad del proyecto, del mismo modo podemos editar el playbook editando o agregando más tareas.

En este caso configurado a una actualización en 7 días, del mismo modo registrar las tareas ya hechas y realizar un seguimiento de estas, y con la designación de tareas a cada uno de los integrantes que fueron invitados al grupo.

En el slash command ubicamos el código /dnd que significa “La función No molestar está activada. No recibirás notificaciones push en tu computadora de escritorio o en tu dispositivo móvil hasta que la función No molestar esté desactivada.

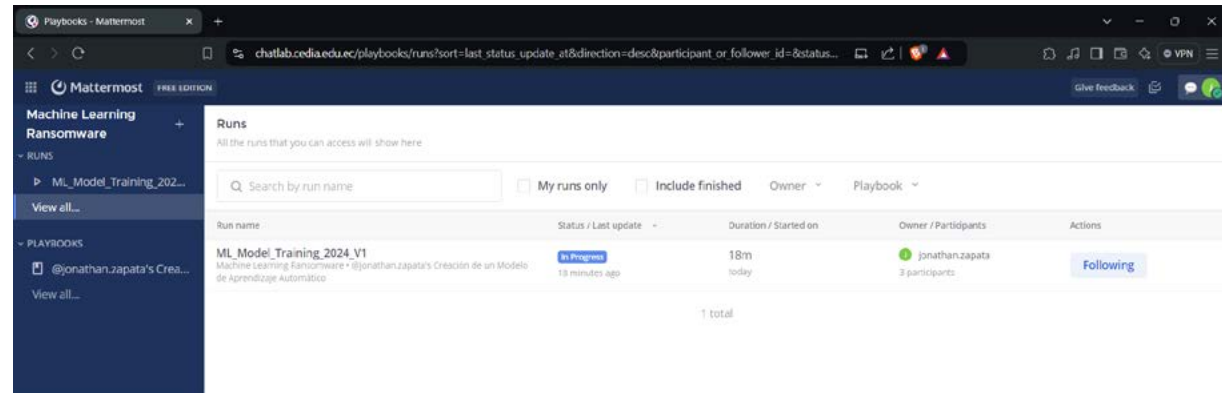


figura 75

Asimismo, Podemos publicar una actualización del proyecto

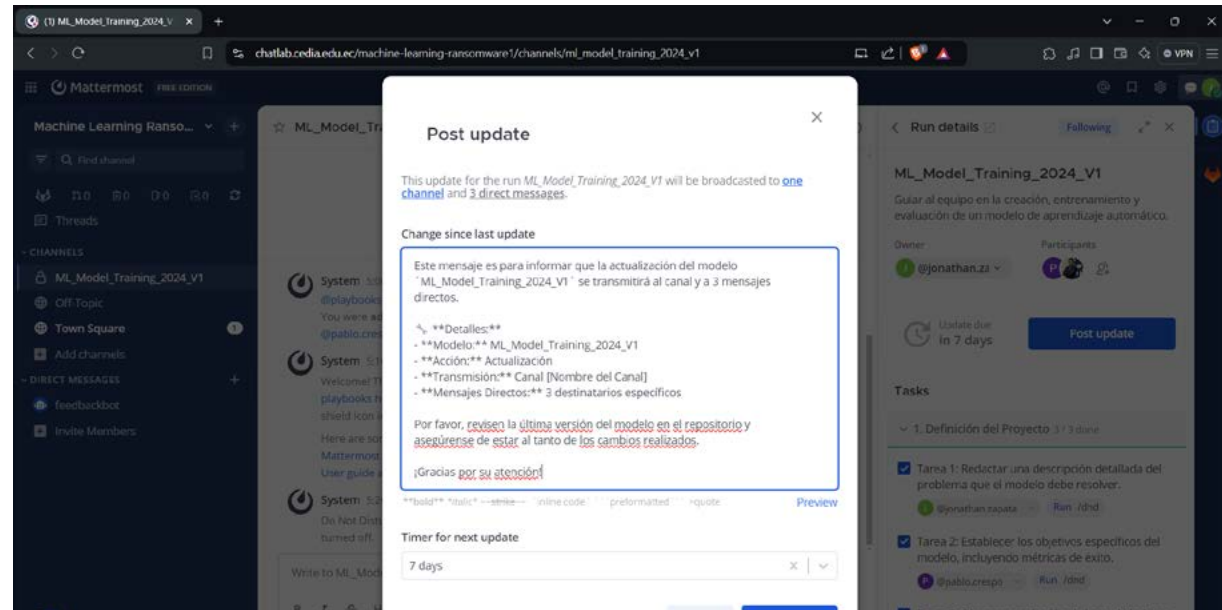


figura 76

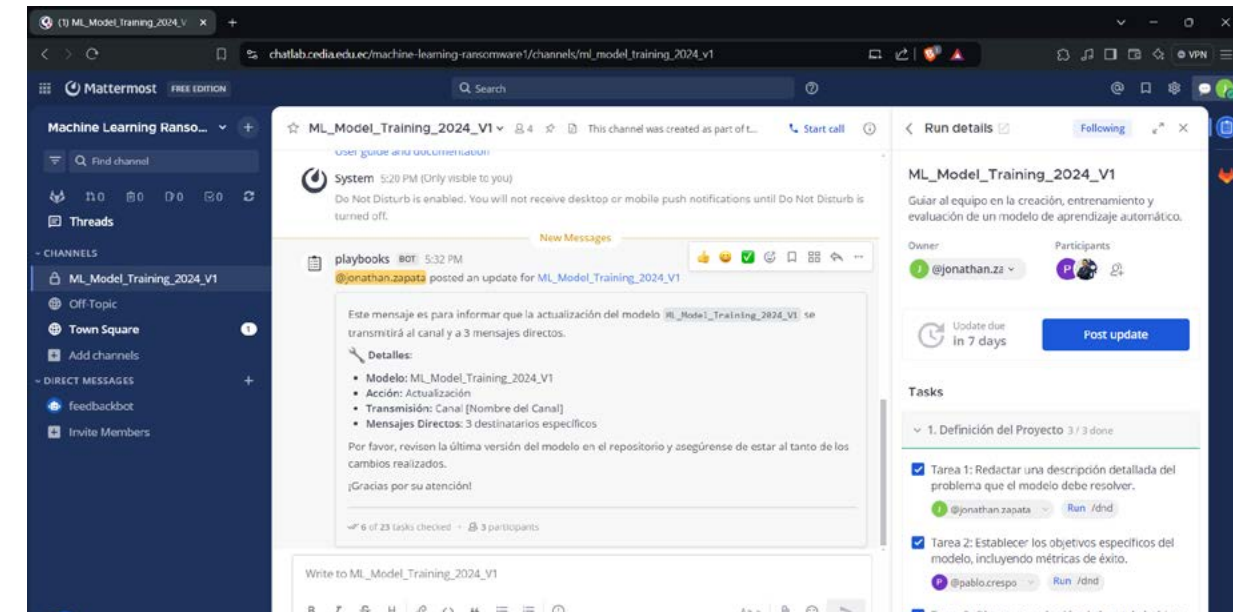


figura 77

Y se trasmitirá al canal principal

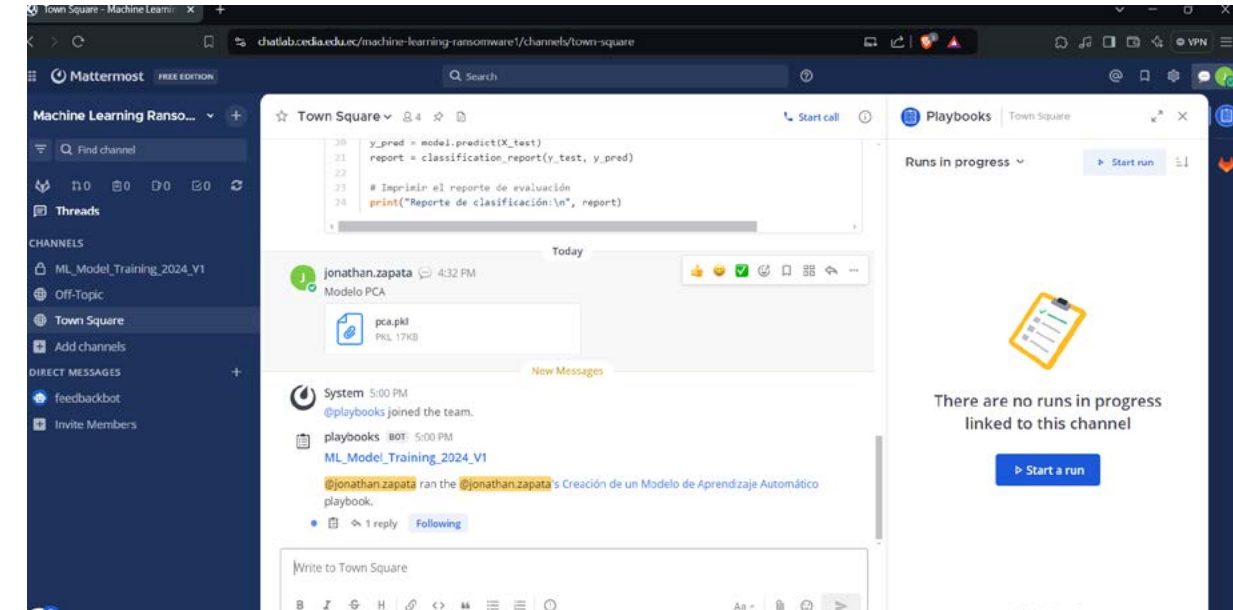


figura 78

1.6

USO DE LA HERRAMIENTA FILESENDER



filesender

Al utilizar la herramienta Filesender para enviar archivos de gran tamaño, designamos el archivo que queremos compartir—en este caso, nuestro dataset de aproximadamente 75.3 MB. Establecemos la fecha límite de expiración como el 11/09/2024 y procedemos a enviar el archivo al destinatario.

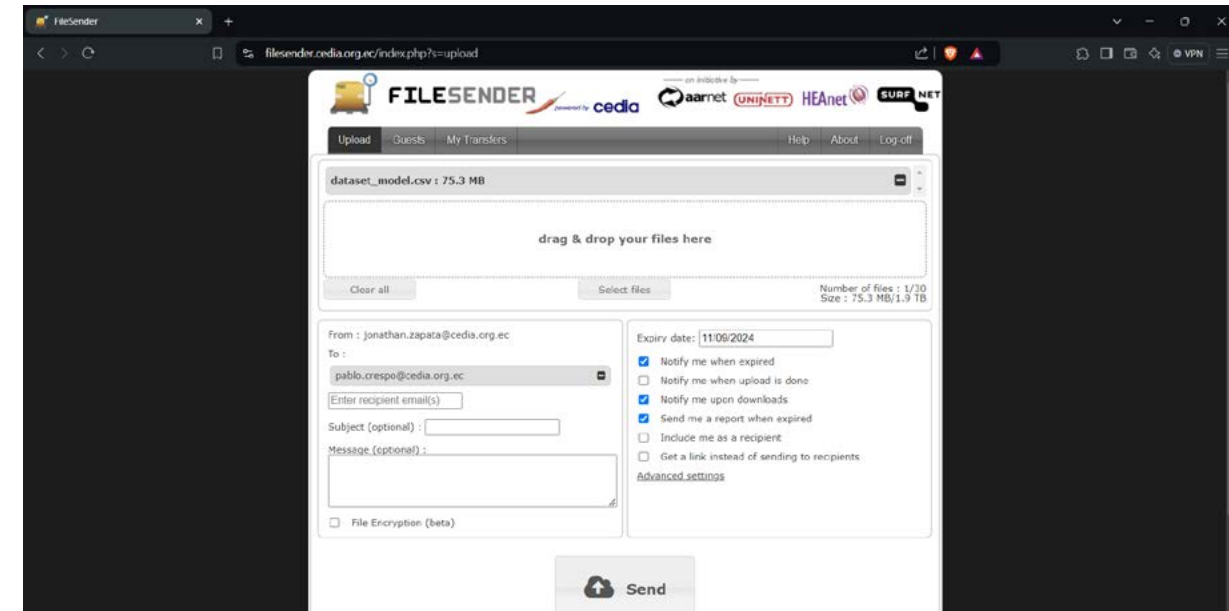


figura 79

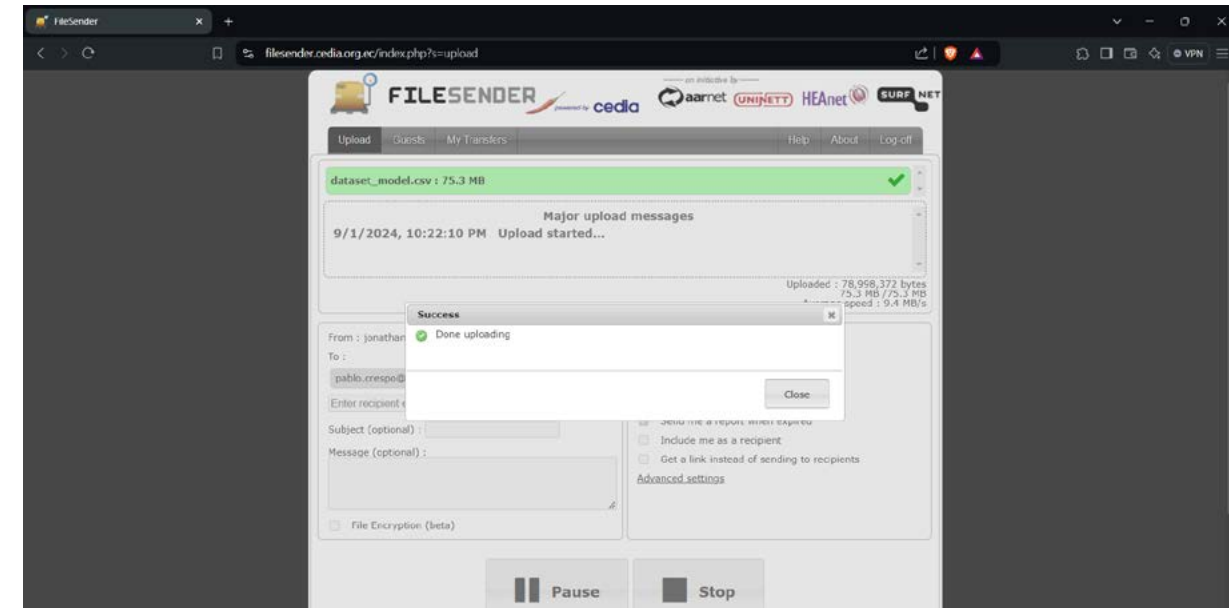


figura 80



research
lab
by cedia

cedia

*“Conectando ideas,
transformando
sociedades”*





cedia.edu.ec

@CEDIAec



CUE

Oficinas_
Gonzalo Cordero 2-122
y J. Fajardo esquina.
DIGITALS_
Miguel Moreno y Av. 10
de Agosto

UIO

Av. 12 de Octubre y
Lizardo García
Edificio Alto Aragón
Oficina 8A

GYE

Av. Del Bombero,
Km 6.5 - Edificio
La Vista de San Eduardo
5to piso Ofics. 510 y 511

PORTOVIEJO

Av. Metropolitana
Eloy Alfaro #2005
y Av. Olímpica.
Univ. San Gregorio

MANTA

Av. Circunvalación
Vía a San Mateo
ULEAM - EP