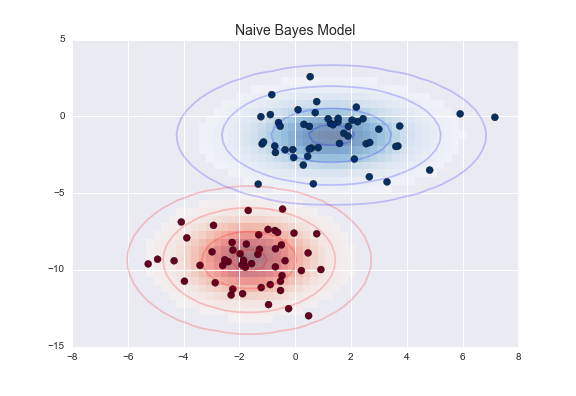
Naive Bayes en diferentes IDE



Adrián Yared Armas de la Nuez

**Contenido**

[**1. Enunciado 2**](#_2c35568x6wg8)

[**2. Ejecución en Visual Studio Code 2**](#_mptzybqzhbp3)

[**3. Ejecución en R Studio 4**](#_es22nskkykre)

[**4. Ejecución en Google Colab 6**](#_hc2sqxb6ez8g)

[**5. Ejecución en Anaconda notebooks – Jupyter 7**](#_qa3dx5rvvmrq)

[**6. Ejecución en Anaconda - R Studio 8**](#_y27cr9bc24xb)

[**7. Ejecución en Anaconda - Spyder 9**](#_x71lnli2xp5)

[**8. Ejecución en Pycharm 10**](#_agau4vqgy9vy)

[**9. Ejecución en Replit 12**](#_s02gilwjyy4o)

[**10. Ejecución en Binder 13**](#_39trwrvzhfwu)

[**11. Ejecución en Kaggle 15**](#_2toxmh3pfda5)

[**12. Código R 16**](#_krru84n6o63i)

[**13. Código Python 19**](#_qzg3dhmzlpjm)

[**14. Código Kaggle 21**](#_d6eh18fgg0qi)

[**14. Github 24**](#_kupbp3vwdn1q)

## 

## 

## 

## 

## 

## 

## **1. Enunciado**

Prueba el modelo Naive Bayes (u otro de tu elección) en diferentes entornos y diferentes lenguajes de programación R y Python.

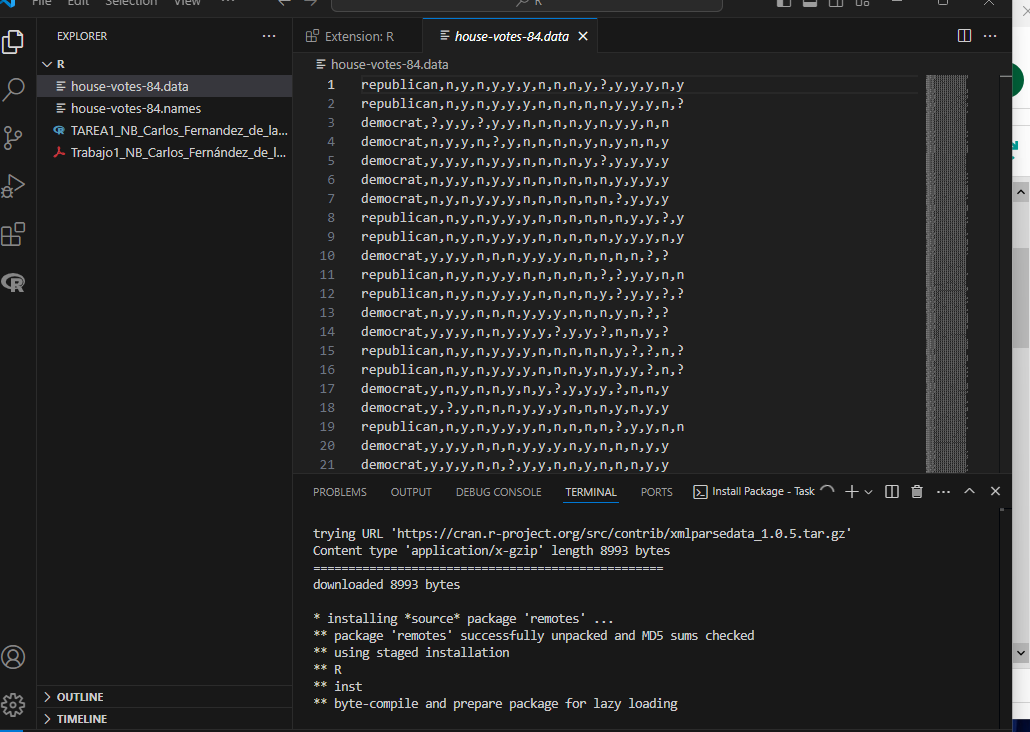
## **2. Ejecución en Visual Studio Code**

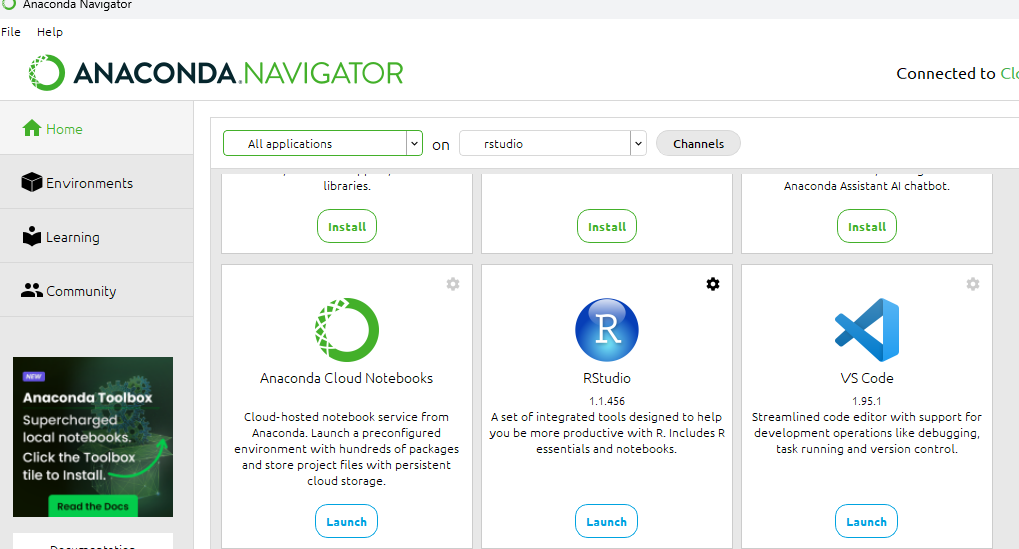
**1.1 Pasos**

Instalo R

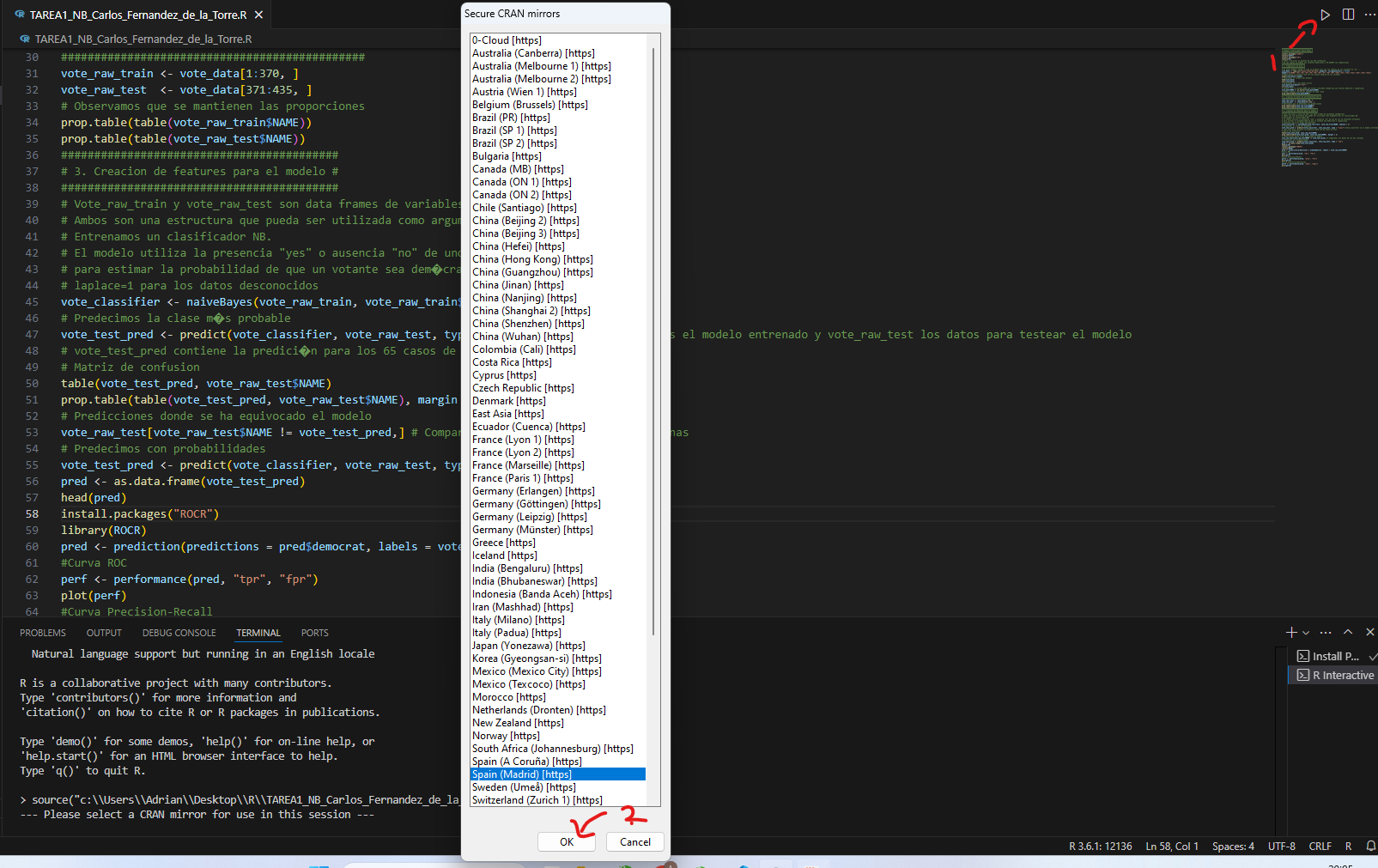


Instalo las dependencias:

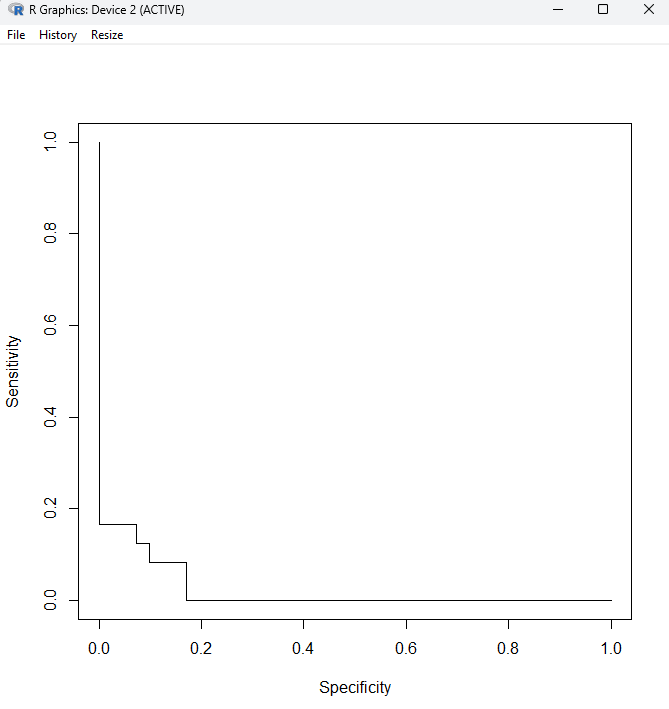


Ejecuto Visual Code****

Ejecuto el programa:



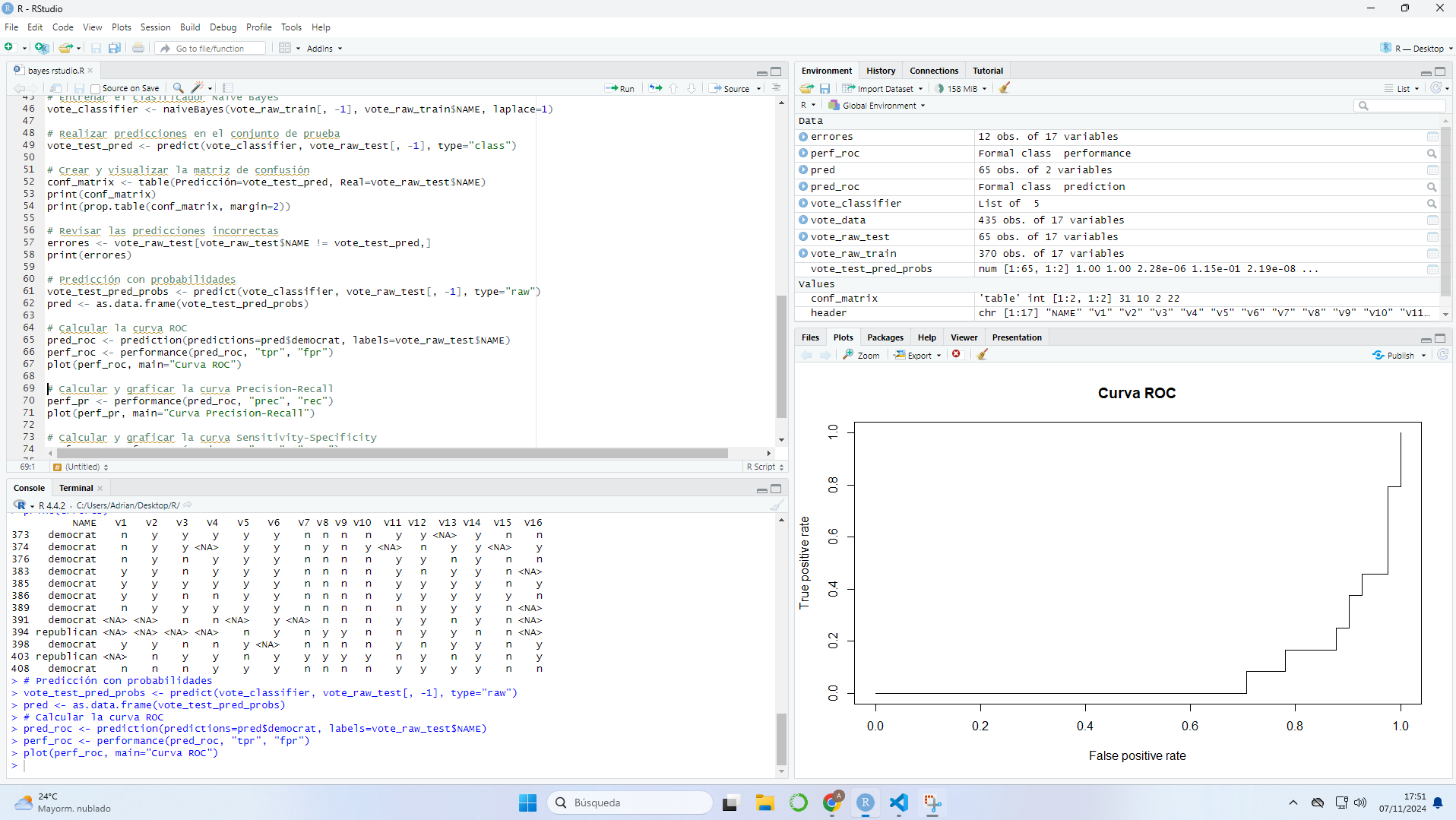
**1.2 Resultado**



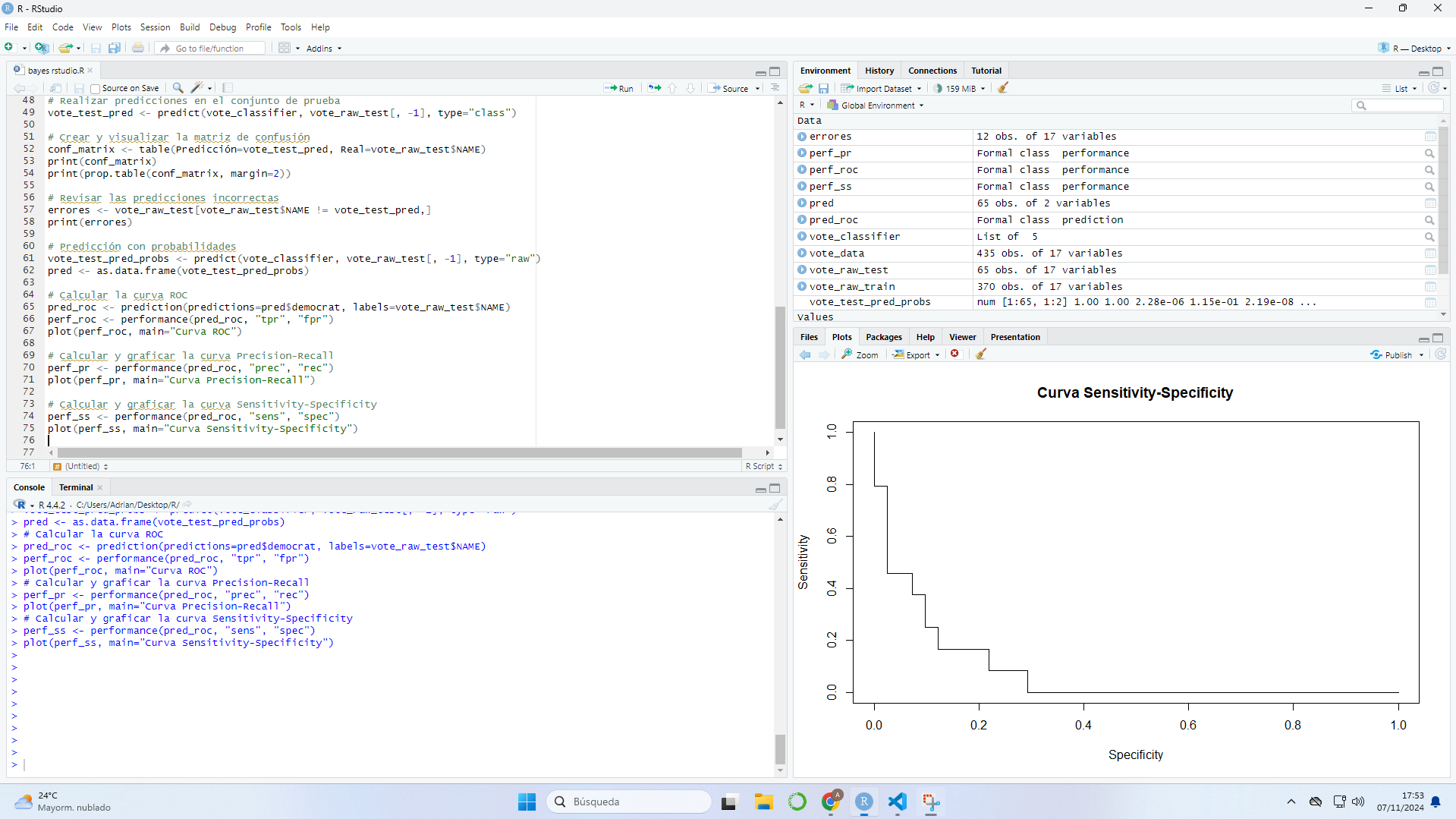
## **3. Ejecución en R Studio**

**1.1 Pasos**

Instalo y abro R Studio y ejecuto el código R

**1.2 Resultado**

## 



## 

## 

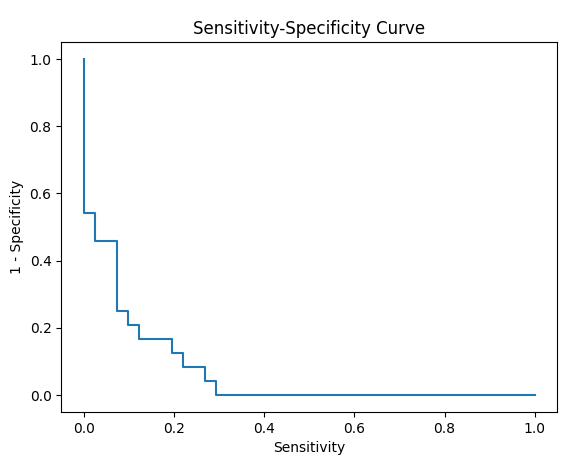
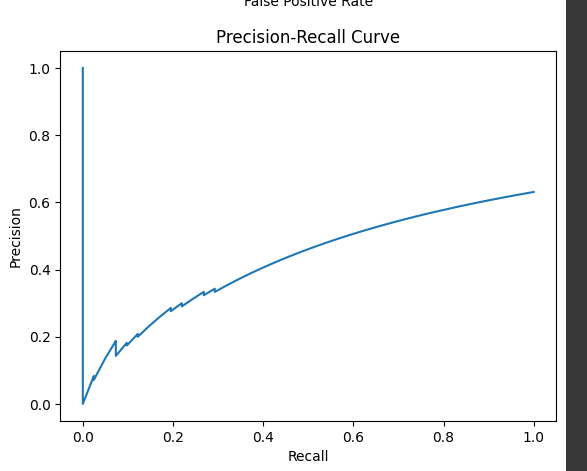
## 

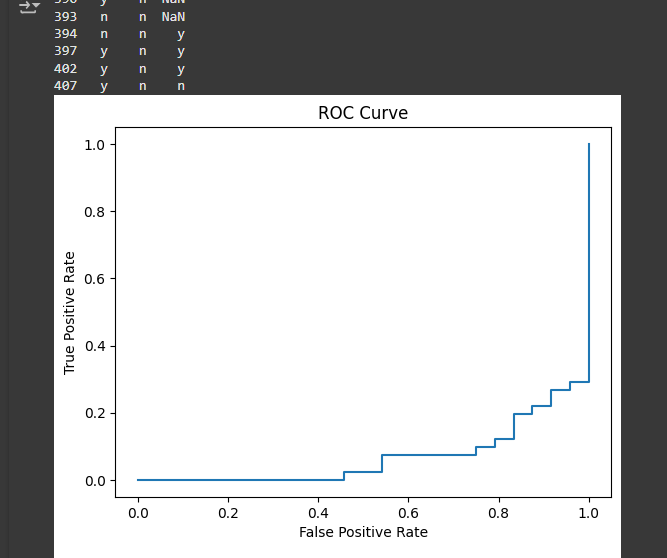
## **4. Ejecución en Google Colab**

**1.1 Pasos**

Ejecuto el código de python en collab y subo el dataset.

**1.2 Resultado**

****

****

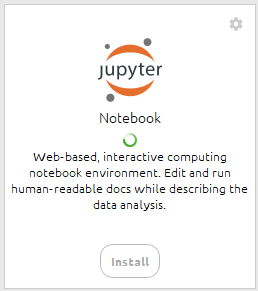
## 

## 

## **5. Ejecución en Anaconda notebooks – Jupyter**

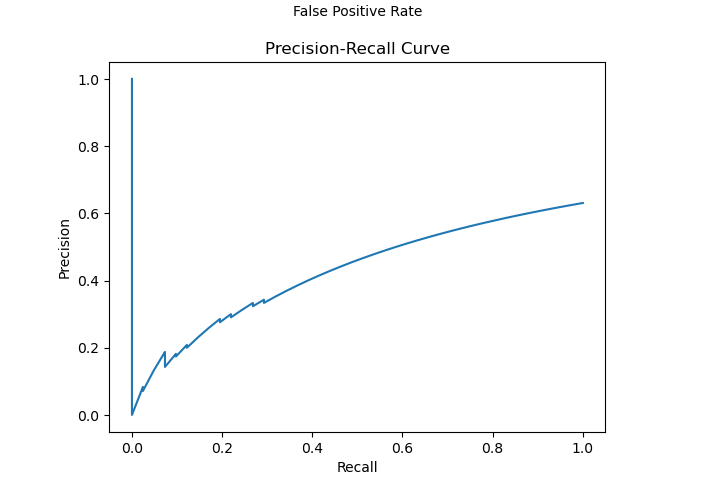
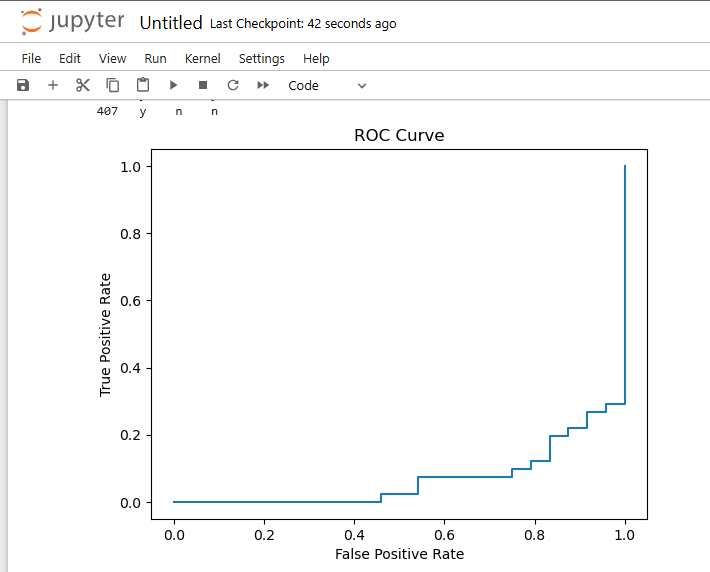
**1.1 Pasos**

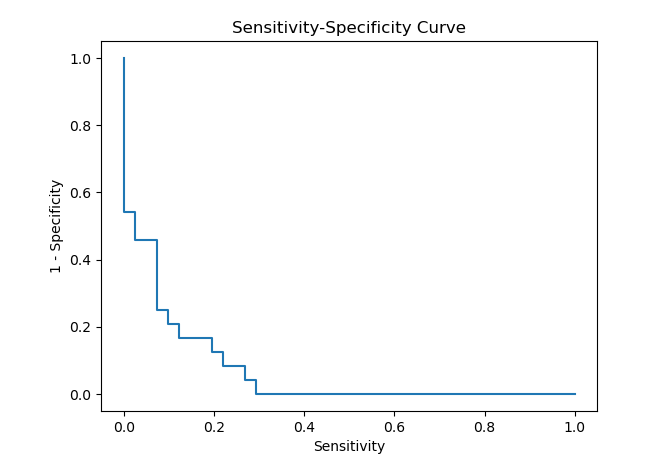
Instalo Jupyter en Anaconda y lo ejecuto



Creo un notebook y pongo el código de python

**1.2 Resultado**

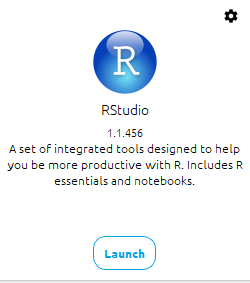




## **6. Ejecución en Anaconda - R Studio**

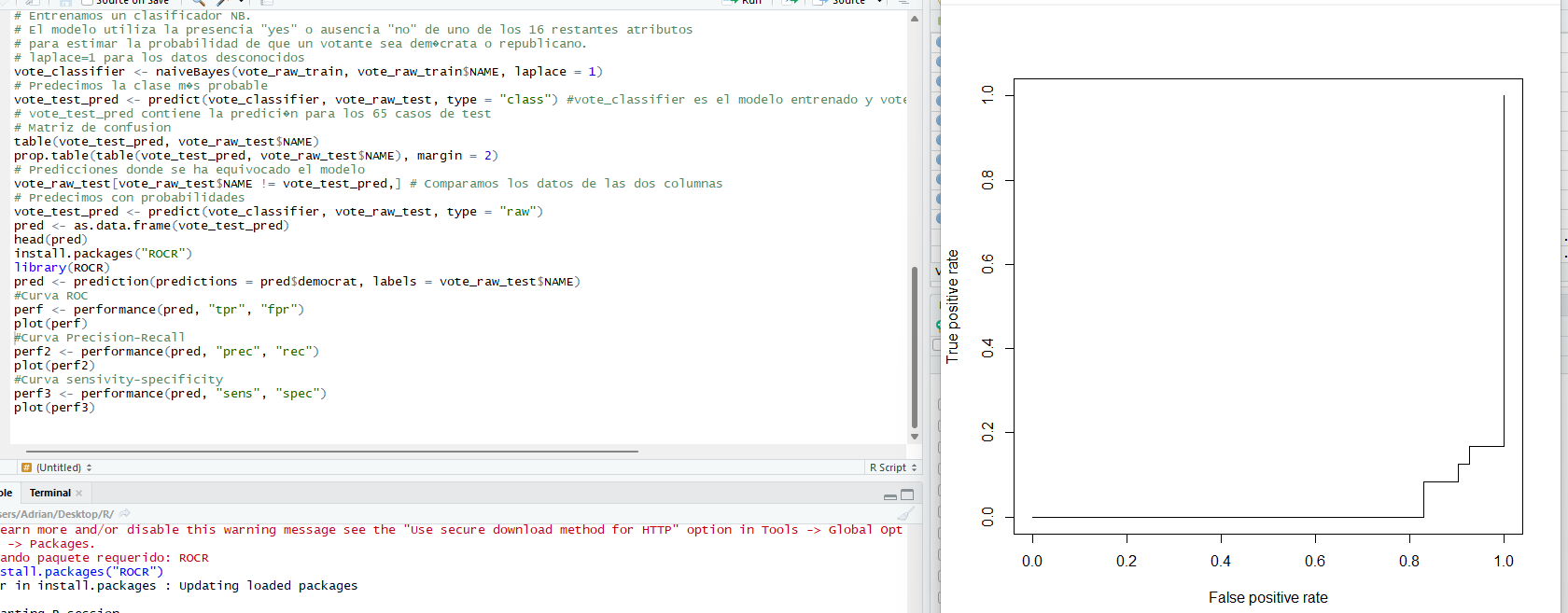
**1.1 Pasos**

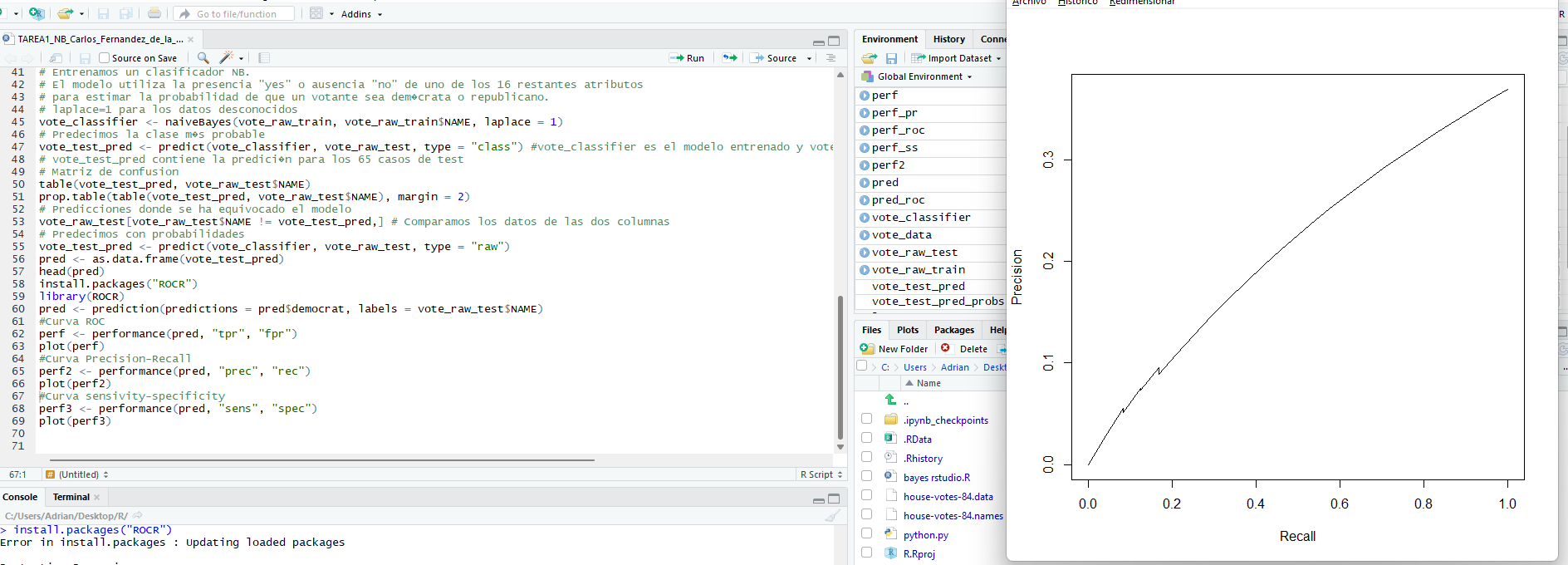
Instalo Rstudio y lo ejecuto

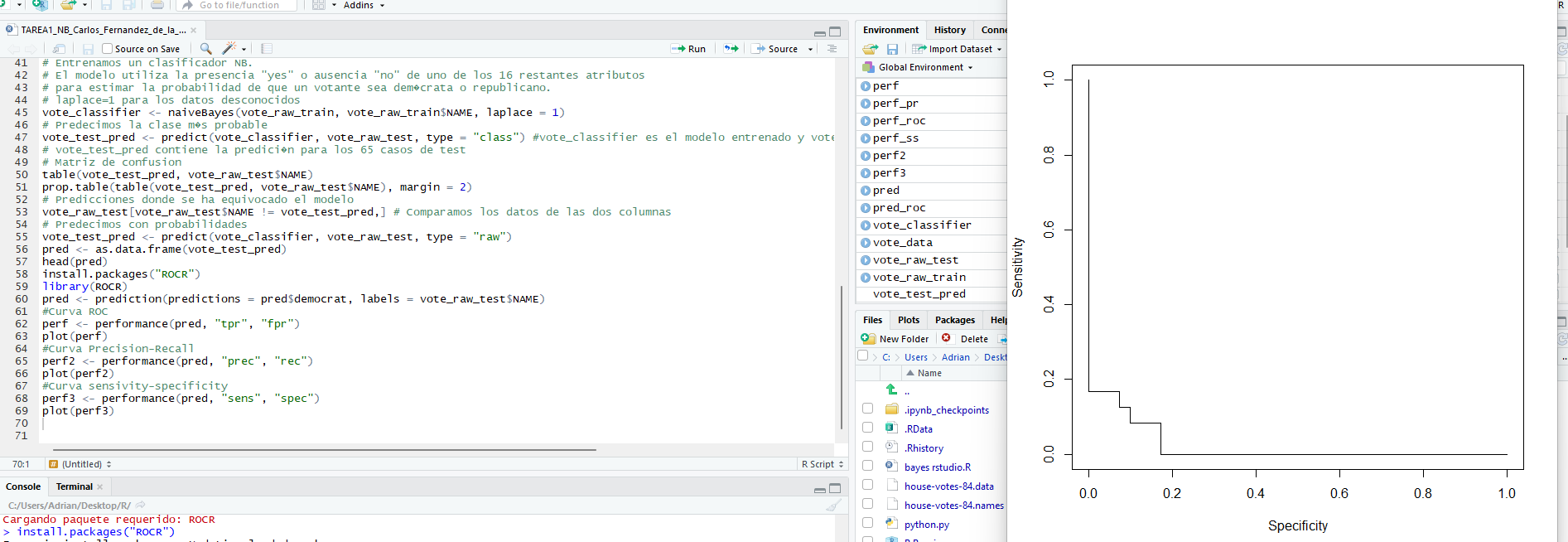


Uso el código en R

**1.2 Resultado**

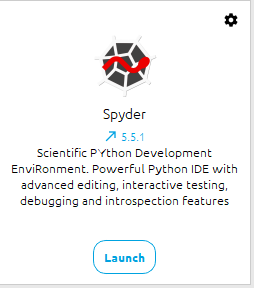
****

****

****

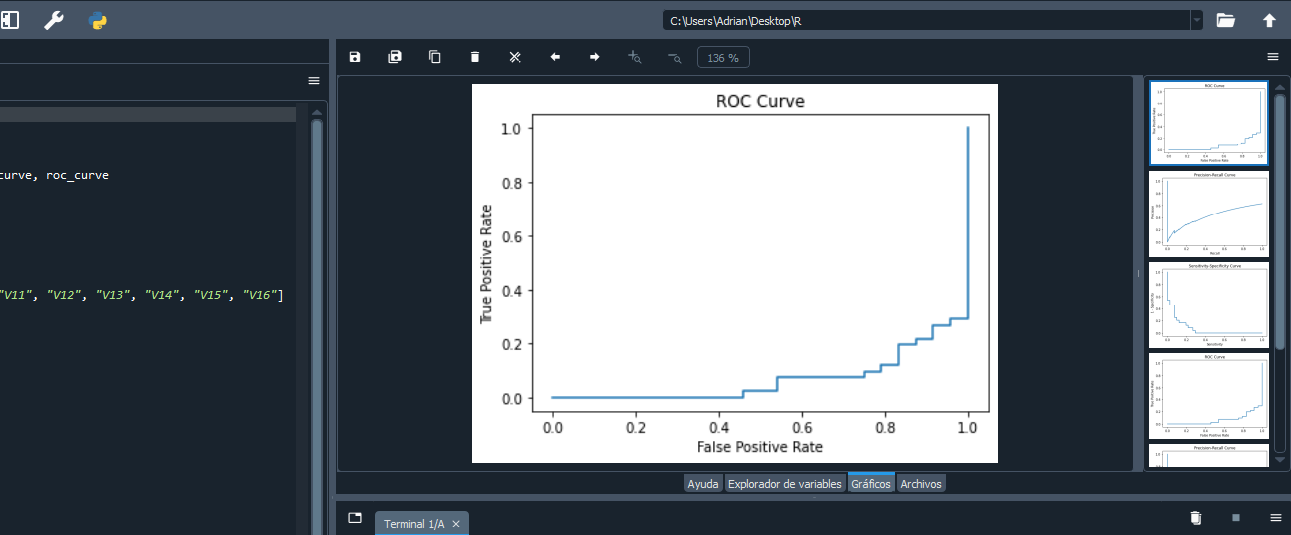
## **7. Ejecución en Anaconda - Spyder**

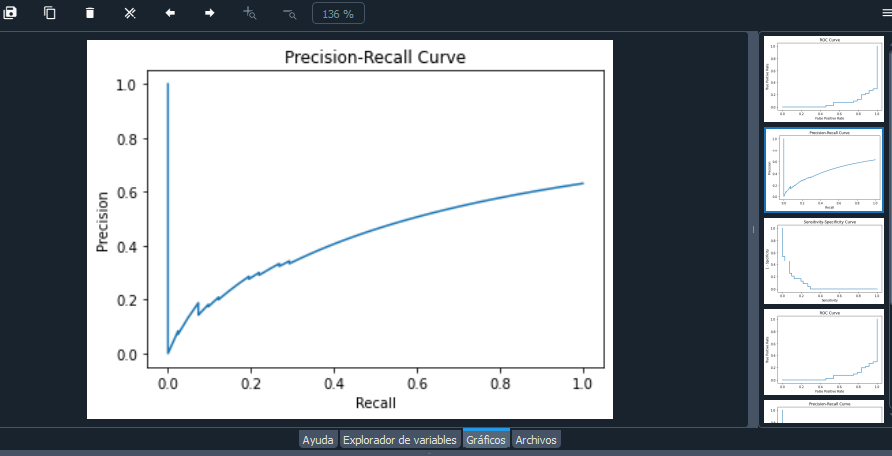
**1.1 Pasos**

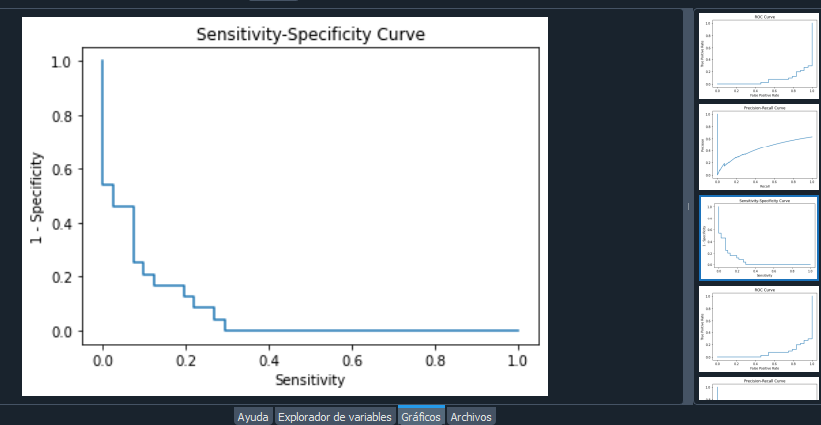
Insatalo spyder   


Ejecuto el código python

**1.2 Resultado**

****

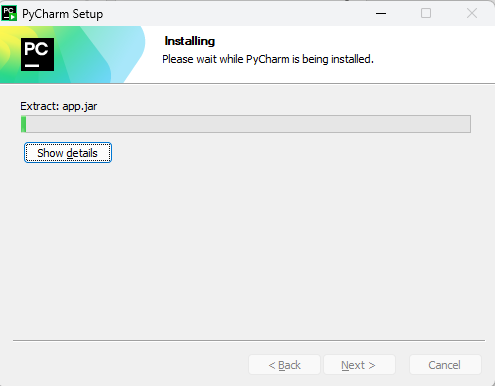
****

****

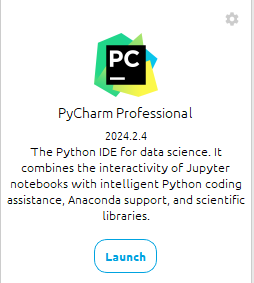
## **8. Ejecución en Pycharm**

**1.1 Pasos**

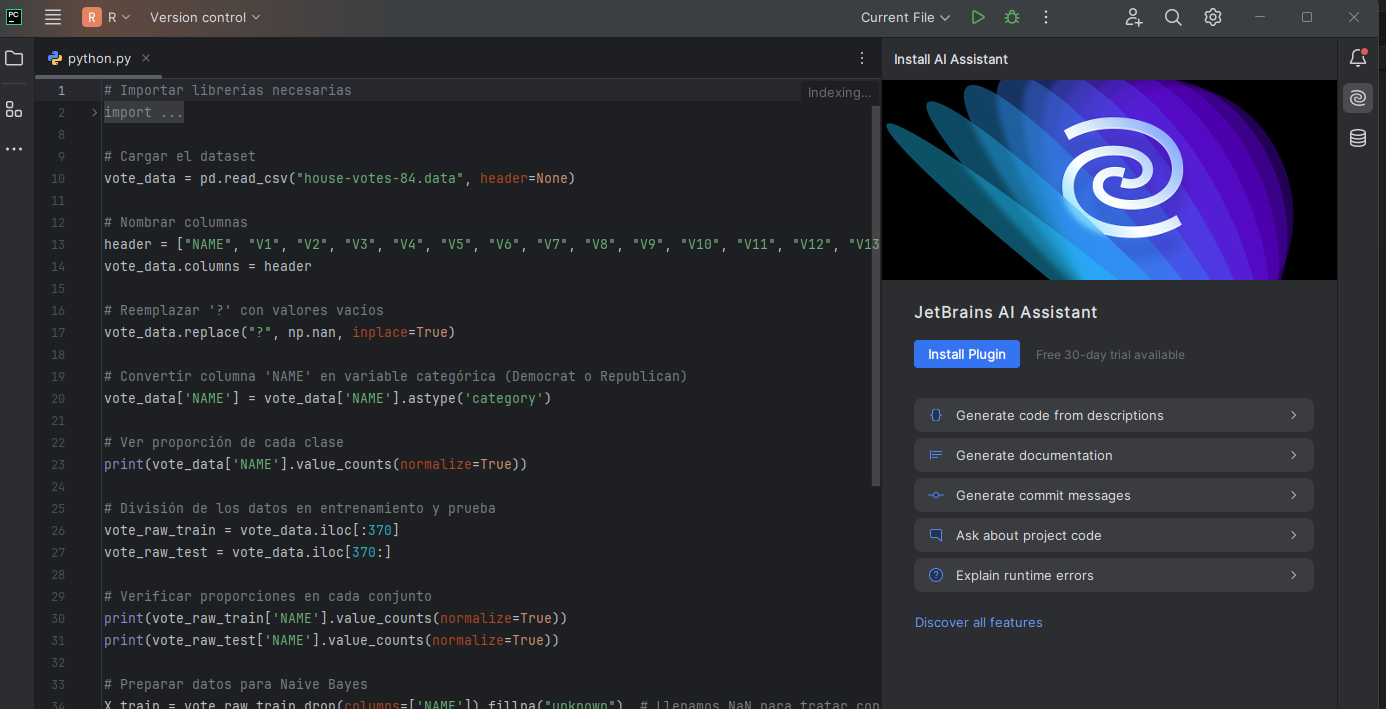
Instalo Pycharm, habiéndolo descargado en google



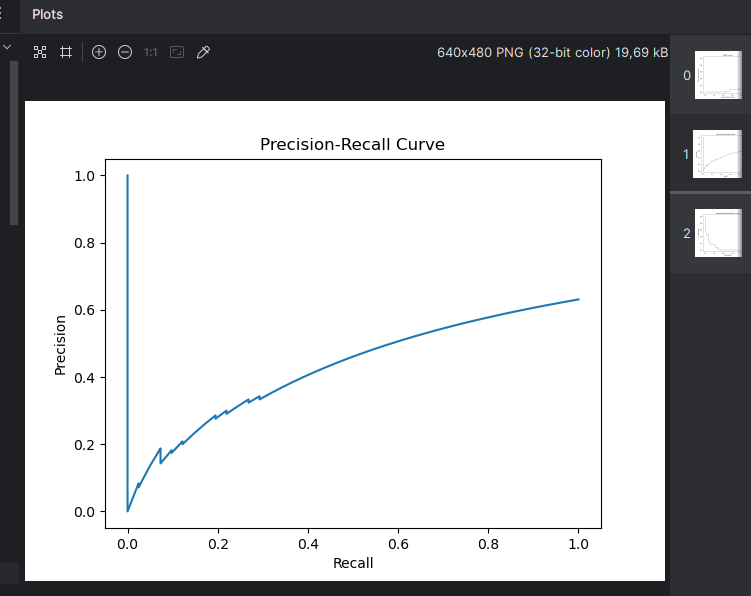
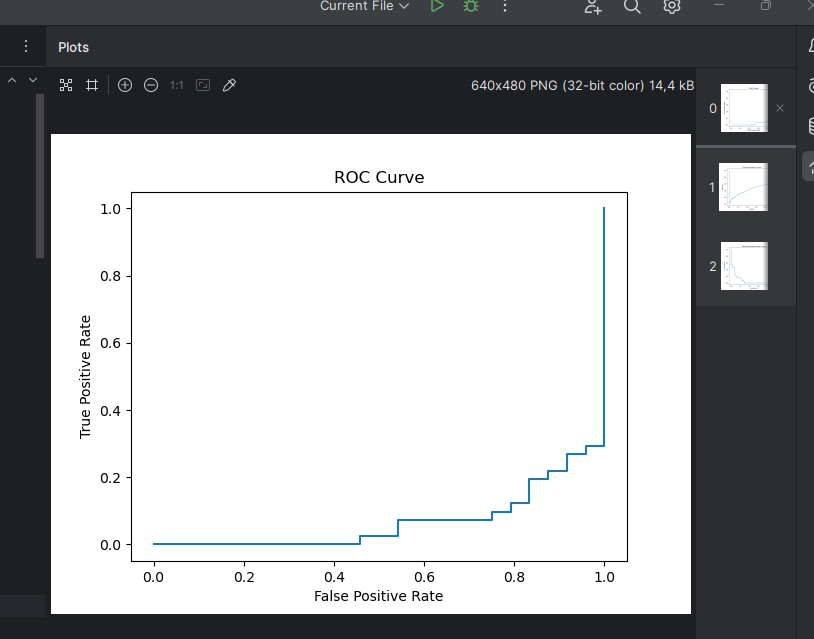
Ejecuto el programa en anaconda

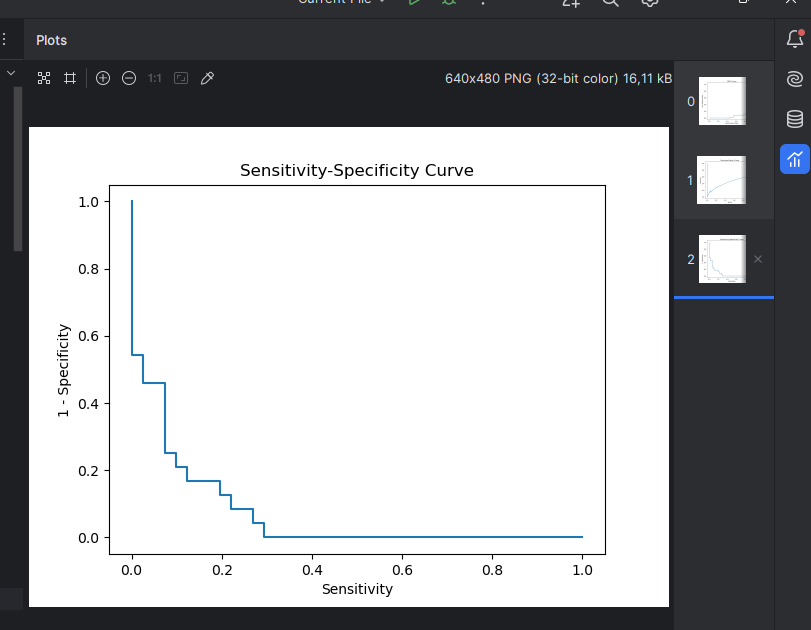


Ejecuto el código python



**1.2 Resultado**

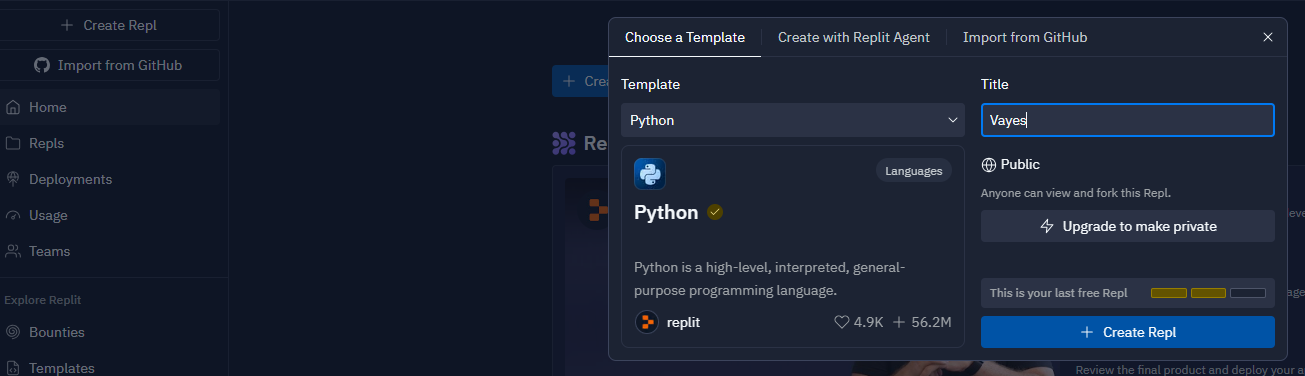
****

****

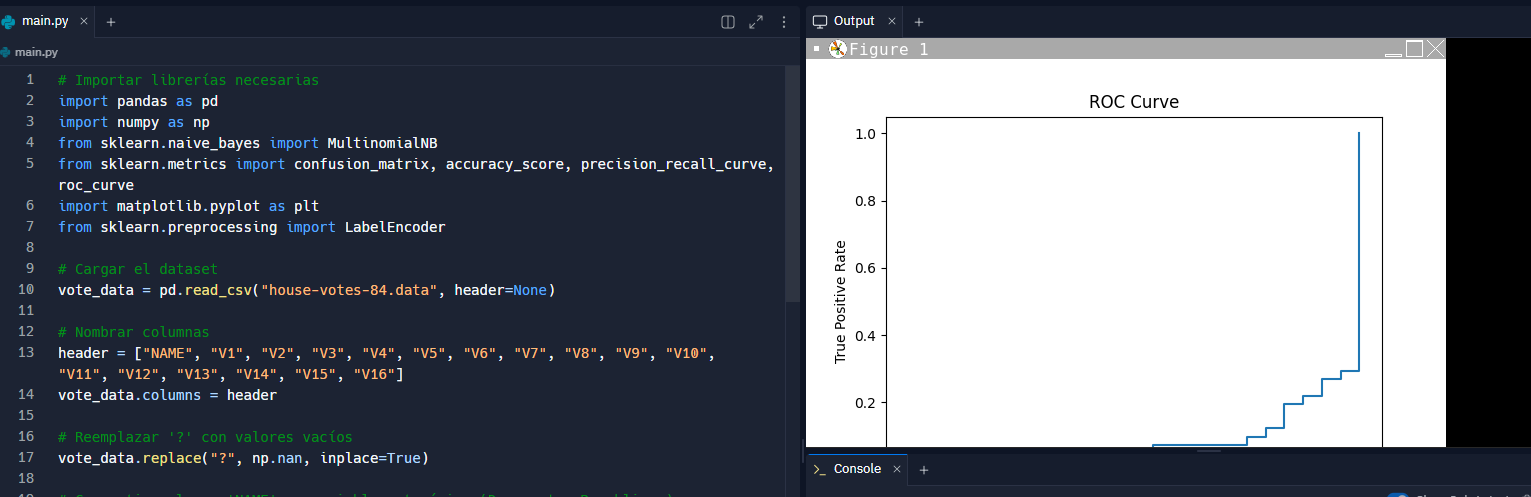
## **9. Ejecución en Replit**

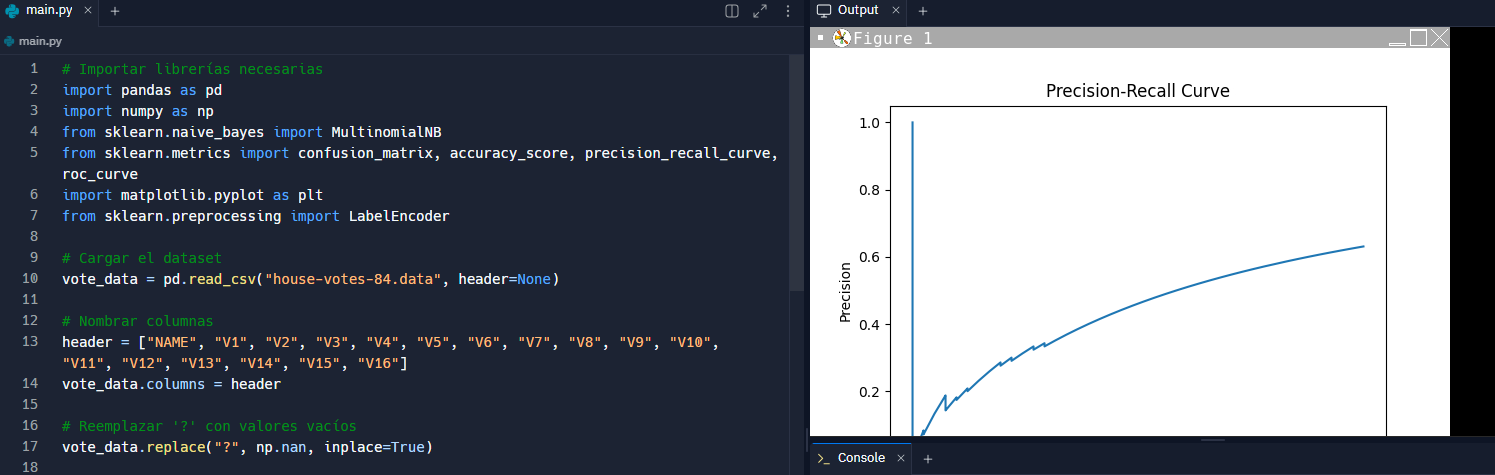
**1.1 Pasos**

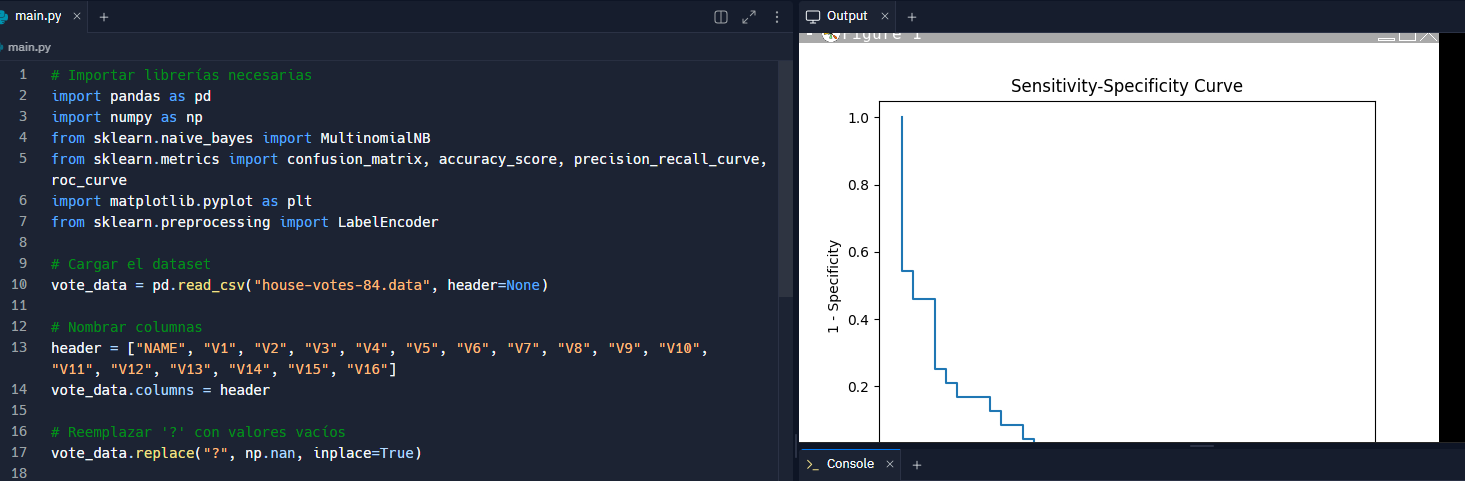
Creo el proyeco en Replit online y pongo el código python



**1.2 Resultado**

****

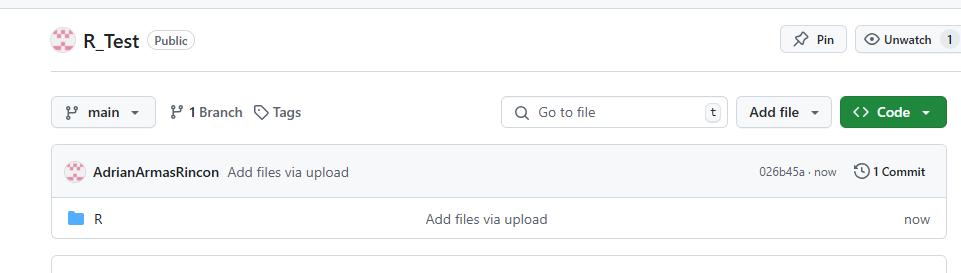
****

****

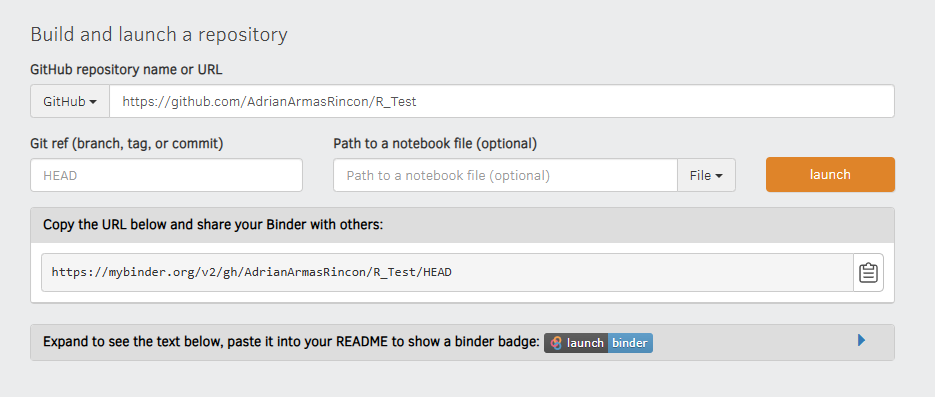
## **10. Ejecución en Binder**

**1.1 Pasos**

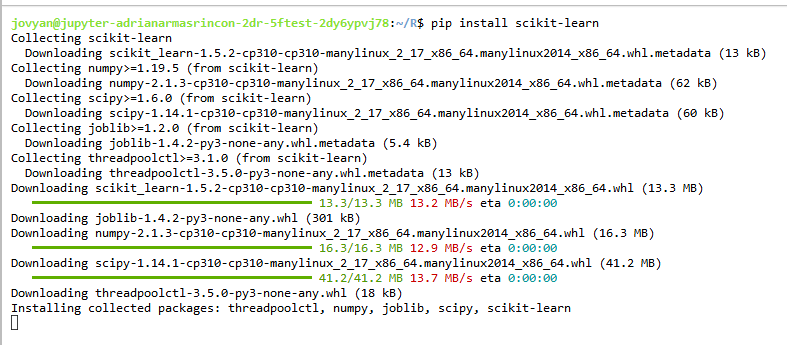
Subo a Github la carpeta de la actividad



Pongo el repositorio en Binder y le doy a ejecutar



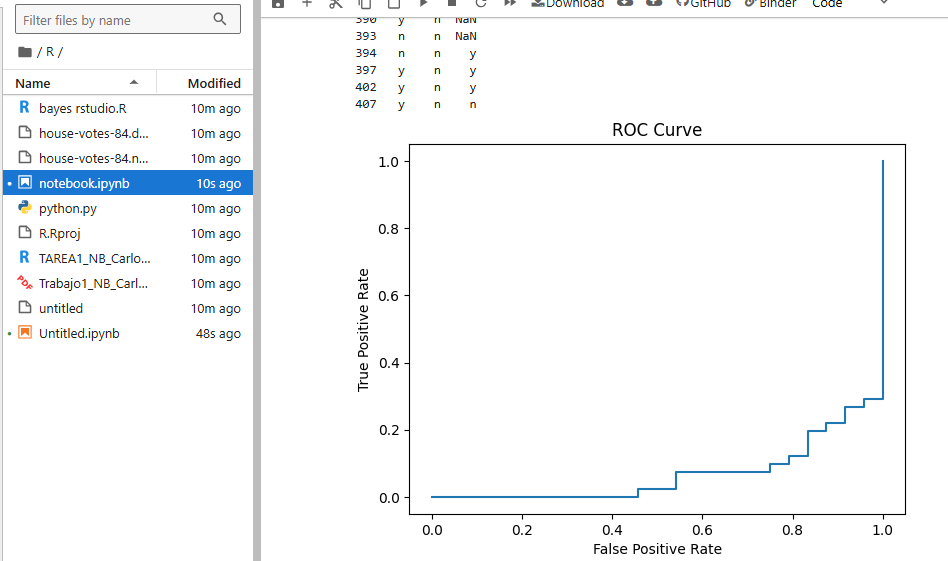
Instalo en binder pandas, mathlib y scikit

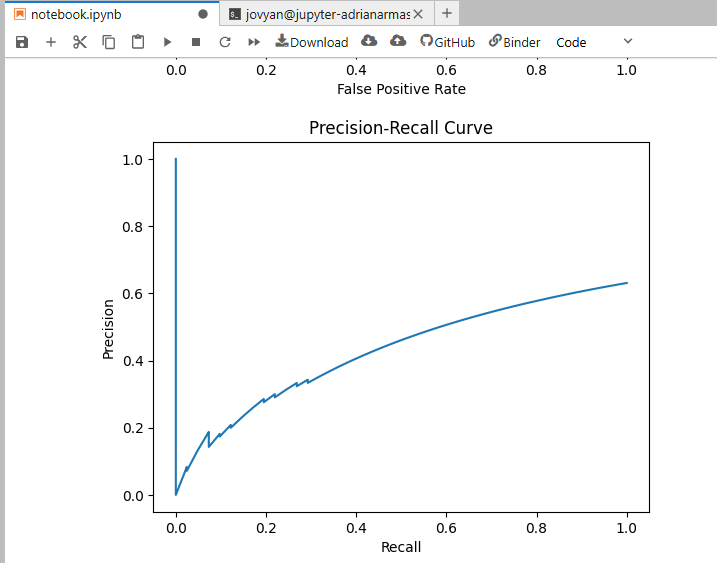


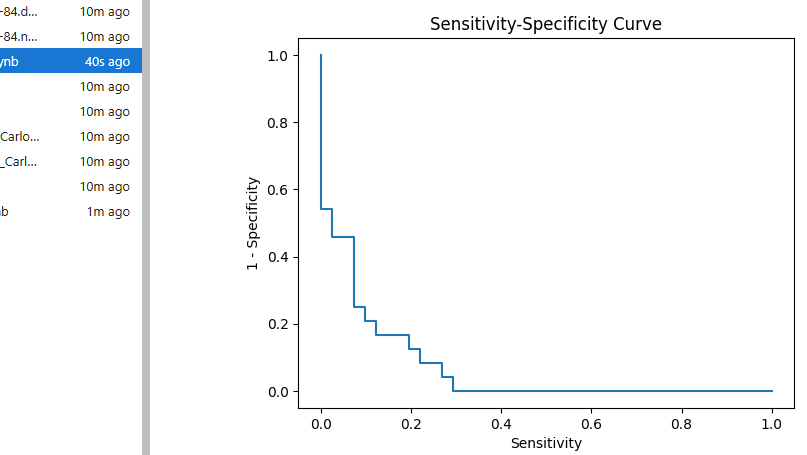




**1.2 Resultado**

****

****

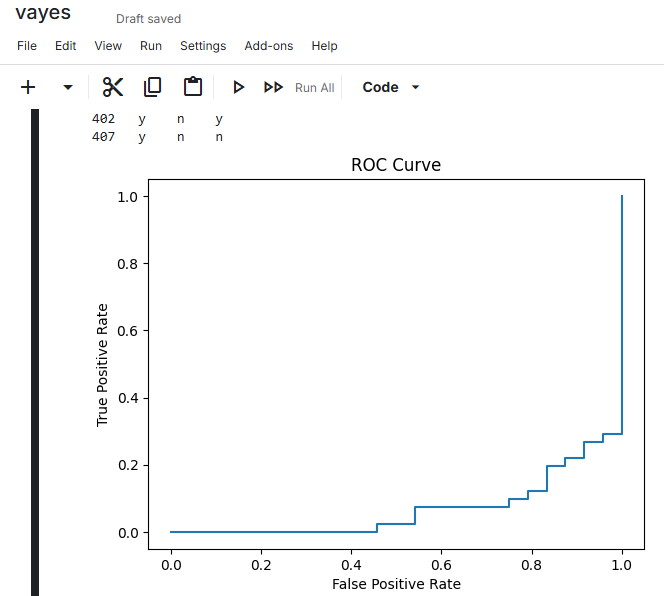
****

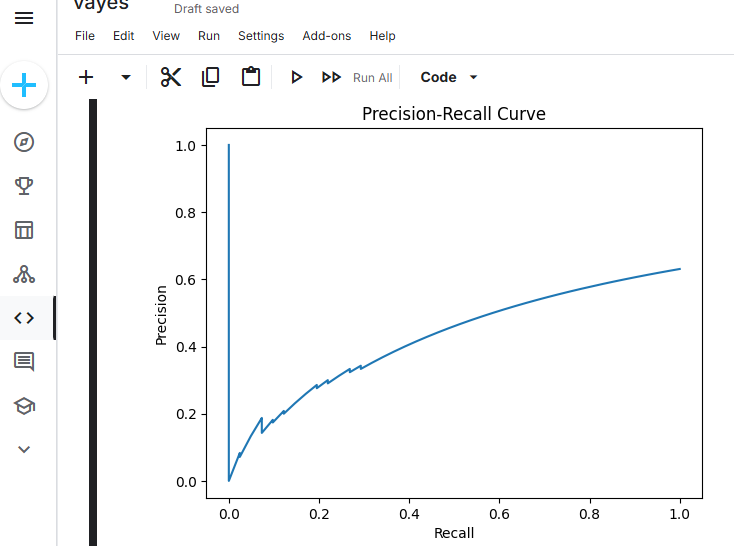
## **11. Ejecución en Kaggle**

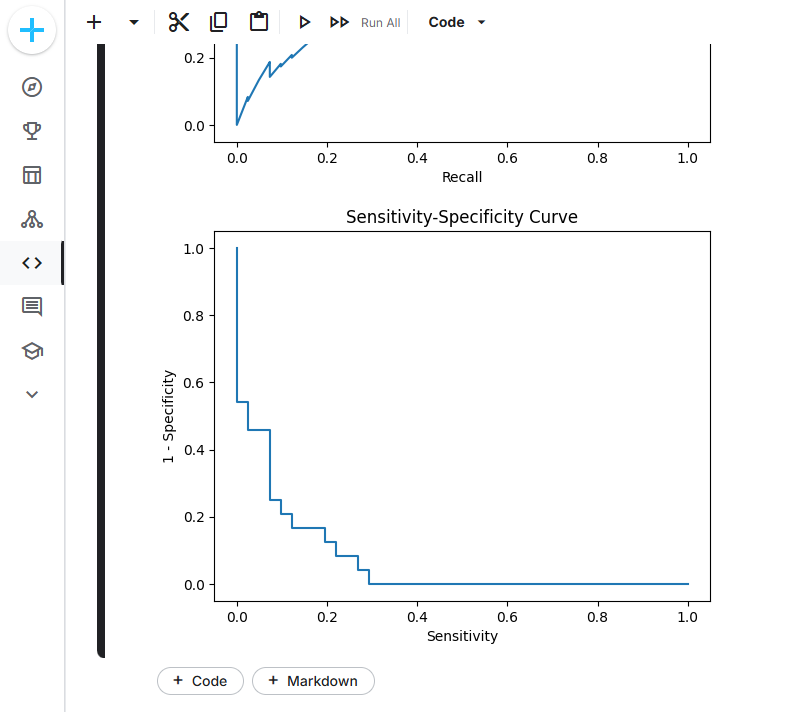
**1.1 Pasos**

Me registro en kagle, creo un proyecto y ejecuto el código modificado de jagle

**1.2 Resultado**

****

****

****

## **12. Código R**

####################################

# Ejemplo Clasificador Naive Bayes #

####################################

# Instala los paquetes solo si no están ya instalados

if(!require("e1071")) install.packages("e1071", dependencies=TRUE)

if(!require("tm")) install.packages("tm", dependencies=TRUE)

if(!require("ROCR")) install.packages("ROCR", dependencies=TRUE)

# Cargar las bibliotecas necesarias

library(e1071)

library(tm)

library(ROCR)

# Cargar el dataset

# Asegúrate de cambiar la ruta del archivo según tu sistema

vote\_data <- read.csv("house-votes-84.data", header=FALSE, stringsAsFactors=FALSE)

# Renombrar las columnas

header <- c("NAME", "V1", "V2", "V3", "V4", "V5", "V6", "V7", "V8", "V9", "V10", "V11", "V12", "V13", "V14", "V15", "V16")

names(vote\_data) <- header

# Reemplazar los "?" con valores vacíos

vote\_data[vote\_data == "?"] <- NA

# Convertir la columna NAME en un factor (variable categórica)

vote\_data$NAME <- as.factor(vote\_data$NAME)

# Visualizar la proporción de cada clase en la columna NAME

print(prop.table(table(vote\_data$NAME)))

##############################################

# 2. Creación de datos de entrenamiento/test #

##############################################

# Dividir el dataset en conjuntos de entrenamiento y prueba

vote\_raw\_train <- vote\_data[1:370, ]

vote\_raw\_test <- vote\_data[371:435, ]

# Revisar las proporciones en cada conjunto

print(prop.table(table(vote\_raw\_train$NAME)))

print(prop.table(table(vote\_raw\_test$NAME)))

##########################################

# 3. Creación de características para el modelo #

##########################################

# Entrenar el clasificador Naive Bayes

vote\_classifier <- naiveBayes(vote\_raw\_train[, -1], vote\_raw\_train$NAME, laplace=1)

# Realizar predicciones en el conjunto de prueba

vote\_test\_pred <- predict(vote\_classifier, vote\_raw\_test[, -1], type="class")

# Crear y visualizar la matriz de confusión

conf\_matrix <- table(Predicción=vote\_test\_pred, Real=vote\_raw\_test$NAME)

print(conf\_matrix)

print(prop.table(conf\_matrix, margin=2))

# Revisar las predicciones incorrectas

errores <- vote\_raw\_test[vote\_raw\_test$NAME != vote\_test\_pred,]

print(errores)

# Predicción con probabilidades

vote\_test\_pred\_probs <- predict(vote\_classifier, vote\_raw\_test[, -1], type="raw")

pred <- as.data.frame(vote\_test\_pred\_probs)

# Calcular la curva ROC

pred\_roc <- prediction(predictions=pred$democrat, labels=vote\_raw\_test$NAME)

perf\_roc <- performance(pred\_roc, "tpr", "fpr")

plot(perf\_roc, main="Curva ROC")

# Calcular y graficar la curva Precision-Recall

perf\_pr <- performance(pred\_roc, "prec", "rec")

plot(perf\_pr, main="Curva Precision-Recall")

# Calcular y graficar la curva Sensitivity-Specificity

perf\_ss <- performance(pred\_roc, "sens", "spec")

plot(perf\_ss, main="Curva Sensitivity-Specificity")

## **13. Código Python**

# Importar librerías necesarias

import pandas as pd

import numpy as np

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import confusion\_matrix, accuracy\_score, precision\_recall\_curve, roc\_curve

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

# Cargar el dataset

vote\_data = pd.read\_csv("house-votes-84.data", header=None)

# Nombrar columnas

header = ["NAME", "V1", "V2", "V3", "V4", "V5", "V6", "V7", "V8", "V9", "V10", "V11", "V12", "V13", "V14", "V15", "V16"]

vote\_data.columns = header

# Reemplazar '?' con valores vacíos

vote\_data.replace("?", np.nan, inplace=True)

# Convertir columna 'NAME' en variable categórica (Democrat o Republican)

vote\_data['NAME'] = vote\_data['NAME'].astype('category')

# Ver proporción de cada clase

print(vote\_data['NAME'].value\_counts(normalize=True))

# División de los datos en entrenamiento y prueba

vote\_raw\_train = vote\_data.iloc[:370]

vote\_raw\_test = vote\_data.iloc[370:]

# Verificar proporciones en cada conjunto

print(vote\_raw\_train['NAME'].value\_counts(normalize=True))

print(vote\_raw\_test['NAME'].value\_counts(normalize=True))

# Preparar datos para Naive Bayes

X\_train = vote\_raw\_train.drop(columns=['NAME']).fillna("unknown") # Llenamos NaN para tratar con datos desconocidos

X\_test = vote\_raw\_test.drop(columns=['NAME']).fillna("unknown")

# Codificar categorías como numéricas

le = LabelEncoder()

X\_train = X\_train.apply(le.fit\_transform)

X\_test = X\_test.apply(le.transform)

# Convertir la columna 'NAME' en valores numéricos

y\_train = le.fit\_transform(vote\_raw\_train['NAME'])

y\_test = le.transform(vote\_raw\_test['NAME'])

# Entrenar modelo Naive Bayes

vote\_classifier = MultinomialNB(alpha=1)

vote\_classifier.fit(X\_train, y\_train)

# Predecir las clases

vote\_test\_pred = vote\_classifier.predict(X\_test)

# Matriz de confusión y precisión por clase

print("Confusion Matrix:\n", confusion\_matrix(y\_test, vote\_test\_pred))

print("Accuracy:", accuracy\_score(y\_test, vote\_test\_pred))

# Comparar predicciones incorrectas

incorrect\_preds = vote\_raw\_test[y\_test != vote\_test\_pred]

print("Incorrect Predictions:\n", incorrect\_preds)

# Predicciones con probabilidades

vote\_test\_pred\_proba = vote\_classifier.predict\_proba(X\_test)

pred = pd.DataFrame(vote\_test\_pred\_proba, columns=le.classes\_)

# Curva ROC

fpr, tpr, \_ = roc\_curve(y\_test, vote\_test\_pred\_proba[:, 1], pos\_label=le.classes\_.tolist().index('democrat'))

plt.plot(fpr, tpr, label='ROC Curve')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve')

plt.show()

# Curva Precision-Recall

precision, recall, \_ = precision\_recall\_curve(y\_test, vote\_test\_pred\_proba[:, 1], pos\_label=le.classes\_.tolist().index('democrat'))

plt.plot(recall, precision, label='Precision-Recall Curve')

plt.xlabel('Recall')

plt.ylabel('Precision')

plt.title('Precision-Recall Curve')

plt.show()

# Curva Sensitivity-Specificity

plt.plot(tpr, 1 - fpr, label='Sensitivity-Specificity Curve')

plt.xlabel('Sensitivity')

plt.ylabel('1 - Specificity')

plt.title('Sensitivity-Specificity Curve')

plt.show()

## **14. Código Kaggle**

# Importar librerías necesarias

import pandas as pd

import numpy as np

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import confusion\_matrix, accuracy\_score, precision\_recall\_curve, roc\_curve

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

# Cargar el dataset

vote\_data = pd.read\_csv("/kaggle/input/house-votes-84-data/house-votes-84.data", header=None)

# Nombrar columnas

header = ["NAME", "V1", "V2", "V3", "V4", "V5", "V6", "V7", "V8", "V9", "V10", "V11", "V12", "V13", "V14", "V15", "V16"]

vote\_data.columns = header

# Reemplazar '?' con valores vacíos

vote\_data.replace("?", np.nan, inplace=True)

# Convertir columna 'NAME' en variable categórica (Democrat o Republican)

vote\_data['NAME'] = vote\_data['NAME'].astype('category')

# Ver proporción de cada clase

print(vote\_data['NAME'].value\_counts(normalize=True))

# División de los datos en entrenamiento y prueba

vote\_raw\_train = vote\_data.iloc[:370]

vote\_raw\_test = vote\_data.iloc[370:]

# Verificar proporciones en cada conjunto

print(vote\_raw\_train['NAME'].value\_counts(normalize=True))

print(vote\_raw\_test['NAME'].value\_counts(normalize=True))

# Preparar datos para Naive Bayes

X\_train = vote\_raw\_train.drop(columns=['NAME']).fillna("unknown") # Llenamos NaN para tratar con datos desconocidos

X\_test = vote\_raw\_test.drop(columns=['NAME']).fillna("unknown")

# Codificar categorías como numéricas

le = LabelEncoder()

X\_train = X\_train.apply(le.fit\_transform)

X\_test = X\_test.apply(le.transform)

# Convertir la columna 'NAME' en valores numéricos

y\_train = le.fit\_transform(vote\_raw\_train['NAME'])

y\_test = le.transform(vote\_raw\_test['NAME'])

# Entrenar modelo Naive Bayes

vote\_classifier = MultinomialNB(alpha=1)

vote\_classifier.fit(X\_train, y\_train)

# Predecir las clases

vote\_test\_pred = vote\_classifier.predict(X\_test)

# Matriz de confusión y precisión por clase

print("Confusion Matrix:\n", confusion\_matrix(y\_test, vote\_test\_pred))

print("Accuracy:", accuracy\_score(y\_test, vote\_test\_pred))

# Comparar predicciones incorrectas

incorrect\_preds = vote\_raw\_test[y\_test != vote\_test\_pred]

print("Incorrect Predictions:\n", incorrect\_preds)

# Predicciones con probabilidades

vote\_test\_pred\_proba = vote\_classifier.predict\_proba(X\_test)

pred = pd.DataFrame(vote\_test\_pred\_proba, columns=le.classes\_)

# Curva ROC

fpr, tpr, \_ = roc\_curve(y\_test, vote\_test\_pred\_proba[:, 1], pos\_label=le.classes\_.tolist().index('democrat'))

plt.plot(fpr, tpr, label='ROC Curve')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve')

plt.show()

# Curva Precision-Recall

precision, recall, \_ = precision\_recall\_curve(y\_test, vote\_test\_pred\_proba[:, 1], pos\_label=le.classes\_.tolist().index('democrat'))

plt.plot(recall, precision, label='Precision-Recall Curve')

plt.xlabel('Recall')

plt.ylabel('Precision')

plt.title('Precision-Recall Curve')

plt.show()

# Curva Sensitivity-Specificity

plt.plot(tpr, 1 - fpr, label='Sensitivity-Specificity Curve')

plt.xlabel('Sensitivity')

plt.ylabel('1 - Specificity')

plt.title('Sensitivity-Specificity Curve')

plt.show()

## **14. Github**

[](https://github.com/AdrianArmasRincon/R_Test)