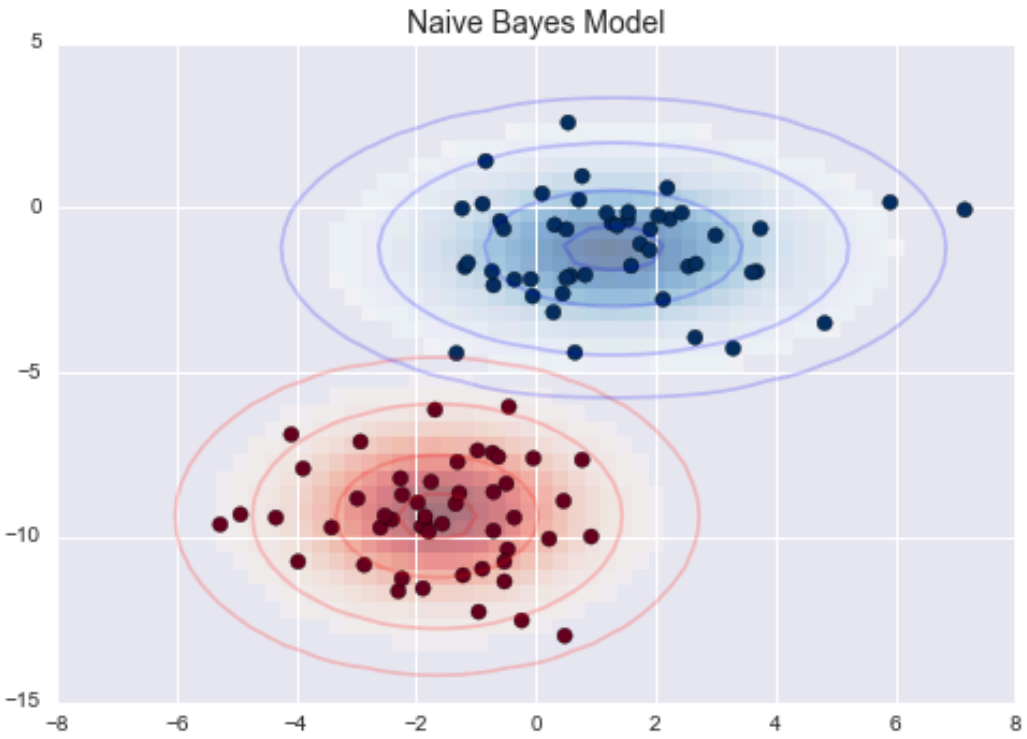
Naive Bayes en diferentes IDE



Adrián Yared Armas de la Nuez

Contenido

1. Enunciado

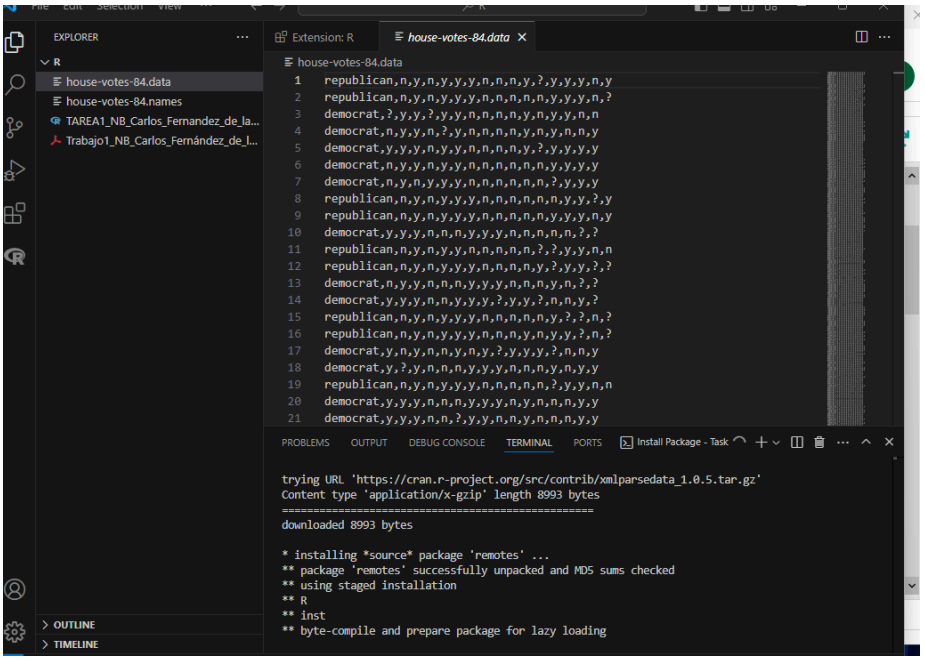
Prueba el modelo Naive Bayes (u otro de tu elección) en diferentes entornos y diferentes lenguajes de programación R y Python.

2. Ejecución en Visual Studio Code

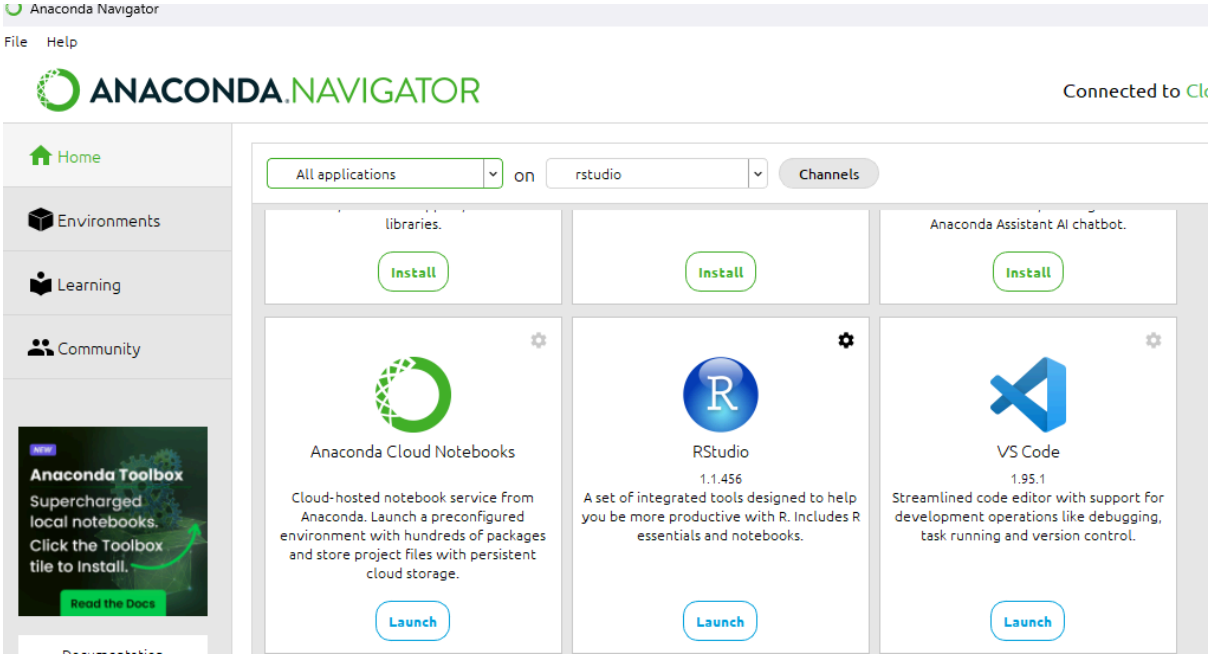
1.1 Pasos

Instalo R

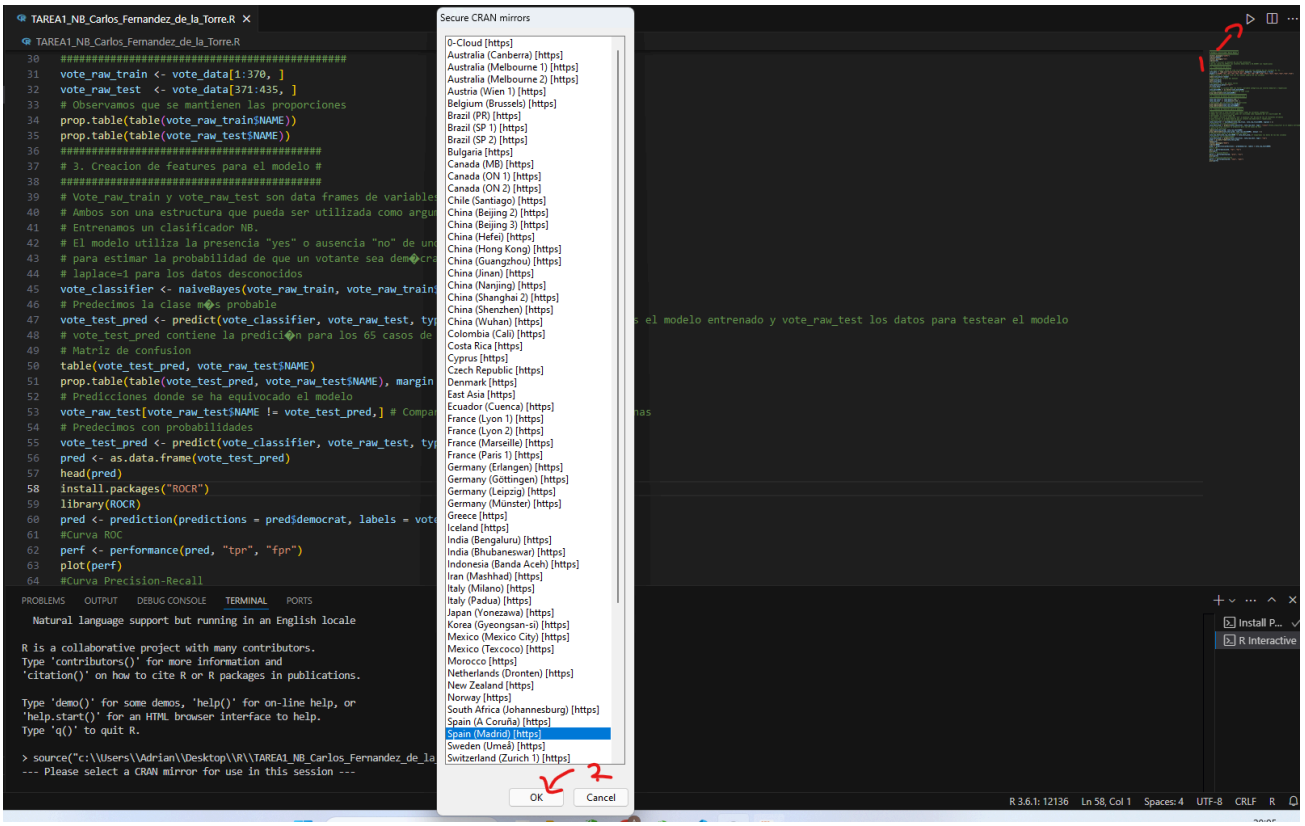
Instalo las dependencias:



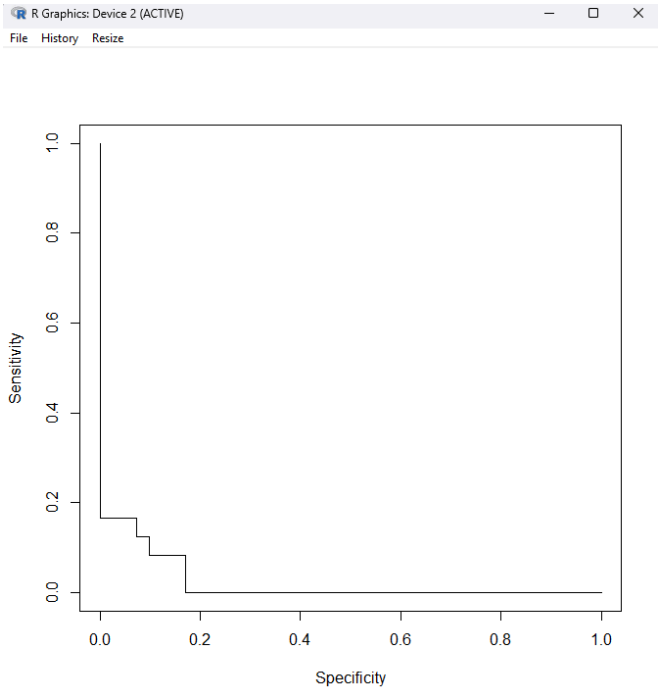
Ejecuto Visual Code



Ejecuto el programa:



1.2 Resultado

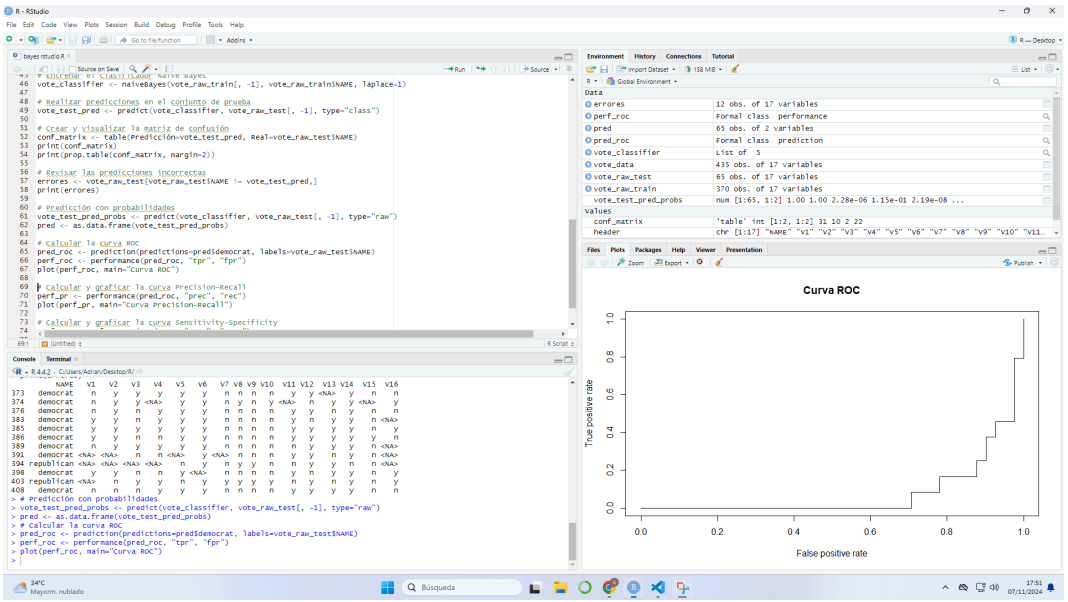


3. Ejecución en R Studio

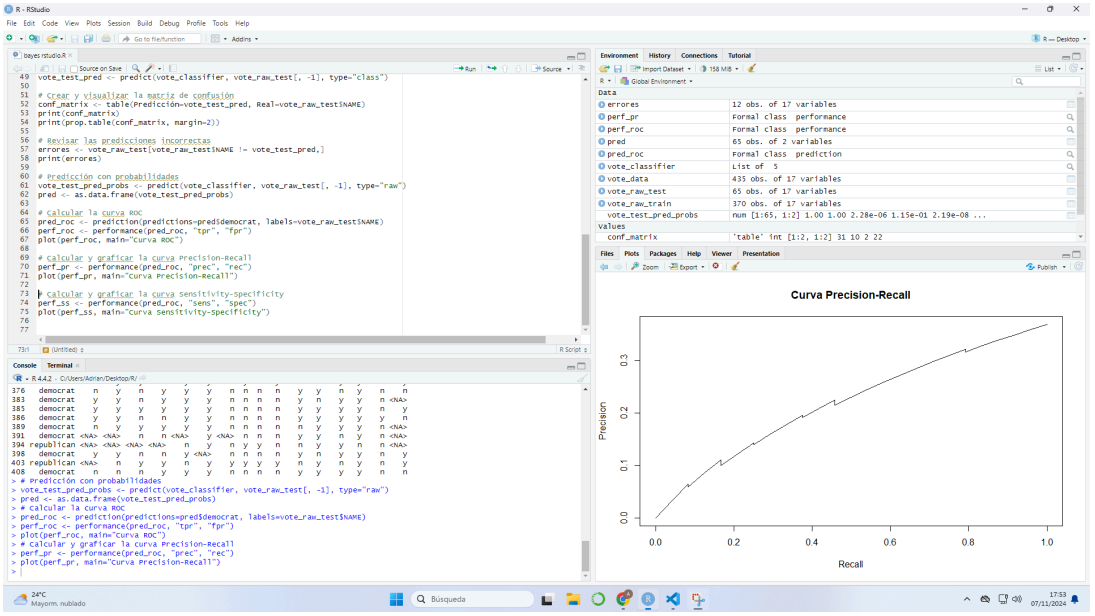
1.1 Pasos

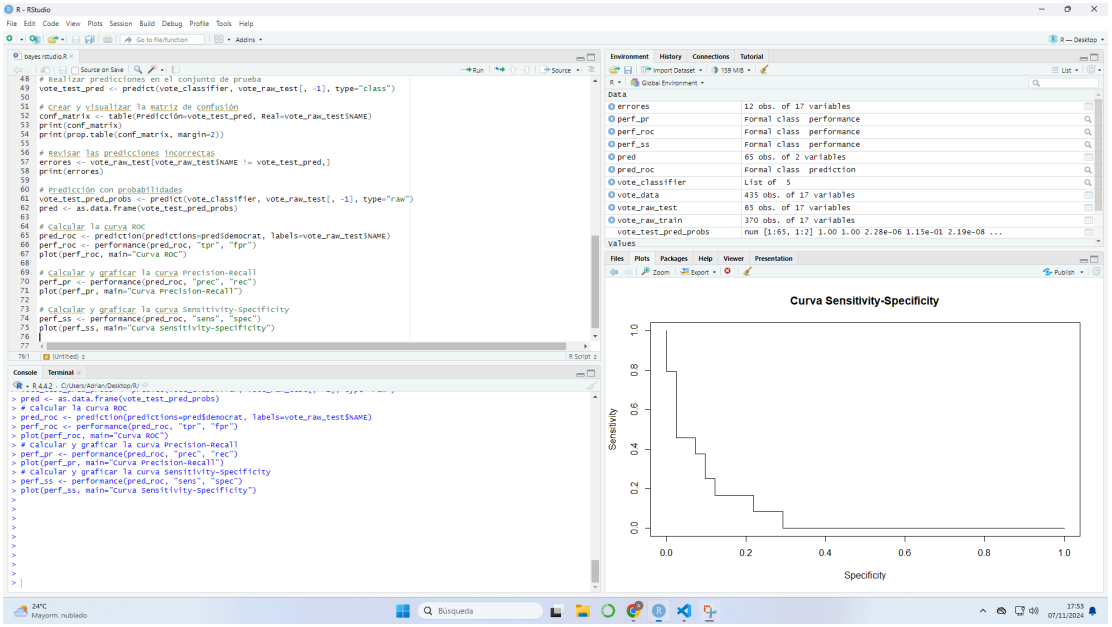
Instalo y abro R Studio y ejecuto el código R

1.2 Resultado



Naive Bayes en diferentes IDE



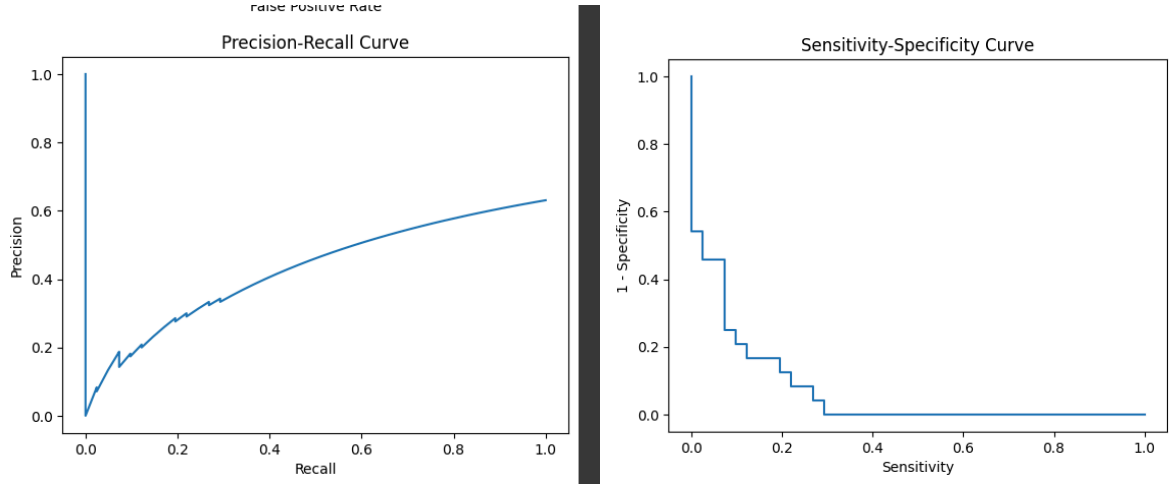


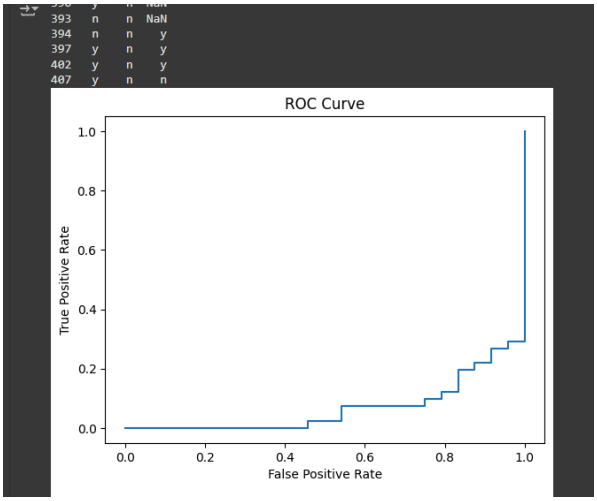
4. Ejecución en Google Colab

1.1 Pasos

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

1.2 Resultado

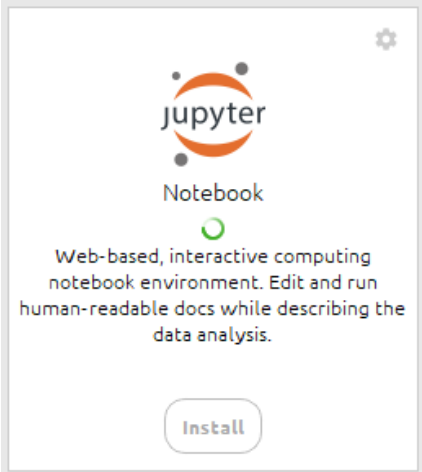




5. Ejecución en Anaconda notebooks – Jupyter

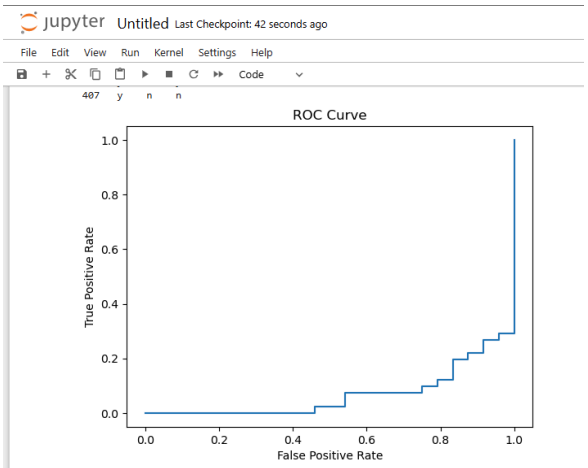
1.1 Pasos

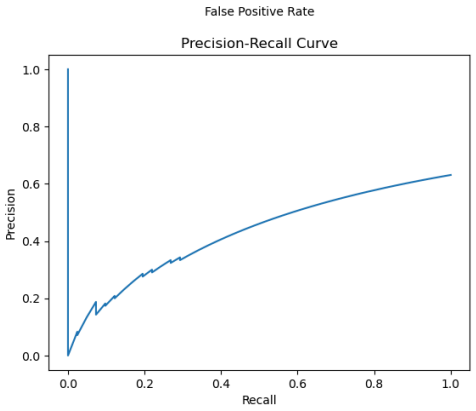
Instalo Jupyter en Anaconda y lo ejecuto

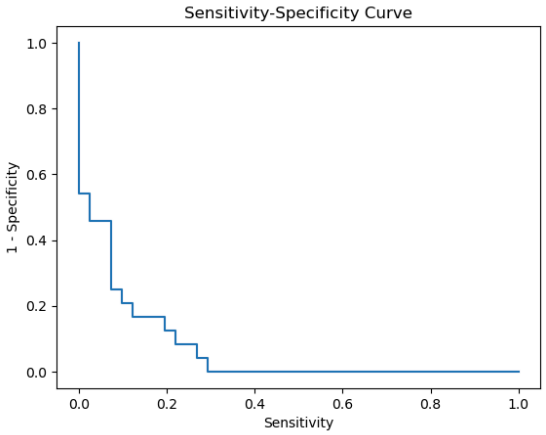


Creo un notebook y pongo el código de python

1.2 Resultado



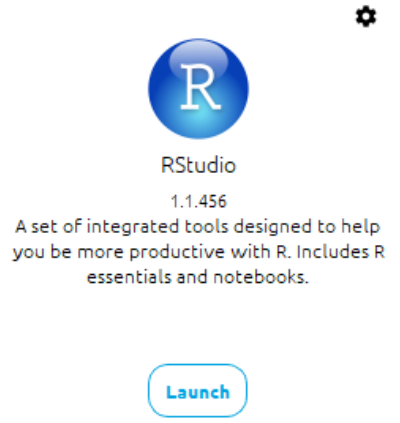




6. Ejecución en Anaconda - R Studio

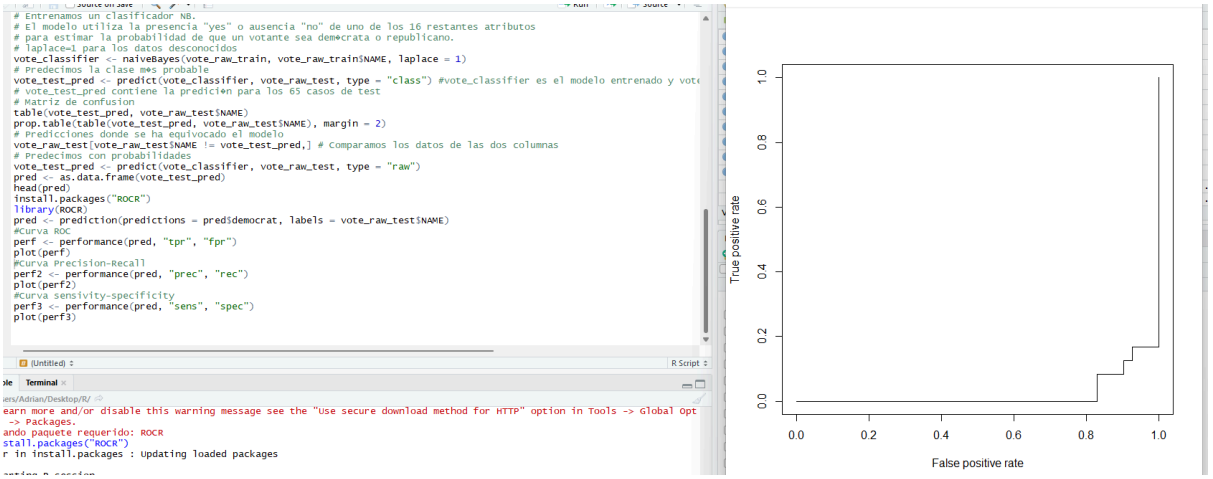
1.1 Pasos

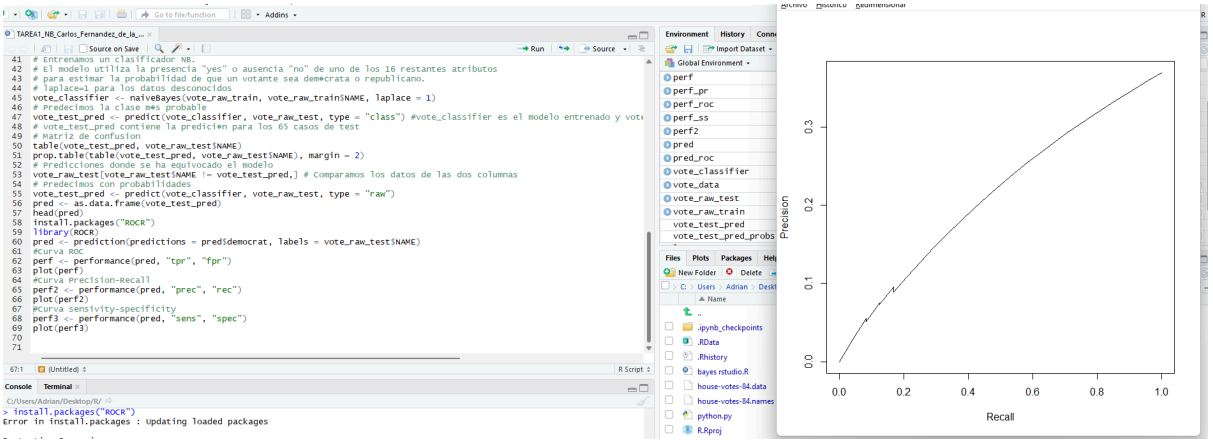
Instalo Rstudio y lo ejecuto

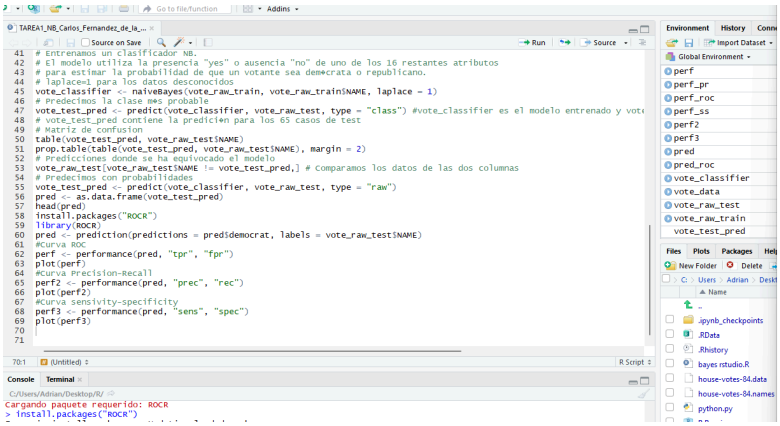


Uso el código en R

1.2 Resultado



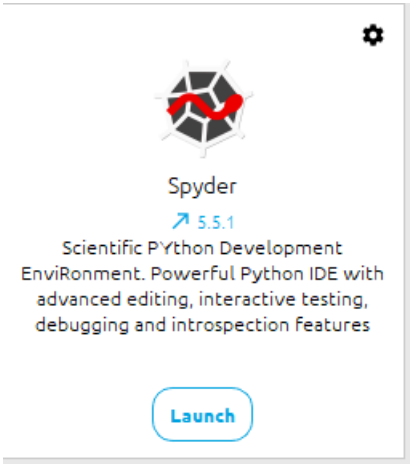




7. Ejecución en Anaconda - Spyder

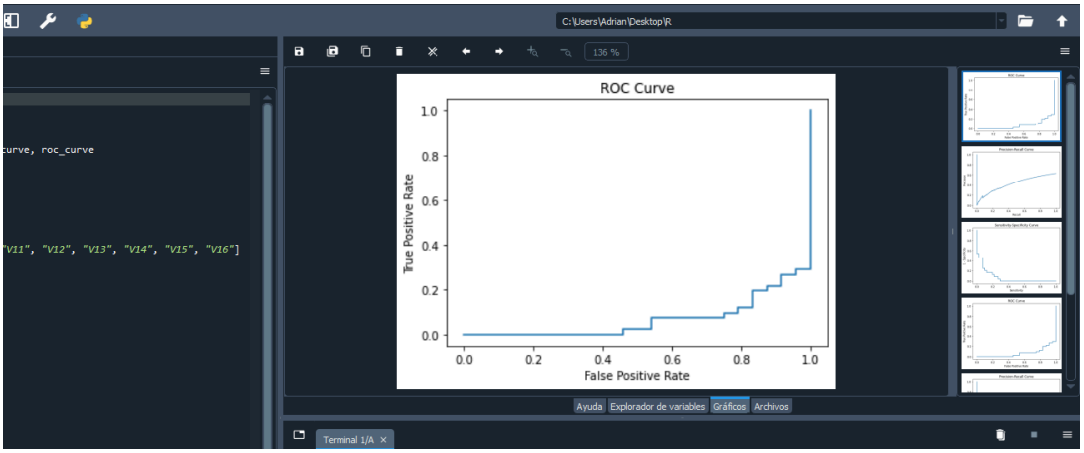
1.1 Pasos

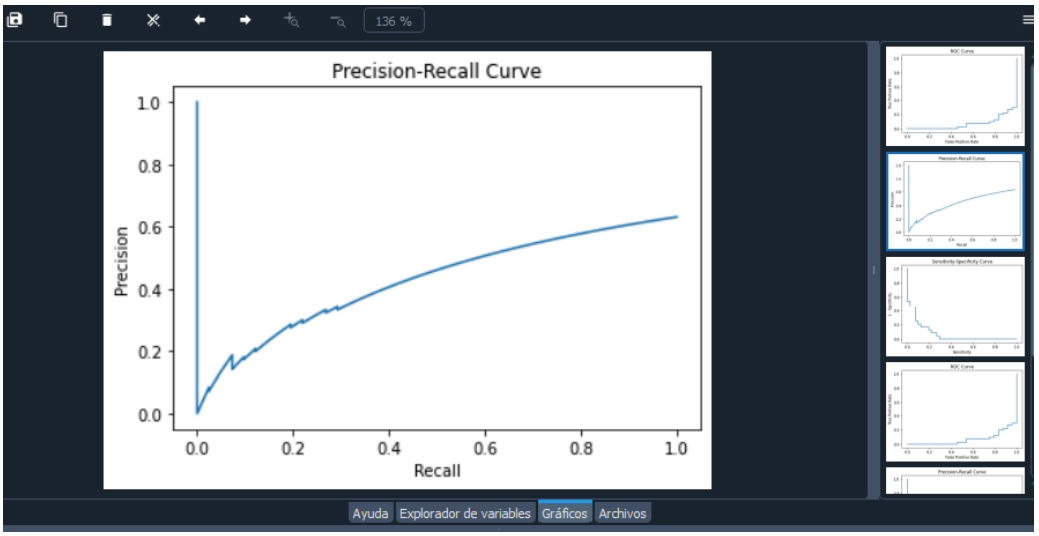
Insatalo spyder

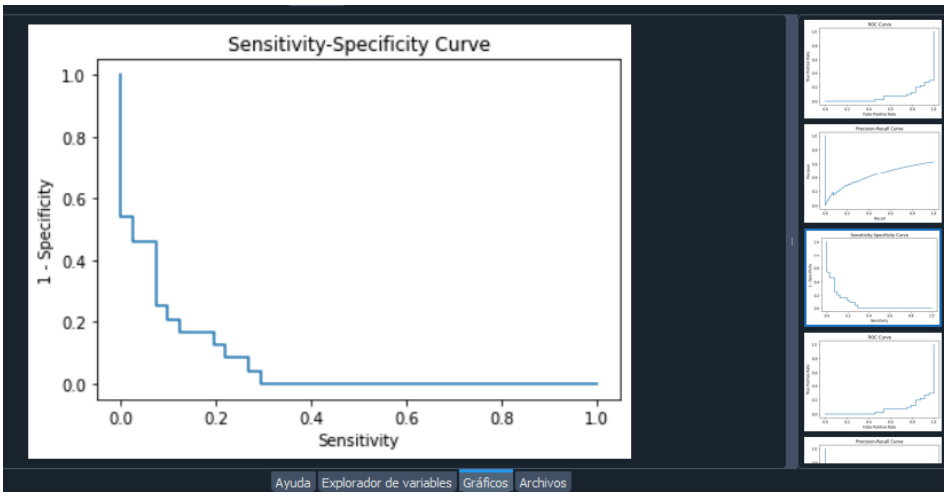


Ejecuto el código python

1.2 Resultado



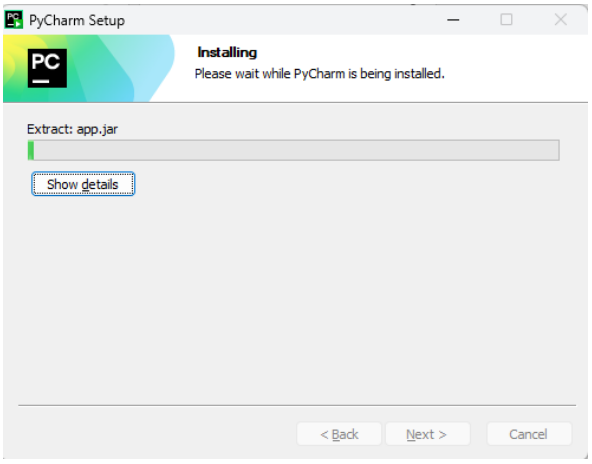




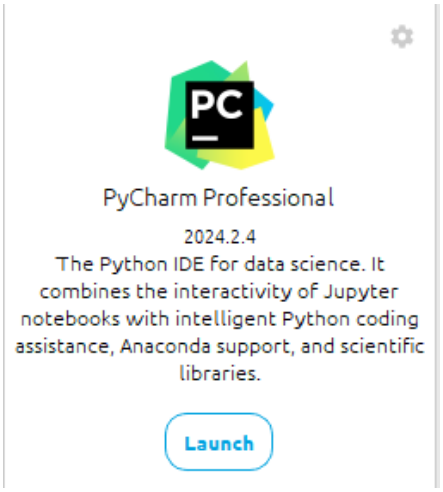
8. Ejecución en Pycharm

1.1 Pasos

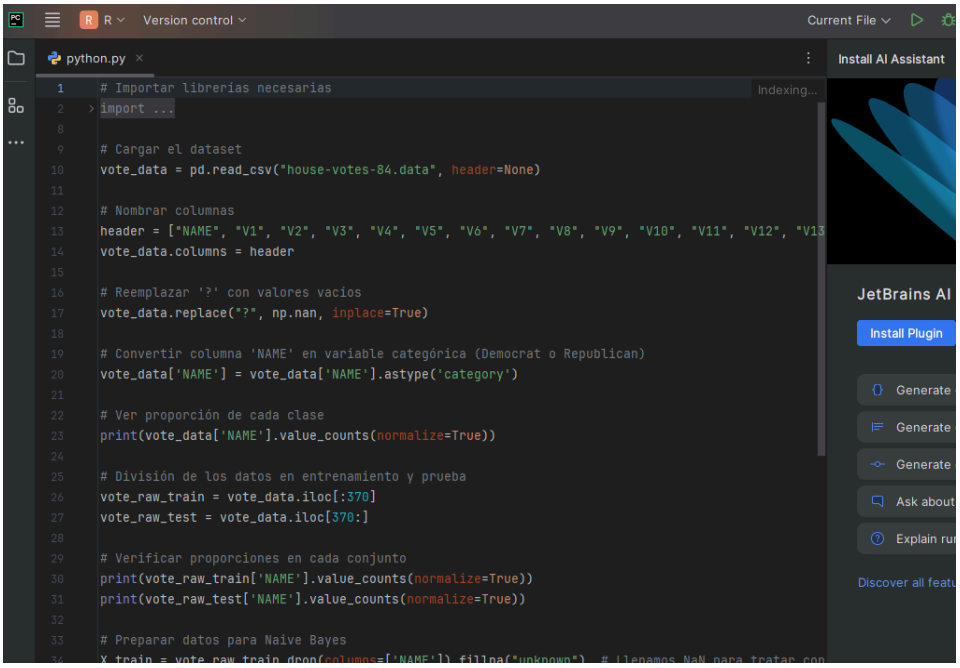
Instalo Pycharm, habiéndolo descargado en google



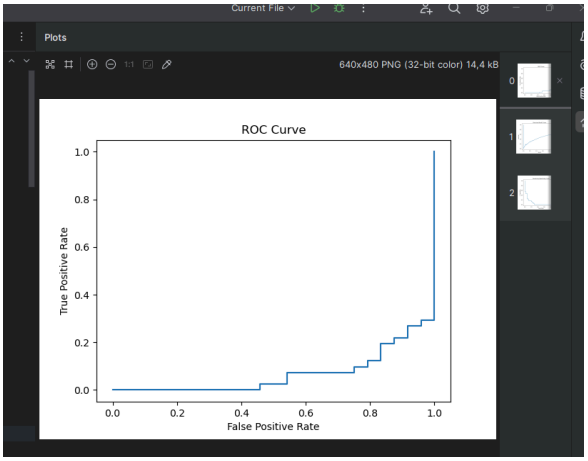
Ejecuto el programa en anaconda

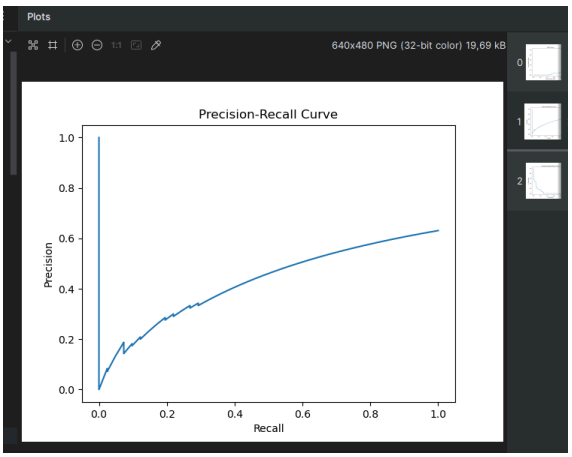


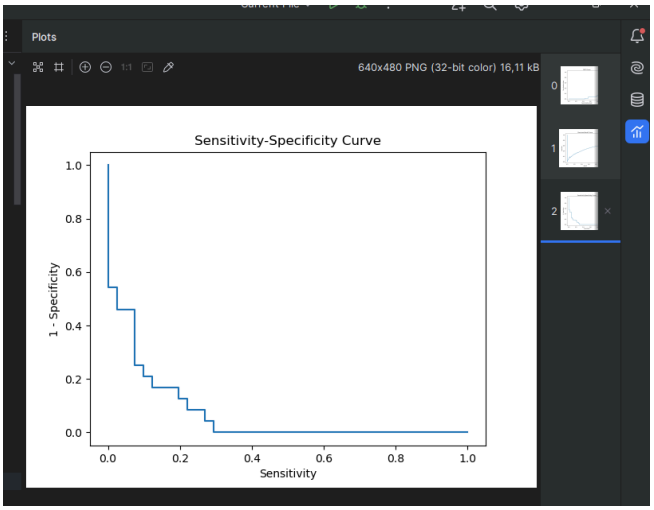
Ejecuto el código python



1.2 Resultado



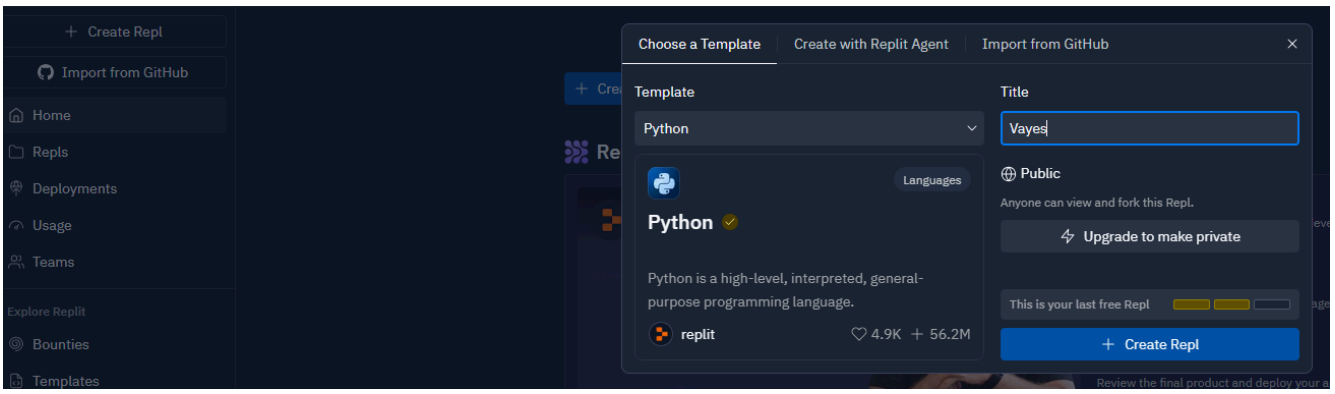




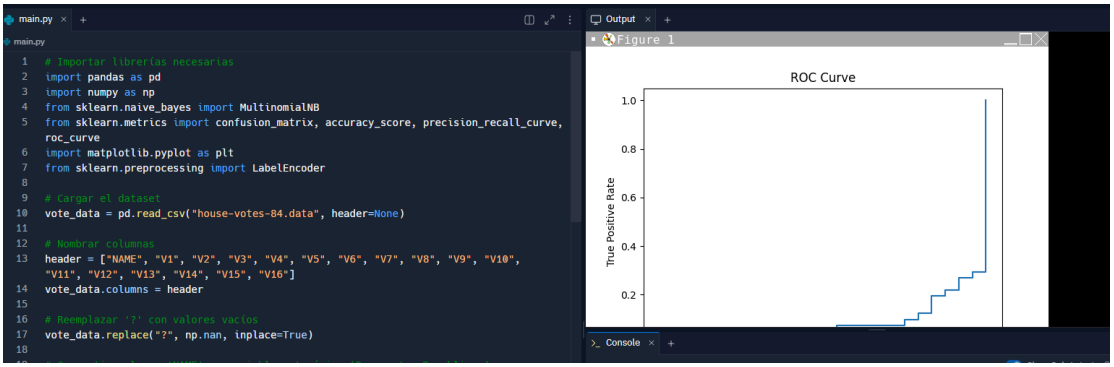
9. Ejecución en Replit

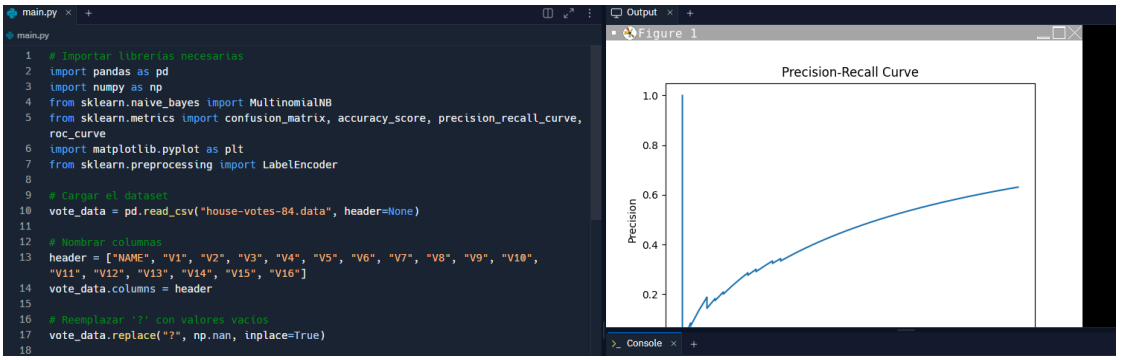
1.1 Pasos

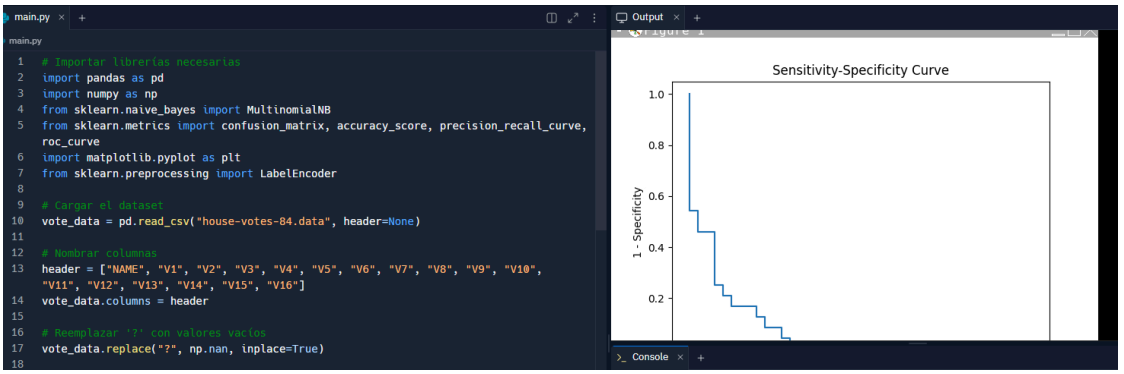
Creo el proyeco en Replit online y pongo el código python



1.2 Resultado



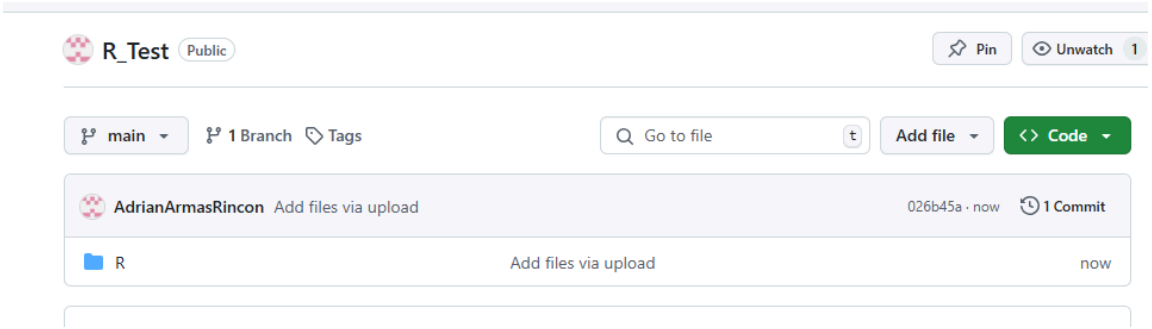




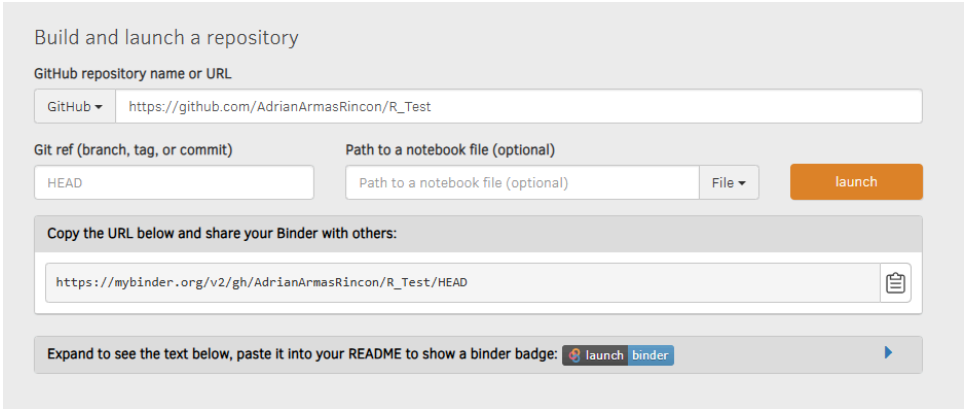
10. Ejecución en Binder

1.1 Pasos

Subo a Github la carpeta de la actividad



Pongo el repositorio en Binder y le doy a ejecutar



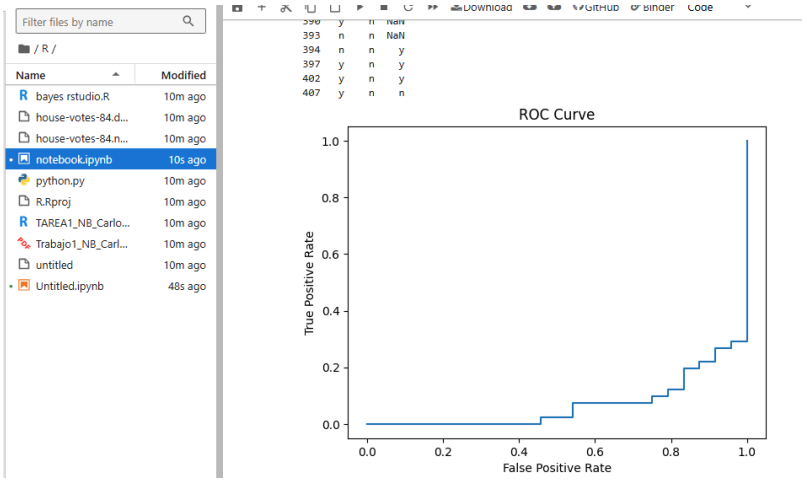
Instalo en binder pandas, mathlib y scikit

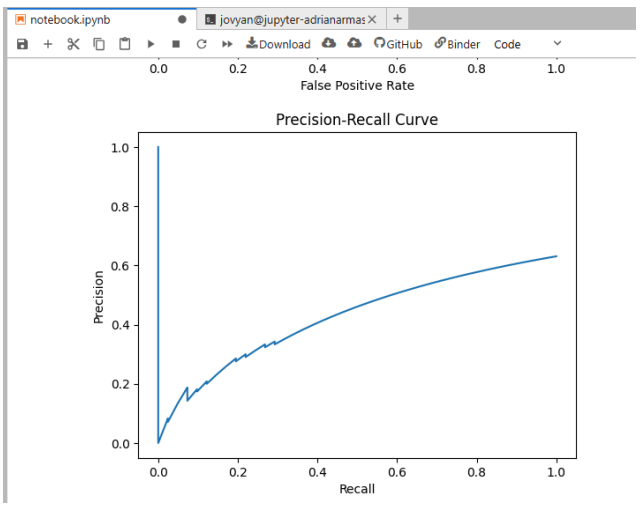
```
jovyan@jupyter-adrianarmasrincon-2dr-5ftest-2dy6ypvj78:~/R$ pip install scikit-learn
Collecting scikit-learn
  Downloading scikit_learn-1.5.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (13 kB)
Collecting numpy>=1.19.5 (from scikit-learn)
  Downloading numpy-2.1.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (62 kB)
Collecting scipy>=1.6.0 (from scikit-learn)
  Downloading scipy-1.14.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (60 kB)
Collecting joblib>=1.2.0 (from scikit-learn)
  Downloading joblib-1.4.2-py3-none-any.whl.metadata (5.4 kB)
Collecting threadpoolctl>=3.1.0 (from scikit-learn)
  Downloading threadpoolctl-3.5.0-py3-none-any.whl.metadata (13 kB)
Downloading scikit_learn-1.5.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (13.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.3/13.3 MB 13.2 MB/s eta 0:00:00
Downloading joblib-1.4.2-py3-none-any.whl (301 kB)
Downloading numpy-2.1.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (16.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.3/16.3 MB 12.9 MB/s eta 0:00:00
Downloading scipy-1.14.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (41.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 41.2/41.2 MB 13.7 MB/s eta 0:00:00
Downloading threadpoolctl-3.5.0-py3-none-any.whl (18 kB)
Installing collected packages: threadpoolctl, numpy, joblib, scipy, scikit-learn
```

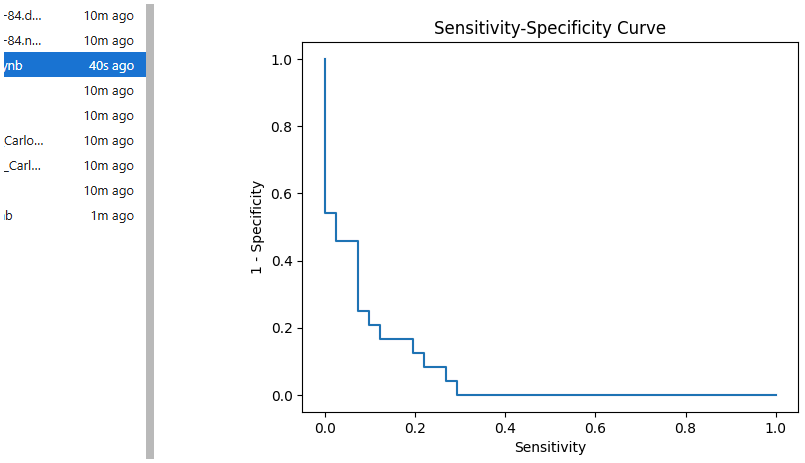
```
jovyan@jupyter-adrianarmasrincon-2dr-5ftest-2dy6ypvj78:~/R$ pip install matplotlib
Collecting matplotlib

jovyan@jupyter-adrianarmasrincon-2dr-5ftest-2dy6ypvj78:~/R$ pip install pandas numpy
Collecting pandas
```

1.2 Resultado





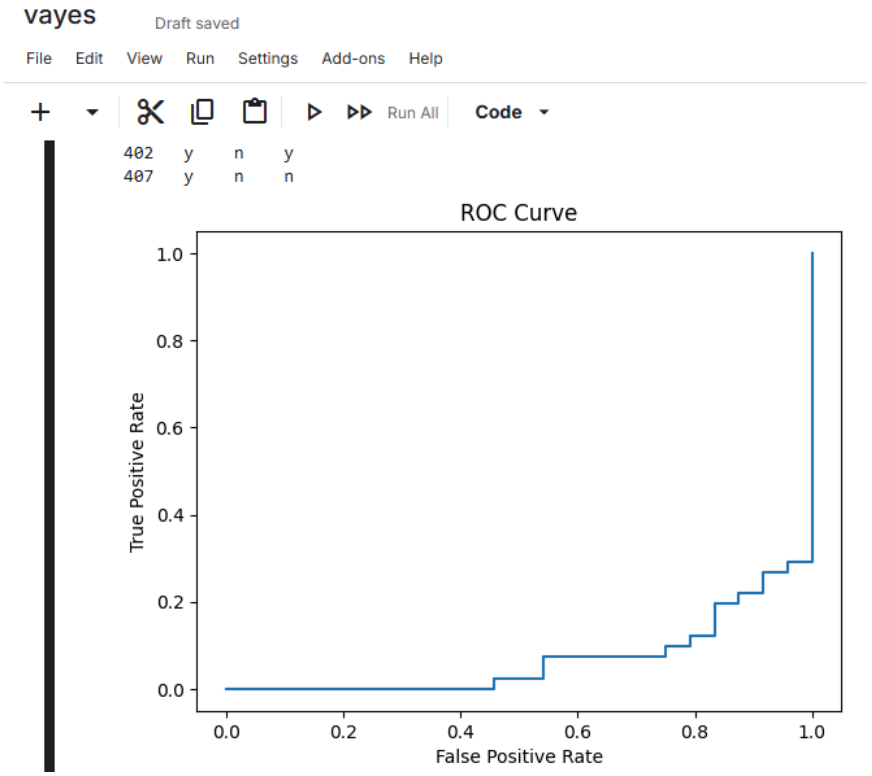


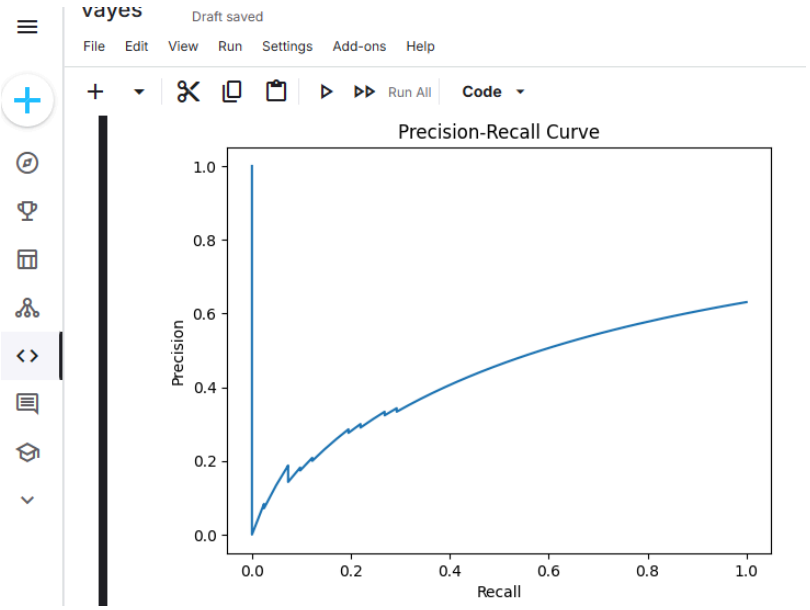
11. Ejecución en Kaggle

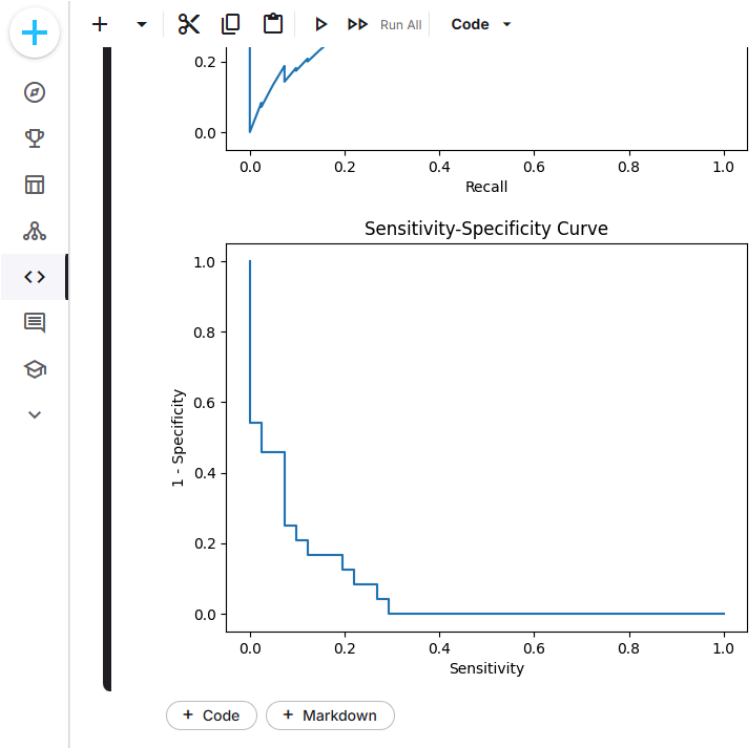
1.1 Pasos

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)
```

```r
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 #
########################################
```

```r
# 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