MEMORIA PRÁCTICA 3

MÉTODOS DE CLASIFICACIÓN

INGENIERÍA DEL CONOCIMIENTO

CURSO 2023-2024

Carlos Canero Mérida
Clara Rodríguez Prieto

ÍNDICE

1. LENGUAJE Y ENTORNO SELECCIONADO

Para realizar la práctica, hemos utilizado Python como lenguaje ya que, a pesar de ser un lenguaje con el que no estamos tan familiarizados, es una mejor opción para realizar esta práctica por las librerías y funciones que ofrece. Como entorno hemos utilizado Anaconda y los Notebooks de Jupyter, junto a GitHub Desktop para poder trabajar en conjunto.

2. PROCEDIMIENTO DE IMPLEMENTACIÓN

Para implementar la práctica, hemos creado un notebook llamado Practica3. El código se podría dividir en cuatro bloques:

1. Importación de bibliotecas y lectura de ficheros (celdas 2-4): utilizaremos la biblioteca NumPy para tener un mejor manejo de datos y leeremos los ficheros, tanto el que contiene los datos de entrenamiento como los ficheros de prueba.

2. Método de clasificación Agrupamiento Borroso (celdas 5-10): inicializamos los parámetros y definimos la funciones necesarias para realizar el método de clasificación. A continuación, realizamos el entrenamiento y realizamos los casos de prueba.

3. Método de clasificación de Lloyd (celdas 11-18): al igual que en el método de Agrupamiento Borroso, inicializamos los parámetros, definimos las funciones, entrenamos y realizamos los casos de prueba.

4. Método de clasificación de Bayes (celdas 19-24): a diferencia de los dos métodos anteriores, volveremos a leer el fichero que contiene los datos de entrenamiento ya que es necesario que los datos se guarden de forma diferente a los otros dos métodos. A continuación, definimos las funciones necesarias, entrenamos y realizamos los casos de prueba.

3. MANUAL DE USUARIO

Para poder ejecutar la práctica, es necesario tener la aplicación de Anaconda Navigator. Al ejecutar el programa, nos aparecerá una interfaz con una serie de aplicaciones como se muestra en la Imagen 1. Se elegirá Jupyter Notebook y se dará al botón de "Launch".

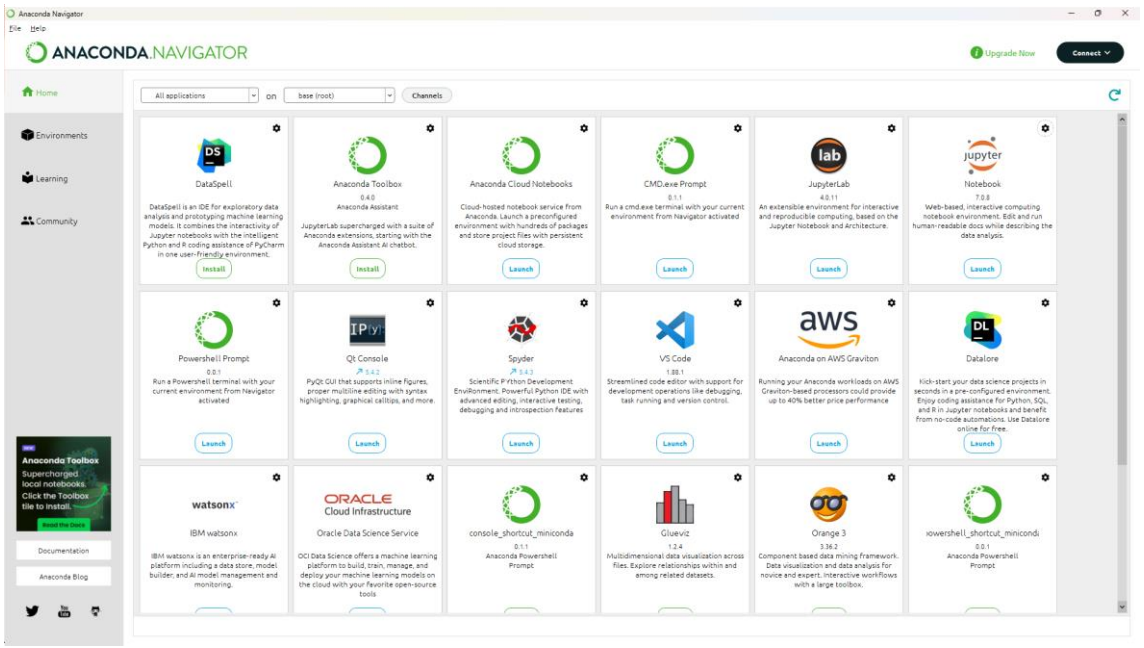

Imagen 1. Interfaz Anaconda Navigator

Una vez hayamos inicializado nuestro programa, se abrirá en nuestro navegador el programa de Jupyter Notebook. Desde ahí, navegaremos por nuestro directorio para buscar el notebook Practica3.ipynb, como se muestra en la Imagen 2.

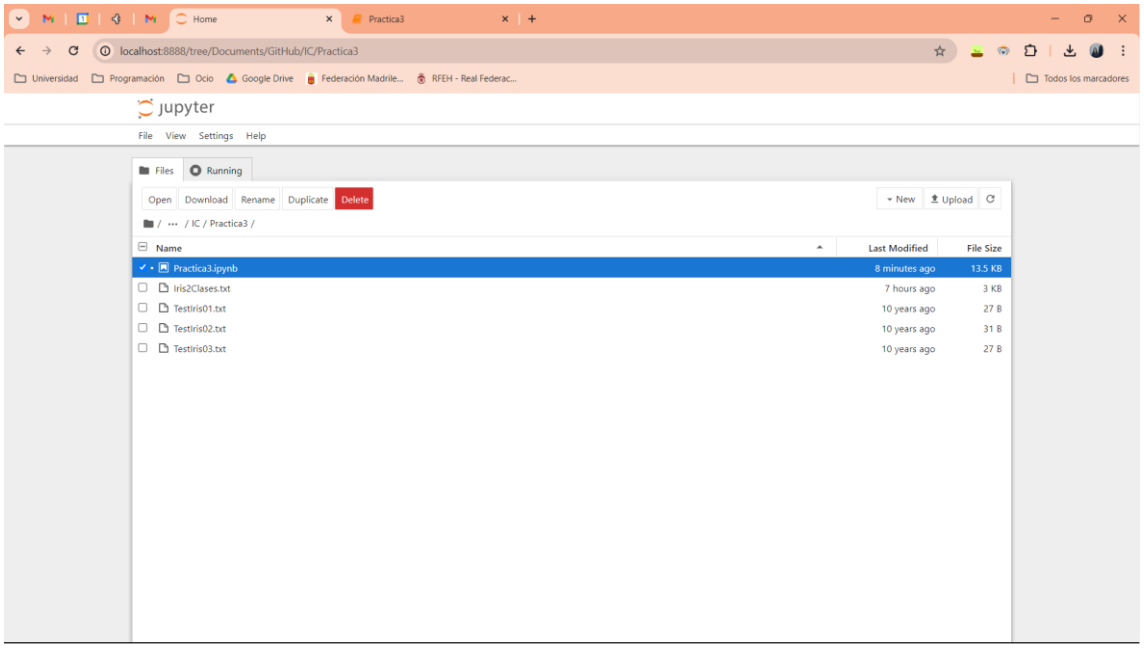

Imagen 2. Notebook de la práctica desde la aplicación de Jupyter

Realizaremos doble clic y se abrirá el fichero en una pestaña nueva, tal y como se puede ver en la Imagen 3. Es importante que los ficheros .txt se encuentren en la misma carpeta que el notebook.

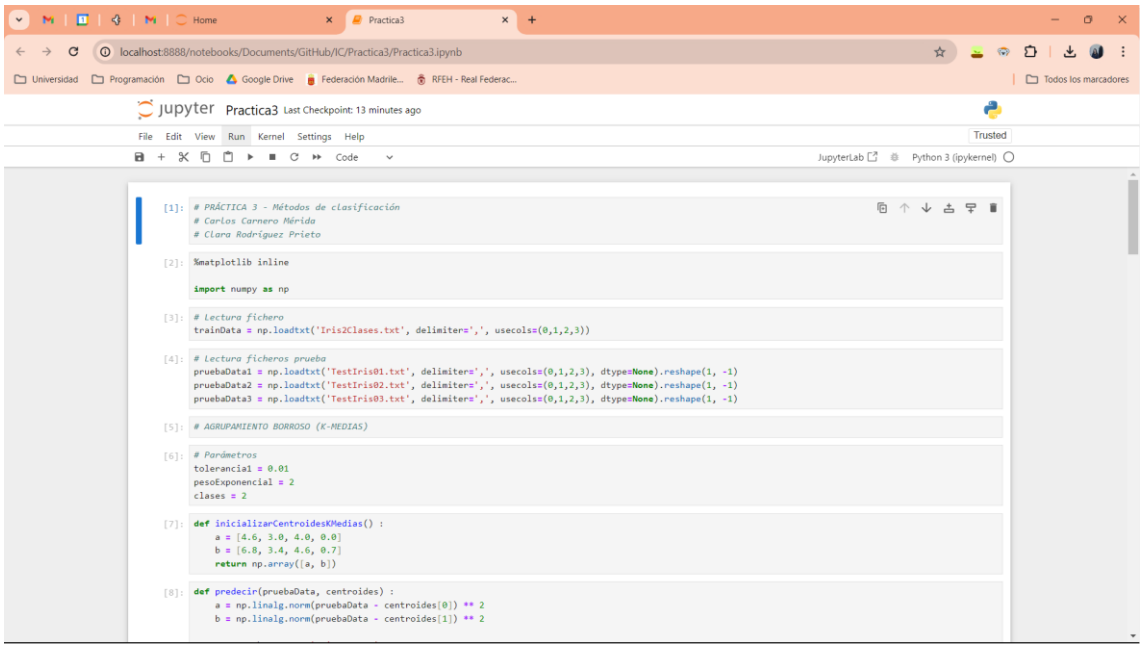

Imagen 3. Notebook Práctica3

Para iniciar la ejecución del fichero, tendremos que pulsar en el botón ⏩, el cual reinicia el Kernel y ejecuta cada una de las celdas de código. Al

pulsar, nos aparecerá el mensaje que se muestra en la Imagen 4. Le daremos a “Restart” y ejecutará el código.

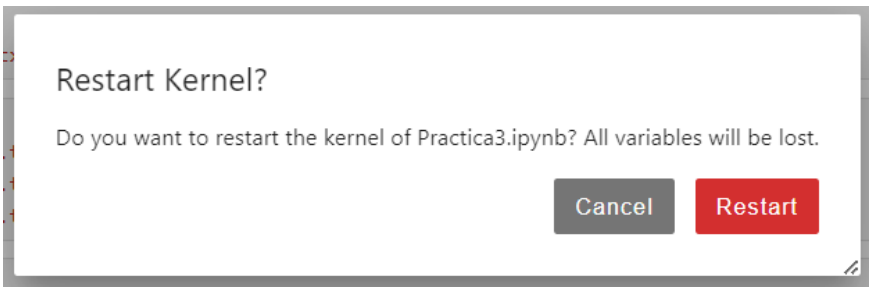

Imagen 4. Mensaje emergente que aparece tras pulsar para ejecutar el código

Una vez ejecutado el código, al lado de cada celda se mostrará un número, indicando la posición que ocupa en la ejecución (debe aparecer en orden ascendente). El resultado de cada uno de los métodos junto a sus centroides (en el caso de K-Medias y Lloyd) se pueden ver debajo de las celdas 9 y 10 en el caso del Agrupamiento Borroso (Imágenes 5 y 6), debajo de las celdas 17 y 18 en el caso de Lloyd (Imágenes 7 y 8), y debajo de la celda 24 en el caso de Bayes (Imagen 9).

```
[9]: # Entrenamiento
numMuestras = trainData.shape[0]
numCaracteristicas = trainData.shape[1]
centroides = inicializarCentroidesKMedias()
probAleatoria = np.random.rand(numMuestras, clases)
centroidesPrevios = np.zeros_like(centroides)
while np.linalg.norm(centroides - centroidesPrevios) > tolerancia1:
    probNormal = probAleatoria / np.sum(probAleatoria, axis=1)[:, np.newaxis]
    centroidesNuevos = np.dot((pesoExponencial ** probNormal).T, trainData) / np.sum((pesoExponencial ** probNormal).T
    distancias = np.linalg.norm(trainData[:, np.newaxis, :] - centroidesNuevos, axis=2)
    probNueva = 1 / np.sum((distancias[:, :, np.newaxis] / distancias[:, np.newaxis, :]) ** (2 / (pesoExponencial - 1)
    centroidesPrevios = centroides
    centroides = centroidesNuevos
    probAleatoria = probNueva

print(f"Centroides: {centroides}")

Centroides: [[5.46963928 3.09499468 2.85780349 0.78337191]
 [5.47236042 3.0930051  2.86619644 0.78662806]]
```

Imagen 5. Centroides del método de clasificación Agrupamiento borroso (K-Medias)

```
[10]: # Realizar casos de prueba
pruebaPredicciones1 = predecir(pruebaData1, centroides)
pruebaPredicciones2 = predecir(pruebaData2, centroides)
pruebaPredicciones3 = predecir(pruebaData3, centroides)

print("RESULTADOS PRUEBAS")
print(f"TestIris01.txt: {pruebaPredicciones1}")
print(f"TestIris02.txt: {pruebaPredicciones2}")
print(f"TestIris03.txt: {pruebaPredicciones3}")

RESULTADOS PRUEBAS
TestIris01.txt: Iris-setosa
TestIris02.txt: Iris-versicolor
TestIris03.txt: Iris-setosa
```

Imagen 6. Solución método de clasificación Agrupamiento borroso (K-Medias)

```
[17]: centroides = inicializarCentroidesLloyd()
for i in range(numMaxIteraciones) :
    centroidesAux = centroides.copy()
    clusters = asignarClusters(trainData, centroides)
    centroides = calcularCentroides(trainData, clusters, iteracion)
    if np.abs(centroides - centroidesAux).max() < tolerancia2:
        break
    centroides = centroidesAux + razonAprendizaje * (centroides - centroidesAux)

print(f"Centroides: {centroides}")
```

```
Centroides: [[5.71354143 2.94040341 4.37278345 0.9388713 ]
 [6.480634   3.07987479 4.58199052 1.1829462 ]]
```

Imagen 7. Centroides del método de clasificación Lloyd

```
[18]: # Realizar casos de prueba
pruebaClusters1 = asignarClusters(pruebaData1, centroides)
pruebaClusters2 = asignarClusters(pruebaData2, centroides)
pruebaClusters3 = asignarClusters(pruebaData3, centroides)

print(f"TestIris01.txt: {pruebaClusters1}")
print(f"TestIris02.txt: {pruebaClusters2}")
print(f"TestIris03.txt: {pruebaClusters3}")
```

```
TestIris01.txt: [0.]
TestIris02.txt: [1.]
TestIris03.txt: [0.]
```

Imagen 8. Solución método de clasificación de Lloyd

```
[24]: # Realizar casos de prueba
# Fichero prueba 1
pruebaVerosimilitudes1 = {}
for i in etiquetas :
    media = medias[i]
    covarianza = covarianzas[i]
    pruebaVerosimilitudes1[i] = calcularVerosimilitud(pruebaData1, media, covarianza)
pruebaPrediccion1 = min(pruebaVerosimilitudes1, key=pruebaVerosimilitudes1.get)
print(f"TestIris01.txt: {etiquetas[pruebaPrediccion1]}")

# Fichero prueba 2
pruebaVerosimilitudes2 = {}
for i in range(len(etiquetas)) :
    media = medias[i]
    covarianza = covarianzas[i]
    pruebaVerosimilitudes2[i] = calcularVerosimilitud(pruebaData2, media, covarianza)
pruebaPrediccion2 = min(pruebaVerosimilitudes2, key=pruebaVerosimilitudes2.get)
print(f"TestIris02.txt: {etiquetas[pruebaPrediccion2]}")

# Fichero prueba 3
pruebaVerosimilitudes3 = {}
for i in range(len(etiquetas)) :
    media = medias[i]
    covarianza = covarianzas[i]
    pruebaVerosimilitudes3[i] = calcularVerosimilitud(pruebaData3, media, covarianza)
pruebaPrediccion3 = min(pruebaVerosimilitudes3, key=pruebaVerosimilitudes3.get)
print(f"TestIris03.txt: {etiquetas[pruebaPrediccion3]}")
```

```
TestIris01.txt: Iris-setosa
TestIris02.txt: Iris-versicolor
TestIris03.txt: Iris-setosa
```

Imagen 9. Solución método de clasificación de Bayes