Proyecto- Big Data

Manual para correr el programa

Estudiante: Danny Valerio Ramírez

Cédula: 402340420

Clasificación de pacientes por riesgos de subir ataque cardiaco

Objetivo General

Determinar el riesgo que tiene los pacientes de sufrir un ataque cardiaco mediante una clasificación con base en sus características físicas y la calidad de vida del país donde viven utilizando algoritmos de maching lerning y pyspark.

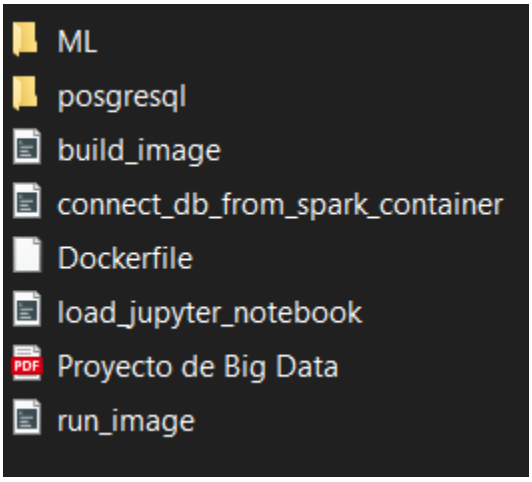
Objetivos específicos:

• Desarrollar un modelo predictivo “Multilayer perceptron classifier” con “Pyspark” para clasificar los pacientes

• Desarrollar un modelo predictivo “Random forest classifier” con “Pyspark” para clasificar los pacientes.

• Comparar los modelos “Multilayer perceptron classifier” y “Random forest classifier” para medir sus resultados por medio de precisión, recall y F1.

Archivos dentro de la carpeta



Nota: Se parte del supuesto que Docker está configurado en el equipo

Instrucciones crear la imágenes y contenedor principal

1. Descargue y descomprima el zip nombrado “Proyecto Danny Valerio” en un directorio local
2. Abrir el Command prompt de su sistema y buscar la dirección donde se guardó la carpeta “Proyecto Danny Valerio”
3. Para construir la imagen utilice el comando: <*Docker build --tag proyecto_danny_valerio .* >
4. *Para correr la imagen ejecute el comando* <*Docker run -p 8888:8888 -i -t proyecto_danny_valerio /bin/bash*>

Instrucciones para crear la Base de datos postgresql

1. Abrir el Command prompt de su sistema y buscar la dirección donde se guardó la carpeta “Proyecto Danny Valerio”
2. Para construir la imagen utilice el comando: < *docker run --name bigdata-db -e POSTGRES_PASSWORD=testPassword -p 5433:5432 -d postgres*>
3. *Recuerde que la clave para conectar será* ` *testPassword* `

Instrucciones de el código para procesamiento y guarda en la base

El archivo main recibe las direcciones de los 2 archivos csv y de la ubicación del Jar (driver de postgresql) y se encarga de preprocesar los datos y guardar 3 tablas en la base, de las cuales veremos el esquema más adelante

1. Asegúrese que tenga ambos contenedores corriendo, tanto el de la base como el principal.
2. Abra una terminal del contenedor principal, encontrara una carpeta llamada “ML” utilice el comando <*cd ML*> para ubicarse en ella.

3. Dentro de esta la carpeta “ML” y en el contenedor principla, ejecute el comando < ***spark-submit --jars postgresql-42.2.14.jar main.py data/heart_attack_prediction_dataset.csv data/Quality_of_Life_Index_by_Country_2023_Mid_Year.csv>***

Esquemas de las tablas creadas

Tabla ***“Heart” :*** almacena la información preprocesada del csv “*heart_attack_prediction_dataset.csv*”.

Columnas

- `Age: float (nullable = true)`
- `Sex: string (nullable = true)`
- `Cholesterol: float (nullable = true)`
- `Heart Rate: float (nullable = true)`
- `Diabetes: float (nullable = true)`
- `Family History: float (nullable = true)`
- `Smoking: float (nullable = true)`
- `Obesity: float (nullable = true)`
- `Alcohol Consumption: float (nullable = true)`
- `Exercise Hours Per Week: float (nullable = true)`
- `Diet: string (nullable = true)`
- `Previous Heart Problems: float (nullable = true)`
- `Medication Use: float (nullable = true)`
- `Stress Level: float (nullable = true)`
- `Sedentary Hours Per Day: float (nullable = true)`
- `Income: float (nullable = true)`
- `BMI: float (nullable = true)`
- `Triglycerides: float (nullable = true)`
- `Physical Activity Days Per Week: float (nullable = true)`
- `Sleep Hours Per Day: float (nullable = true)`
- `Country: string (nullable = true)`
- `systolic: float (nullable = true)`
- `diastolic: float (nullable = true)`
- `Heart Attack Risk: float (nullable = true)`

Tabla ***“country” :*** almacena la información preprocesada del csv “*Quality_of_Life_Index_by_Country_2023_Mid_Year.csv*”.

Columnas

- `Country: string (nullable = true)`
- `Quality of Life Index: float (nullable = true)`
- `Purchasing Power Index: float (nullable = true)`
- `Safety Index: float (nullable = true)`
- `Health Care Index: float (nullable = true)`
- `Cost of Living Index: float (nullable = true)`
- `Property Price to Income Ratio: float (nullable = true)`
- `Traffic Commute Time Index: float (nullable = true)`
- `Pollution Index: float (nullable = true)`
- `Climate Index: float (nullable = true)`

Tabla ***"Heart_country":*** almacena la información preprocesada y unida de los dos del csv anteriores.

Columnas

- `Age: float (nullable = true)`
- `Sex: string (nullable = true)`
- `Cholesterol: float (nullable = true)`
- `Heart Rate: float (nullable = true)`
- `Diabetes: float (nullable = true)`
- `Family History: float (nullable = true)`
- `Smoking: float (nullable = true)`
- `Obesity: float (nullable = true)`
- `Alcohol Consumption: float (nullable = true)`
- `Exercise Hours Per Week: float (nullable = true)`
- `Diet: string (nullable = true)`
- `Previous Heart Problems: float (nullable = true)`
- `Medication Use: float (nullable = true)`
- `Stress Level: float (nullable = true)`
- `Sedentary Hours Per Day: float (nullable = true)`
- `Income: float (nullable = true)`
- `BMI: float (nullable = true)`
- `Triglycerides: float (nullable = true)`
- `Physical Activity Days Per Week: float (nullable = true)`
- `Sleep Hours Per Day: float (nullable = true)`
- `Country: string (nullable = true)`
- `systolic: float (nullable = true)`
- `diastolic: float (nullable = true)`

- `Quality of Life Index: float (nullable = true)`
- `Purchasing Power Index: float (nullable = true)`
- `Safety Index: float (nullable = true)`
- `Health Care Index: float (nullable = true)`
- `Cost of Living Index: float (nullable = true)`
- `Property Price to Income Ratio: float (nullable = true)`
- `Traffic Commute Time Index: float (nullable = true)`
- `Pollution Index: float (nullable = true)`
- `Climate Index: float (nullable = true)`
- `Heart Attack Risk: float (nullable = true)`

Instrucciones para las pruebas unitarias

Se crearon 10 pruebas unitarias probar las funciones creadas: test_drop_columnas, test_select_columnas, test_split_col, test_split_col_2, test_cast_float, test_cast_string, test_Join_left, test_Join_inner, test_drop_null y test_order_columns.

1. Dentro del contenedor principal encontrara una carpeta llamada “ML” utilice el comando <*cd ML*> para ubicarse en ella.
2. Para correr las pruebas ejecutar el comando <*pytest -s*>. Si desea omitir el contenido de estas utilizar <*pytest*>.

Instrucciones para correr el Notebook

1. Asegúrese que tenga ambos contenedores corriendo, tanto el de la base como el principal.
2. Dentro de una ternimal del contenedor principal corra el comando: <*jupyter notebook --ip=0.0.0.0 --port=8888 --allow-root*>
3. *En el CMD aparecerá el link para abrir el jupyter en el navegador, suele ser el tercero, similar a “http://127.0.0.1:8888/?token=596eba35e83348b583518b7e5fca5dab32a5cbc7f0b599af”*
4. *Dentro de la carpeta “ML” o si ya esta en ella, encontrara el Notebook, el controlador para la base y una carpeta csv con los conjuntos de datos.*

5. *Abra el archivo del notebook “proyecto_Danny_Valerio”.*