

Estudiante: Steven Jimenez

Correo: Steven.jimenez.bustamante@gmail.com

Carnet 201229730

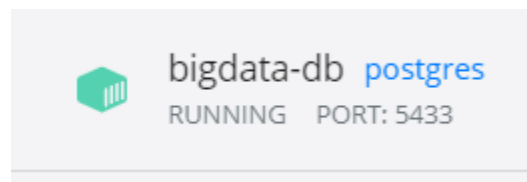
Modo de ejecución del código (Windows Terminal)

Creación de imagen/contenedor de la base de datos

Para la creación de la base de datos ejecute el siguiente código en la terminal

```
1. docker rm bigdata-db
2. docker run --name bigdata-db ^
3.   -e POSTGRES_PASSWORD=testPassword ^
4.   -p 5433:5432 ^
5.   -d postgres
```

Si la creación de la base de datos fue exitosa, debería observarse un Container de Docker corriendo la base de datos llamada postgres



Creación del contenedor principal – Ubicación del Código Principal

1.Descomprima el archivo .zip/.rar en una carpeta de su preferencia.

2.Abra la terminal de comandos de Windows

3.En terminal, diríjase a la carpeta que fue descomprimida. En mi caso esto lo logro mediante el comando

```
1. cd C:\Users\steve\OneDrive\004BigData\Proyecto\ProyectoBigData
```

4.Para crear la imagen en Docker, ejecute el siguiente comando

```
1. docker build --tag bigdata .
```

5.Para ejecutar/correr el contenedor, ejecute el siguiente comando

```
1. docker run -p 8888:8888 -i -t bigdata /bin/bash
```

Lo esperado es que el contenedor inicie de manera correcta, y espere instrucciones en modo IDLE:

```
1. bash-5.0#
```

Diríjase a la carpeta del proyecto mediante el comando

```
1. cd codigoProyecto
```

Descripción de los datasets:

Indicadores globales

Este dataframe contiene indicadores por cada país (country) como el porcentaje de pobreza (poverty_percent), Producto Interno Bruto Per Capita (gdp_per_capita), población (population) y la cantidad promedio de años de estudio para persona mayores de 25 años (years_of_education)

country	Code	poverty_percent	gdp_per_capita	population	years_of_education
Afghanistan	AFG	51.7	1929.0	35383028	3.9
Albania	ALB	0.858683	10342.0	2886427	10.0
Algeria	DZA	0.3268976	14331.0	40551398	8.0

Carga inicial de Indicadores GlobalesDF
Qty Filas: 254
Cantidad Columnas: 6
root
-- country: string (nullable = true)
-- Code: string (nullable = true)
-- poverty_percent: double (nullable = true)
-- gdp_per_capita: double (nullable = true)
-- population: long (nullable = true)
-- years_of_education: double (nullable = true)

Atletas Rio de Janeiro 2016

Este dataset contiene la información de todos los competidores que participaron en las olimpiadas de Río de Janeiro 2016, tanto como los competidores que Sí Ganaron medallas como los competidores que No Ganaron medallas. Este dataset contiene distintos features de cada competidor como ID, Nombre, Nacionalidad, Sexo, Fecha de Nacimiento, Altura (height), Peso(weight), Disciplina(sport), Qty Oro ganados, Qty Plata ganados, Qty bronce ganados, countrypopulation, countrygdp.

id	name	nationality	sex	dob	height	weight	sport	gold	silver	bronze	country	countrypopulation	countrygdp_per_capita
736041664	A Jesus Garcia	ESP	male	254931	1.72	64.0	athletics	0	0	0	Spain	46418269	25831.58231
87689776	Aauri Lorena Bokesa	ESP	female	32491	1.8	62.0	athletics	0	0	0	Spain	46418269	25831.58231
803161695	Abdelaziz Merzougui	ESP	male	33480	1.75	67.0	athletics	0	0	0	Spain	46418269	25831.58231

Carga inicial de AtletasDF
Qty Filas: 11538
Cantidad Columnas: 14
root
-- id: long (nullable = true)
-- name: string (nullable = true)
-- nationality: string (nullable = true)
-- sex: string (nullable = true)
-- dob: long (nullable = true)
-- height: double (nullable = true)
-- weight: double (nullable = true)
-- sport: string (nullable = true)
-- gold: long (nullable = true)
-- silver: long (nullable = true)
-- bronze: long (nullable = true)
-- country: string (nullable = true)
-- countrypopulation: long (nullable = true)
-- countrygdp_per_capita: double (nullable = true)

Objetivo Predictivo

Predecir si competidor Gana / “No Gana” medalla en olimpiadas (GanaMedalla) en base a los features de Indicadores Globales e información del Atleta. (Revisar a detalle la investigación preliminar)

Después de la unión de los datos

Features	Parámetro Para Predecir
Dato 1.1.1: Extreme Poverty (por país): Dato 1.1.2: GDP per capita (por país): Dato 1.1.3: Population (por país): Dato 1.1.4: Education – Years of Schooling (por país): Dato 1.1.5: Country (país): Dato 1.2.1: Competitor ID: Dato 1.2.2: Nationality (Country): Dato 1.2.3: Sex: Dato 1.2.4: Height: Dato 1.2.5: Weight: Dato 1.2.6: Gold Dato 1.2.7: Silver Dato 1.2.8: Bronze	GanaMedalla. Predecir si competidor Gana / “No Gana” medalla en olimpiadas.

Cargado y preprocesamiento de datos:

Pruebas Unitarias

Para ejecutar el código de pruebas unitarias ejecute el siguiente comando

```
1.  pytest testing.py
```

A continuación, se describe el objetivo de cada prueba unitaria

Nombre de la prueba	Objetivo de la prueba
test_loading_IndicesGlobalesDF	Comprobar que el "testDS1_loadcsv_world_indicators.csv" se cargó correctamente
test_loading_AleatasDF	Comprobar que el " testDS2_loadcsv_athletes.csv" se cargó correctamente
test_transformDatasetIndicesGlobales	Comprobar que se seleccionó únicamente las columnas "country" , "poverty_percent" , "gdp_per_capita" , "population" , "years_of_education" del dataframe IndicesGlobalesDF
test_transformDatasetAleatas	<p>1. Comprobar que los participantes que registran alguna medalla como "Null", esos espacios se rellenen con cantidad 0 (columnas "gold" , "silver", "bronze")</p> <p>2. Comprobar que se asignó a la columna "total_Medallas" la suma de la cantidad de medallas ganadas por cada participantes</p> <p>3. Comprobar que se asignó a la columna "TieneMedalla", si un participante ganó al menos una medalla para lo cual se asigna el valor de 1, si el participante no ganó medalla se asigna el valor de 0</p> <p>Después de los pasos anteriores</p> <p>4. Comprobar que se seleccionó únicamente las columnas "country", "sex", "height", "weight", "sport", "TieneMedalla"</p>
test_imputacionIndicesGlobales	Por falta de conocimiento técnico en el área, se removieron todas las filas que tuvieran valores "Null". Comprobar que este imputación se realizó de manera correcta.
test_imputacionAleatas	Por falta de conocimiento técnico en el área, se removieron todas las filas que tuvieran valores "Null". Comprobar que este imputación se realizó de manera correcta.
test_joinDataframes	Comprobar que la unión de los dataframes se realizó de manera correcta - innerJoin mediante columna "country"
test_MuestraEstratificado	<p>Supuestos (al observar los datos):</p> <p>La cantidad de participantes por disciplina/deporte que No Ganaron Medalla "TieneMedalla"=0, siempre es mayor a la cantidad de participantes por la misma disciplina/deporte que Si Ganaron "TieneMedalla"=1.</p> <p>Bajo el supuesto anterior, el objetivo de esta prueba consiste en hacer un muestreo estratificado para solucionar el desbalance de esta dataset. El muestro consiste que la cantidad de participantes por disciplina/deporte y Sexo/Genero que No Ganaron Medalla "TieneMedalla"=0 es igual o menor que la cantidad de participantes por la misma disciplina/deporte y mismo Sexo/Genero que Si Ganaron Medalla "TieneMedalla"=1</p>

Importante: A la función `EncodeAndStandardizeFeatures(sample_df)` no logré implementar la forma correcta de la prueba unitaria. La función `EncodeAndStandardizeFeatures(sample_df)` se encarga de normalizar todos los Features de variables continuas, además se encarga de aplicar el algoritmo de One Hot Encoder las Features de Variables Categóricas.

Para ejecutar el código principal, corra la siguiente instrucción

```
1. spark-submit \  
2. --driver-class-path postgresql-42.2.14.jar \  
3. --jars postgresql-42.2.14.jar \  
4.CodigoProyecto.py \  
5. dataset1_world_indicators.csv \  
6. dataset2_athletes.csv \
```

Materialización en PostgreSQL

Función para escribir a la base de datos: `escribir_en_DB(DF,nombreDF)`

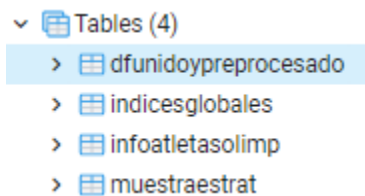
La función de escritura a la base de datos, toma como parámetros el Dataframe (DF) que queremos escribir en la base de datos y el nombre de la tabla (`nombreDF`) con el que queremos guardar el DF.

Función para escribir a la base de datos: `leer_desde_DB(nombreDF)`

La función de lectura desde la base de datos, toma como nombre de la tabla (`nombreDF`) que queremos leer/obtener desde la base de datos y la retorna como un Dataframe de PySpark

Tablas escritas en la base de datos

Se logran escribir 4 tablas en la base de datos.



A continuación se explica cada tabla

Tabla indicesglobales

En esta tabla se guardó el dataframe de Indices Globales preprocesado. El dataframe que se guardó en la base de datos es el asignado a la variable "IndicesPreprocesadosDF", y se realizó mediante el siguiente código

```
escribir_en_DB(IndicesPreprocesadosDF , "IndicesGlobales")
```

Query Editor

Query History

1

SELECT * FROM public.indicesglobales

2

Data Output

Explain

Messages


Notifications

	country text	poverty_percent double precision	gdp_per_capita double precision	population bigint	years_of_education double precision
1	Afghanistan	51.7	1929	35383028	3.9
2	Albania	0.858683	10342	2886427	10
3	Algeria	0.3268976	14331	40551398	8
4	Andorra	0.2298	40897	77295	10.5
5	Angola	42.16528	8453	28842482	5.1
6	Argentina	1.070877	18875	43508459	10.95

Tabla infoatletasolimp

En esta tabla se guardó el dataframe de Información de Atletas preprocesado. El dataframe que se guardó en la base de datos es el asignado a la variable "AtletasPreprocesadosDF", y se realizó mediante el siguiente código

```
escribir_en_DB(AtletasPreprocesadosDF , "InfoAtletasOlimp")
```



public.infoatletasolimp/postgres/postgres@bigdataDB

Query Editor

Query History

1

SELECT * FROM public.infoatletasolimp

2

LIMIT 100

3

Data Output

Explain

Messages

Notifications






	<div>country</div> <div>text</div>	<div> sex</div> <div>text</div>	<div> height</div> <div>double precision</div>	<div> weight</div> <div>double precision</div>	<div> sport</div> <div>text</div>	<div> TieneMedalla</div> <div>bigint</div>
1	Spain	male	1.72	64	athletics	0
2	Spain	female	1.8	62	athletics	0
3	Spain	male	1.75	67	athletics	0
4	Spain	male	1.82	71	athletics	0
5	Spain	male	1.93	89	volleyball	0
6	Spain	female	1.62	65	archery	0

Tabla muestraestrat

En esta tabla se guardó el dataframe del muestreo estratificado. El dataframe que se guardó en la base de datos es el asignado a la variable “muestraEstratificadaDF”, y se realizó mediante el siguiente código

```
escribir_en_DB(muestraEstratificadaDF , "MuestraEstrat")
```

Query Editor		Query History						Scratch Pad					
1 SELECT * FROM public.muestraestrat													
2 LIMIT 100													
3													
Data Output		Explain		Messages		Notifications							
	country text	poverty_percent double precision	gdp_per_capita double precision	population bigint	years_of_education double precision	sex text	height double precision	weight double precision	sport text	TieneMedalla bigint			
1	Egypt	2.366573	11351	94447071	7.2	female		1.63	75 shooting	0			
2	Poland	0.004540368	24838	37989218	12.2	female		1.69	61 shooting	0			
3	Taiwan	0	42165	23618201		12 female		1.57	52 shooting	0			
4	Australia	0.4984838	48845	24262710	12.9	female		1.57	51 shooting	0			
5	Japan	0.7278879	37465	127763267	12.7	female		1.62	58 shooting	0			
6	Russia	0.0292134	23635	145275374	12	female		1.65	50 shooting	0			

(No existente) Tabla dfunidoypreprocesado

Basado en lo aprendido del curso, yo asumí que antes de escribir el dataframe a la base de datos era **requisito Normalizar los features de variables continuas, además de aplicar el algoritmo de One Hot Encoder a los features de variables categóricas**. Llegué a un punto donde todas las columnas de mi dataframe se podían escribir a la base de datos, pero la columna (“featuresCategoricos”) donde se ejecutó el OneHotEncoding quedó bajo el formato “vector”, lo cual no me permitió escribir todo el dataframe a la base de datos. Además, no logré convertir esta columna de “vector” a un “string” u otra variable que si fuera permitido guardar en la base de datos. Esta transformación de Normalización y OneHotEncoding se ejecuta mediante la función EncodeAndStandardizeFeatures(sample_df). De haber sido posible, mi intención sería guardar el dataframe preprocesado en la base de datos bajo esta estructura

```
root
|-- featuresCategoricos: vector (nullable = true)
|-- poverty_percent: double (nullable = true)
|-- gdp_per_capita: double (nullable = true)
|-- population: long (nullable = true)
|-- years_of_education: double (nullable = true)
|-- height: double (nullable = true)
|-- weight: double (nullable = true)
|-- TieneMedalla: long (nullable = true)
```

En el main() del código, la escritura a la base de datos se pudo haber realizado bajo un instrucción similar a

```
escribir_en_DB(DF_Unido_y_preprocesado , "DFUnidoypreprocesado")
```

Modelos de predicción

Supuestos:

El dataframe preprocesado que tuvo que haber sido escrito a la base de datos es la variable "DF_Unido_y_preprocesado". Como no pude guardar a la base de datos, por lo tanto tampoco pude leer desde la base de datos, seguiré trabajando con el mismo dataframe pero bajo otro nombre "desdeDB_DF_Unido_y_preprocesado", donde "desdeDB_DF_Unido_y_preprocesado = DF_Unido_y_preprocesado"

```
417     ###Aquí debió haber sucedido la lectura desde la base de datos###
418     desdeDB_DF_Unido_y_preprocesado = DF_Unido_y_preprocesado
419
420     print("resultado ClasificadorArbolDecision")
421     ClasificadorArbolDecision(desdeDB_DF_Unido_y_preprocesado)
422     print("resultado ClasificadorRegresionLogistica")
423     ClasificadorRegresionLogistica(desdeDB_DF_Unido_y_preprocesado)
424
```

Clasificador Árbol de decisión:

El modelo de este clasificador se entrenó con el algoritmo de validación cruzada K-Fold con una cantidad de 5 folds. La función que describe a este modelo es ClasificadorArbolDecision(preprocessedDF)

Clasificador Regresión Logística:

El modelo de este clasificador se entrenó con el algoritmo de validación cruzada K-Fold con una cantidad de 5 folds. La función que describe a este modelo es ClasificadorRegresionLogistica(preprocessedDF)

Análisis de resultados

Resultados - Clasificador Árbol de decisión:

El clasificador indica que tiene un accuracy del 100%. Aunque suena un clasificador ideal, uno como analista de datos se cuestiona la perfección del algoritmo. En mi caso estuve revisando todo el código para detectar fallos, pero no encontré ninguno, además de que se me acabó el tiempo. Estos son los resultados que puedo mostrar.

```
dtpredictionsTrain
areaUnderROC Train 1.0
*****

dtpredictionsTest
areaUnderROC Test 1.0
*****
```

Resultados - Clasificador Regresión Logística:

El clasificador indica que tiene un accuracy del 100%. Aunque suena un clasificador ideal, uno como analista de datos se cuestiona la perfección del algoritmo. En mi caso estuve revisando todo el código para detectar fallos, pero no encontré ninguno, además de que se me acabó el tiempo. Estos son los resultados que puedo mostrar.

```
dtpredictionsTrain
areaUnderROC Train 1.0
*****

dtpredictionsTest
areaUnderROC Test 1.0
*****
```