Variables

1. Años
2. Sexo
3. Tipo de dolor torácico (4 valores)
4. Presión arterial en reposo
5. Colesterol sérico en mg/dl
6. Azúcar en sangre en ayunas > 120 mg/dl
7. Resultados electrocardiográficos en reposo (valores 0,1,2)
8. Frecuencia cardíaca máxima alcanzada
9. Angina inducida por el ejercicio
10. Oldpeak = depresión del ST inducida por el ejercicio en relación con el reposo
11. Pendiente del segmento ST de ejercicio máximo
12. Número de vasos principales (0-3) coloreados por fluoroscopia
13. Thal: 3 = normal; 6 = defecto fijo; 7 = defecto reversible
14. Variable Interes

Setup

```
!pip install pyspark
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pyspark
  Downloading pyspark-3.4.0.tar.gz (310.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 310.8/310.8 MB 2.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.4.0-py2.py3-none-any.whl size=311317130 sha256=1d2eb22ea7fc16a810f5ef0dde5ac5fb96cdd6be5
  Stored in directory: /root/.cache/pip/wheels/7b/1b/4b/3363a1d04368e7ff0d408e57ff57966fcdf00583774e761327
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.4.0
```

```
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession
import pyspark.sql
from pyspark.sql.functions import col, when
from pyspark.sql import functions as F
from pyspark.ml.feature import VectorAssembler, StandardScaler
from pyspark.sql.types import *
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import BinaryClassificationEvaluator, MulticlassClassificationEvaluator


# sc.stop()
```

Inicializando Spark

```
conf = SparkConf().setMaster("local").setAppName("Enfermedad al Corazon")

# Inicializo el Spark Context
sc = SparkContext(conf = conf)

distFile = sc.textFile('https://raw.githubusercontent.com/Rwyld/Data-Science-Models/main/BigData/Heart%20Data.data')

#distFile.collect() # visualización de datos
```

Creando el DataFrame

```
spark = SparkSession.builder.getOrCreate()

columnas = ['Años', 'Sexo', 'TipoDolor', 'PresionArterial', 'Colesterol', 'AzucarAyuna', 'Electrocardio', 'FrecuenciaCardio', 'Angina', 'Oldp

data = distFile.map(lambda x: x.split()).toDF(columnas)

data.show()
```

```
+----+----+---------+--------------+----------+-----------+-------------+---------------+------+-------+-----------+-----------+----+-
|Años|Sexo|TipoDolor|PresionArterial|Colesterol|AzucarAyuna|Electrocardio|FrecuenciaCardio|Angina|Oldpeak|PendienteST|NumeroVasos|Thal|V
+----+----+---------+--------------+----------+-----------+-------------+---------------+------+-------+-----------+-----------+----+-
|70.0| 1.0|      4.0|         130.0|     322.0|        0.0|          2.0|          109.0|   0.0|    2.4|        2.0|        3.0| 3.0|
|67.0| 0.0|      3.0|         115.0|     564.0|        0.0|          2.0|          160.0|   0.0|    1.6|        2.0|        0.0| 7.0|
|57.0| 1.0|      2.0|         124.0|     261.0|        0.0|          0.0|          141.0|   0.0|    0.3|        1.0|        0.0| 7.0|
|64.0| 1.0|      4.0|         128.0|     263.0|        0.0|          0.0|          105.0|   1.0|    0.2|        2.0|        1.0| 7.0|
|74.0| 0.0|      2.0|         120.0|     269.0|        0.0|          2.0|          121.0|   1.0|    0.2|        1.0|        1.0| 3.0|
|65.0| 1.0|      4.0|         120.0|     177.0|        0.0|          0.0|          140.0|   0.0|    0.4|        1.0|        0.0| 7.0|
|56.0| 1.0|      3.0|         130.0|     256.0|        1.0|          2.0|          142.0|   1.0|    0.6|        2.0|        1.0| 6.0|
|59.0| 1.0|      4.0|         110.0|     239.0|        0.0|          2.0|          142.0|   1.0|    1.2|        2.0|        1.0| 7.0|
|60.0| 1.0|      4.0|         140.0|     293.0|        0.0|          2.0|          170.0|   0.0|    1.2|        2.0|        2.0| 7.0|
|63.0| 0.0|      4.0|         150.0|     407.0|        0.0|          2.0|          154.0|   0.0|    4.0|        2.0|        3.0| 7.0|
|59.0| 1.0|      4.0|         135.0|     234.0|        0.0|          0.0|          161.0|   0.0|    0.5|        2.0|        0.0| 7.0|
|53.0| 1.0|      4.0|         142.0|     226.0|        0.0|          2.0|          111.0|   1.0|    0.0|        1.0|        0.0| 7.0|
|44.0| 1.0|      3.0|         140.0|     235.0|        0.0|          2.0|          180.0|   0.0|    0.0|        1.0|        0.0| 3.0|
|61.0| 1.0|      1.0|         134.0|     234.0|        0.0|          0.0|          145.0|   0.0|    2.6|        2.0|        2.0| 3.0|
|57.0| 0.0|      4.0|         128.0|     303.0|        0.0|          2.0|          159.0|   0.0|    0.0|        1.0|        1.0| 3.0|
|71.0| 0.0|      4.0|         112.0|     149.0|        0.0|          0.0|          125.0|   0.0|    1.6|        2.0|        0.0| 3.0|
|46.0| 1.0|      4.0|         140.0|     311.0|        0.0|          0.0|          120.0|   1.0|    1.8|        2.0|        2.0| 7.0|
|53.0| 1.0|      4.0|         140.0|     203.0|        1.0|          2.0|          155.0|   1.0|    3.1|        3.0|        0.0| 7.0|
|64.0| 1.0|      1.0|         110.0|     211.0|        0.0|          2.0|          144.0|   1.0|    1.8|        2.0|        0.0| 3.0|
|40.0| 1.0|      1.0|         140.0|     199.0|        0.0|          0.0|          178.0|   1.0|    1.4|        1.0|        0.0| 7.0|
+----+----+---------+--------------+----------+-----------+-------------+---------------+------+-------+-----------+-----------+----+-
only showing top 20 rows
```

Creando la Variable "Enfermo"

```
data = data.withColumn("Enfermo", when(col("Thal") > 3, 1).otherwise(0))
data.show()
```

```
+----+----+---------+--------------+----------+-----------+-------------+---------------+------+-------+-----------+-----------+----+-
|Años|Sexo|TipoDolor|PresionArterial|Colesterol|AzucarAyuna|Electrocardio|FrecuenciaCardio|Angina|Oldpeak|PendienteST|NumeroVasos|Thal|V
+----+----+---------+--------------+----------+-----------+-------------+---------------+------+-------+-----------+-----------+----+-
|70.0| 1.0|      4.0|         130.0|     322.0|        0.0|          2.0|          109.0|   0.0|    2.4|        2.0|        3.0| 3.0|
|67.0| 0.0|      3.0|         115.0|     564.0|        0.0|          2.0|          160.0|   0.0|    1.6|        2.0|        0.0| 7.0|
|57.0| 1.0|      2.0|         124.0|     261.0|        0.0|          0.0|          141.0|   0.0|    0.3|        1.0|        0.0| 7.0|
|64.0| 1.0|      4.0|         128.0|     263.0|        0.0|          0.0|          105.0|   1.0|    0.2|        2.0|        1.0| 7.0|
|74.0| 0.0|      2.0|         120.0|     269.0|        0.0|          2.0|          121.0|   1.0|    0.2|        1.0|        1.0| 3.0|
|65.0| 1.0|      4.0|         120.0|     177.0|        0.0|          0.0|          140.0|   0.0|    0.4|        1.0|        0.0| 7.0|
|56.0| 1.0|      3.0|         130.0|     256.0|        1.0|          2.0|          142.0|   1.0|    0.6|        2.0|        1.0| 6.0|
|59.0| 1.0|      4.0|         110.0|     239.0|        0.0|          2.0|          142.0|   1.0|    1.2|        2.0|        1.0| 7.0|
|60.0| 1.0|      4.0|         140.0|     293.0|        0.0|          2.0|          170.0|   0.0|    1.2|        2.0|        2.0| 7.0|
|63.0| 0.0|      4.0|         150.0|     407.0|        0.0|          2.0|          154.0|   0.0|    4.0|        2.0|        3.0| 7.0|
|59.0| 1.0|      4.0|         135.0|     234.0|        0.0|          0.0|          161.0|   0.0|    0.5|        2.0|        0.0| 7.0|
|53.0| 1.0|      4.0|         142.0|     226.0|        0.0|          2.0|          111.0|   1.0|    0.0|        1.0|        0.0| 7.0|
|44.0| 1.0|      3.0|         140.0|     235.0|        0.0|          2.0|          180.0|   0.0|    0.0|        1.0|        0.0| 3.0|
|61.0| 1.0|      1.0|         134.0|     234.0|        0.0|          0.0|          145.0|   0.0|    2.6|        2.0|        2.0| 3.0|
|57.0| 0.0|      4.0|         128.0|     303.0|        0.0|          2.0|          159.0|   0.0|    0.0|        1.0|        1.0| 3.0|
|71.0| 0.0|      4.0|         112.0|     149.0|        0.0|          0.0|          125.0|   0.0|    1.6|        2.0|        0.0| 3.0|
|46.0| 1.0|      4.0|         140.0|     311.0|        0.0|          0.0|          120.0|   1.0|    1.8|        2.0|        2.0| 7.0|
|53.0| 1.0|      4.0|         140.0|     203.0|        1.0|          2.0|          155.0|   1.0|    3.1|        3.0|        0.0| 7.0|
|64.0| 1.0|      1.0|         110.0|     211.0|        0.0|          2.0|          144.0|   1.0|    1.8|        2.0|        0.0| 3.0|
|40.0| 1.0|      1.0|         140.0|     199.0|        0.0|          0.0|          178.0|   1.0|    1.4|        1.0|        0.0| 7.0|
+----+----+---------+--------------+----------+-----------+-------------+---------------+------+-------+-----------+-----------+----+-
only showing top 20 rows
```

Preprocesando Data

```
data.printSchema()
```

```
root
 |-- Años: string (nullable = true)
 |-- Sexo: string (nullable = true)
 |-- TipoDolor: string (nullable = true)
 |-- PresionArterial: string (nullable = true)
 |-- Colesterol: string (nullable = true)
 |-- AzucarAyuna: string (nullable = true)
 |-- Electrocardio: string (nullable = true)
 |-- FrecuenciaCardio: string (nullable = true)
 |-- Angina: string (nullable = true)
 |-- Oldpeak: string (nullable = true)
 |-- PendienteST: string (nullable = true)
 |-- NumeroVasos: string (nullable = true)
 |-- Thal: string (nullable = true)
 |-- VariableInteres: string (nullable = true)
 |-- Enfermo: integer (nullable = false)
```

Observando las variables de la data, se encontro que los datos estaban siendo reconocidos como strings y no en datos numericos, por lo tanto habia que procesarlos nuevamente con su correspondiente tipo de variable.

```
columnas = ['Años', 'Sexo', 'TipoDolor', 'PresionArterial', 'Colesterol', 'AzucarAyuna', 'Electrocardio', 'FrecuenciaCardio', 'Angina', 'Oldp
columnasTipos = [DoubleType(), DoubleType(), DoubleType(), DoubleType(), DoubleType(), DoubleType(), DoubleType(), DoubleType(), DoubleType()


for colName, colType in zip(columnas, columnasTipos):
    data = data.withColumn(colName, data[colName].cast(colType))

data = data.withColumn('VariableInteres', data['VariableInteres'].cast(DoubleType()))
```

```
data.printSchema()
```

```
root
 |-- Años: double (nullable = true)
 |-- Sexo: double (nullable = true)
 |-- TipoDolor: double (nullable = true)
 |-- PresionArterial: double (nullable = true)
 |-- Colesterol: double (nullable = true)
 |-- AzucarAyuna: double (nullable = true)
 |-- Electrocardio: double (nullable = true)
 |-- FrecuenciaCardio: double (nullable = true)
 |-- Angina: double (nullable = true)
 |-- Oldpeak: double (nullable = true)
 |-- PendienteST: double (nullable = true)
 |-- NumeroVasos: double (nullable = true)
 |-- Thal: double (nullable = true)
 |-- VariableInteres: double (nullable = true)
 |-- Enfermo: integer (nullable = false)
```

Comprobamos que todas las variables corresponden a una variable numerica.

Assembler Data

```
#Probando sin la variable "VariableInteres", pero dio el mismo resultado
#columnas2 = ['Años', 'Sexo', 'TipoDolor', 'PresionArterial', 'Colesterol', 'AzucarAyuna', 'Electrocardio', 'FrecuenciaCardio', 'Angina', 'Ol

assembler = VectorAssembler(inputCols = columnas, outputCol='features')

output = assembler.transform(data)
modelData = output.select('features','Enfermo')
modelData.show()
```

```
+--------------------+-------+
|            features|Enfermo|
+--------------------+-------+
|[70.0,1.0,4.0,130...|      0|
|[67.0,0.0,3.0,115...|      1|
|[57.0,1.0,2.0,124...|      1|
|[64.0,1.0,4.0,128...|      1|
|[74.0,0.0,2.0,120...|      0|
|[65.0,1.0,4.0,120...|      1|
|[56.0,1.0,3.0,130...|      1|
|[59.0,1.0,4.0,110...|      1|
|[60.0,1.0,4.0,140...|      1|
|[63.0,0.0,4.0,150...|      1|
|[59.0,1.0,4.0,135...|      1|
|[53.0,1.0,4.0,142...|      1|
|[44.0,1.0,3.0,140...|      0|
|[61.0,1.0,1.0,134...|      0|
|[57.0,0.0,4.0,128...|      0|
|[71.0,0.0,4.0,112...|      0|
|[46.0,1.0,4.0,140...|      1|
|[53.0,1.0,4.0,140...|      1|
|[64.0,1.0,1.0,110...|      0|
|[40.0,1.0,1.0,140...|      1|
+--------------------+-------+
only showing top 20 rows
```

StandardScaler Data

```
scaler = StandardScaler(inputCol = 'features', outputCol= 'scaledFeatures', withMean = True, withStd= True)

scalerModel = scaler.fit(modelData)
scaledData = scalerModel.transform(modelData)
finalData = scaledData.select('scaledFeatures','Enfermo')
finalData.show(truncate=False)
```

```
+--------------------------------------------------------------------------------------------------------------------------------------------------------------
|scaledFeatures
+--------------------------------------------------------------------------------------------------------------------------------------------------------------
|[1.7089200771370505,0.6882216640697679,0.8693133244601348,-0.07527006652510361,1.3996132196232811,-0.41625583610924666,0.97984406415487
|[1.3795778781170618,-1.4476386726984776,-0.1832185033615519,-0.9150600649456696,6.081710688913948,-0.41625583610924666,0.97984406415487
|[0.2817705480504328,0.6882216640697679,-1.2357503311832385,-0.41118606589333,0.21941509719877408,-0.41625583610924666,-1.02438243070737
|[1.050235679097073,0.6882216640697679,0.8693133244601348,-0.18724206631451243,0.25811011760613495,-0.41625583610924666,-1.0243824307073
|[2.148043009163702,-1.4476386726984776,-1.2357503311832385,-0.6351300654721477,0.3741951788282176,-0.41625583610924666,0.97984406415487
|[1.160016412103736,0.6882216640697679,0.8693133244601348,-0.6351300654721477,-1.405775759910383,-0.41625583610924666,-1.024382430707371
|[0.1719898150437699,0.6882216640697679,-0.1832185033615519,-0.07527006652510361,0.12267754618037187,2.3934710576281684,0.97984406415487
|[0.5013320140637586,0.6882216640697679,0.8693133244601348,-1.1949900644191918,-0.20623012728219564,-0.41625583610924666,0.9798440641548
|[0.6111127470704215,0.6882216640697679,0.8693133244601348,0.48458993242194043,0.8385354237165482,-0.41625583610924666,0.979844064154877
|[0.9404549460904101,-1.4476386726984776,0.8693133244601348,1.0444499313689843,3.0441515869361186,-0.41625583610924666,0.979844064154877
|[0.5013320140637586,0.6882216640697679,0.8693133244601348,0.2046599329484184,-0.30296767830059784,-0.41625583610924666,-1.0243824307073
|[-0.15735238397621878,0.6882216640697679,0.8693133244601348,0.5965619322113492,-0.4577477599300414,-0.41625583610924666,0.9798440641548
|[-1.145378981036185,0.6882216640697679,-0.1832185033615519,0.48458993242194043,-0.2836201680969174,-0.41625583610924666,0.9798440641548
|[0.7208934800770844,0.6882216640697679,-2.2882821590049254,0.14867393305371399,-0.30296767830059784,-0.41625583610924666,-1.02438243070
|[0.2817705480504328,-1.4476386726984776,0.8693133244601348,-0.18724206631451243,1.0320105257533527,-0.41625583610924666,0.9798440641548
|[1.8187008101437134,-1.4476386726984776,0.8693133244601348,-1.083018064629783,-1.9475060456134354,-0.41625583610924666,-1.0243824307073
|[-0.925817515022859,0.6882216640697679,0.8693133244601348,0.48458993242194043,1.1867906073827963,-0.41625583610924666,-1.02438243070737
|[-0.15735238397621878,0.6882216640697679,0.8693133244601348,0.48458993242194043,-0.9027404946146915,2.3934710576281684,0.97984406415487
|[1.050235679097073,0.6882216640697679,-2.2882821590049254,-1.1949900644191918,-0.7479604129852481,-0.41625583610924666,0.97984406415487
|[-1.5845019130628364,0.6882216640697679,-2.2882821590049254,0.48458993242194043,-0.9801305354294133,-0.41625583610924666,-1.02438243070
+--------------------------------------------------------------------------------------------------------------------------------------------------------------
only showing top 20 rows
```

Train Test Split

```
trainData , testData = finalData.randomSplit([0.5,0.5])
```

Modelamiento Regresión Logística

```
logicClass = LogisticRegression(featuresCol = 'scaledFeatures', labelCol = 'Enfermo')

model = logicClass.fit(trainData)


pred = model.transform(testData)
```

Metricas AUC y Precisión

```
evaluator = BinaryClassificationEvaluator(rawPredictionCol='prediction',labelCol = 'Enfermo')
AUC = evaluator.evaluate(pred)
print("AUC-ROC:", AUC)

accuracy = MulticlassClassificationEvaluator(labelCol="Enfermo", metricName="accuracy")
accuracy = accuracy.evaluate(pred)
print("Exactitud:", accuracy)

recall = MulticlassClassificationEvaluator(labelCol="Enfermo", metricName="recallByLabel")
recall = recall.evaluate(pred)
print("Recall:", recall)

f1 = MulticlassClassificationEvaluator(labelCol="Enfermo", metricName="f1")
f1Score = f1.evaluate(pred)
print("F1-Score:", f1Score)
```

```
AUC-ROC: 1.0
Exactitud: 1.0
Recall: 1.0
F1-Score: 1.0
```

Segun las metricas obtenidas, el obtener un "1.0" como resultado indica que el modelo tiene un rendimiento perfecto y que todas sus predicciones los esta clasificando correctamente.

```
# pred.select('Enfermo','prediction').show()
pred.show()
```

```
+--------------------+-------+--------------------+--------------------+----------+
|      scaledFeatures|Enfermo|       rawPrediction|         probability|prediction|
+--------------------+-------+--------------------+--------------------+----------+
|[-2.2431863111028...|      0|[19.7274290219951...|[0.99999999729301...|       0.0|
|[-2.1334055780961...|      0|[20.3120875148059...|[0.99999999849140...|       0.0|
|[-2.1334055780961...|      1|[-25.172143898952...|[1.16916769511768...|       1.0|
|[-1.9138441120828...|      0|[21.4560037986716...|[0.99999999951940...|       0.0|
|[-1.9138441120828...|      0|[15.4570281103857...|[0.99999980631414...|       0.0|
|[-1.8040633790761...|      1|[-26.379777685277...|[3.49468565622738...|       1.0|
|[-1.6942826460694...|      0|[22.6160868120582...|[0.99999999984935...|       0.0|
|[-1.5845019130628...|      1|[-26.484884640965...|[3.14601469287166...|       1.0|
|[-1.4747211800561...|      0|[21.8130480123959...|[0.99999999966371...|       0.0|
|[-1.4747211800561...|      0|[19.9805527583414...|[0.99999999789837...|       0.0|
|[-1.4747211800561...|      0|[20.4196818805424...|[0.99999999864529...|       0.0|
|[-1.4747211800561...|      0|[17.7024520484382...|[0.99999997949202...|       0.0|
|[-1.3649404470495...|      0|[20.4932979045916...|[0.99999999874144...|       0.0|
|[-1.3649404470495...|      0|[18.7931952854680...|[0.99999999311000...|       0.0|
|[-1.2551597140428...|      0|[19.5513892058550...|[0.99999999677195...|       0.0|
|[-1.2551597140428...|      1|[-24.573100147960...|[2.12832625479817...|       1.0|
|[-1.2551597140428...|      0|[18.8820517026223...|[0.99999999369581...|       0.0|
|[-1.1453789810361...|      0|[19.4357354767141...|[0.99999999637617...|       0.0|
|[-1.1453789810361...|      0|[19.2670637648659...|[0.99999999571036...|       0.0|
|[-1.1453789810361...|      1|[-23.688431043673...|[5.15519518096946...|       1.0|
+--------------------+-------+--------------------+--------------------+----------+
```

```
only showing top 20 rows
```

Comprobando los datos obtenidos en la prediccion con los datos de testeo, encontramos que los datos con una rawPrediction con un valor negativo los clasifica como 1 ("Enfermos") y los positivos en 0 ("Sanos").

Corroborando Prediccion vs Enfermo

```
predData = pred.withColumn('correcta', F.when(F.col('Enfermo') == F.col('prediction'), True).otherwise(False))
predData.select('Enfermo','prediction','correcta').show()
```

```
+-------+----------+--------+
|Enfermo|prediction|correcta|
+-------+----------+--------+
|      0|       0.0|    true|
|      0|       0.0|    true|
|      1|       1.0|    true|
|      0|       0.0|    true|
|      0|       0.0|    true|
|      1|       1.0|    true|
|      0|       0.0|    true|
|      1|       1.0|    true|
|      0|       0.0|    true|
|      0|       0.0|    true|
|      0|       0.0|    true|
|      0|       0.0|    true|
|      0|       0.0|    true|
|      0|       0.0|    true|
|      0|       0.0|    true|
|      1|       1.0|    true|
|      0|       0.0|    true|
|      0|       0.0|    true|
|      0|       0.0|    true|
|      1|       1.0|    true|
+-------+----------+--------+
only showing top 20 rows
```

Corroboramos la prediccion realizada contra la variable Enfermo si corresponden efectivamente. Se observa en la variable 'Correcta' si ambos datos son iguales se da un True y sino es un False. En los datos que se observan todos corresponden.