Tarea 3 - Big Data

Estudiante: Steven Jimenez

Datos de entrada

Dataset: Calidad del Vino

El archivo winequality-red.csv contiene métricas sobre la calidad de Vinos Rojos. La intención con este dataset es predecir si un vino es de Alta Calidad o Baja Calidad, basado en el análisis de distintos features. Explícitamente:

Features

- "fixed_acidity": la mayoría de los ácidos involucrados con el vino o fijos o no volátiles (no se evaporan fácilmente)
- "volatile_acidity":la cantidad de ácido acético en el vino, que en niveles demasiado altos puede provocar un sabor desagradable a vinagre
- "citric_acid":Encontrado en pequeñas cantidades, el ácido cítrico puede agregar 'frescura' y sabor a los vinos.
- "residual_sugar": la cantidad de azúcar que queda después de que se detiene la fermentación, es raro encontrar vinos con menos de 1 gramo / litro y los vinos con más de 45 gramos / litro se consideran dulces
- "chlorides": la cantidad de sal en el vino
- "free_sulfur_dioxide": la forma libre del SO2 existe en equilibrio entre el SO2 molecular (como gas disuelto) y el ion bisulfito; Previene el crecimiento microbiano y la oxidación del vino.
- "total_sulfur_dioxide":cantidad de formas libres y ligadas de SO2; en bajas concentraciones, el SO2 es mayormente indetectable en el vino, pero en concentraciones de SO2 libre superiores a 50 ppm, el SO2 se hace evidente en la nariz y el sabor del vino.
- "density": la densidad del agua es cercana a la del agua dependiendo del porcentaje de alcohol y contenido de azúcar
- "pH": describe qué tan ácido o básico es un vino en una escala de 0 (muy ácido) a 14 (muy básico); la mayoría de los vinos están entre 3-4 en la escala de pH
- "sulphates": un aditivo para el vino que puede contribuir a los niveles de dióxido de azufre (SO2), que actúa como antimicrobiano y antioxidante
- "alcohol": el porcentaje de contenido de alcohol del vino
- "quality": variable de salida (basada en datos sensoriales, puntuación entre 0 y 10)

Variable de predicción (Variable sintetica)

Variable sintética, creada para cumplir el requisito de que sea un problema de clasificación binaria.

- BinaryQuality: Si el parámetro "quality" es mayor a 5, se catalogará al vino como 1 (Refiriendose a Vino de Alta Calidad), de lo contrario se catalogará como 0 (Refiriendose a Vino de Baja Calidad)

Dataset source:https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009

Preprocesamiento de datos

Carga/lectura de datos (.csv) Limpieza

Definición del "schema" y lectura del archivo .csv

In [55]:

```
# Cargar el conjunto de datos completo. Este paso no realiza ningún ajuste; simplemente lectura
import findspark
from pyspark.sql.functions import isnan, when, count, col
import pandas as pd
findspark.init('/usr/lib/python3.7/site-packages/pyspark')

from pyspark.sql.types import (StringType, IntegerType, FloatType,
                               DecimalType, StructField, StructType)

from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .appName("Basic JDBC pipeline") \
    .config("spark.driver.extraClassPath", "postgresql-42.2.14.jar") \
    .config("spark.executor.extraClassPath", "postgresql-42.2.14.jar") \
    .getOrCreate()

RawWine_df = spark \
    .read \
    .format("csv") \
    .option("path", "winequality-red.csv") \
    .option("header", True) \
    .schema(StructType([
                StructField("fixed_acidity",FloatType()),
                StructField("volatile_acidity",FloatType()),
                StructField("citric_acid",FloatType()),
                StructField("residual_sugar",FloatType()),
                StructField("chlorides",FloatType()),
                StructField("free_sulfur_dioxide",FloatType()),
                StructField("total_sulfur_dioxide",FloatType()),
                StructField("density",FloatType()),
                StructField("pH",FloatType()),
```

```
                StructField("sulphates",FloatType()),
                StructField("alcohol",FloatType()),
                StructField("quality",IntegerType())])) \
    .load()
print('Qty Filas: {}\n Cantidad Columnas: {}'.format(RawWine_df.count(), len(RawWine_df.columns)))
RawWine_df.printSchema()
RawWine_df.show(truncate=False,n=3)
```

```
Qty Filas: 1599
 Cantidad Columnas: 12
root
 |-- fixed_acidity: float (nullable = true)
 |-- volatile_acidity: float (nullable = true)
 |-- citric_acid: float (nullable = true)
 |-- residual_sugar: float (nullable = true)
 |-- chlorides: float (nullable = true)
 |-- free_sulfur_dioxide: float (nullable = true)
 |-- total_sulfur_dioxide: float (nullable = true)
 |-- density: float (nullable = true)
 |-- pH: float (nullable = true)
 |-- sulphates: float (nullable = true)
 |-- alcohol: float (nullable = true)
 |-- quality: integer (nullable = true)

+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+
-------+
|fixed_acidity|volatile_acidity|citric_acid|residual_sugar|chlorides|free_sulfur_dioxide|total_sulfur_dioxide|density|pH  |sulphates|alcohol|
quality|
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+
-------+
|7.4          |0.7             |0.0        |1.9           |0.076    |11.0               |34.0                |0.9978 |3.51|0.56     |9.4    |
5      |
|7.8          |0.88            |0.0        |2.6           |0.098    |25.0               |67.0                |0.9968 |3.2 |0.68     |9.8    |
5      |
|7.8          |0.76            |0.04       |2.3           |0.092    |15.0               |54.0                |0.997  |3.26|0.65     |9.8    |
5      |
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+
-------+
only showing top 3 rows

21/11/04 04:23:30 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sul
phates, alcohol, quality
 Schema: fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH, sul
phates, alcohol, quality
Expected: fixed_acidity but found: fixed acidity
CSV file: file:///src/tarea3Jupyter/winequality-red.csv
```

Creación de variable sintética - 'BinaryQuality'

In [56]:

```
from pyspark.sql import functions as F
threshold = 5
WineQualityDF = RawWine_df.withColumn('BinaryQuality', F.when(F.col("quality") <= threshold, 0)\
                                                      .when(F.col("quality") > threshold, 1))
WineQualityDF.show(n=5)
```

```
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+
-------+-------------+
|fixed_acidity|volatile_acidity|citric_acid|residual_sugar|chlorides|free_sulfur_dioxide|total_sulfur_dioxide|density|  pH|sulphates|alcohol|
quality|BinaryQuality|
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+
-------+-------------+
|          7.4|             0.7|        0.0|           1.9|    0.076|               11.0|                34.0| 0.9978|3.51|     0.56|    9.4|
5|            0|
|          7.8|            0.88|        0.0|           2.6|    0.098|               25.0|                67.0| 0.9968| 3.2|     0.68|    9.8|
5|            0|
|          7.8|            0.76|       0.04|           2.3|    0.092|               15.0|                54.0|  0.997|3.26|     0.65|    9.8|
5|            0|
|         11.2|            0.28|       0.56|           1.9|    0.075|               17.0|                60.0|  0.998|3.16|     0.58|    9.8|
6|            1|
|          7.4|             0.7|        0.0|           1.9|    0.076|               11.0|                34.0| 0.9978|3.51|     0.56|    9.4|
5|            0|
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+
-------+-------------+
only showing top 5 rows

21/11/04 04:23:30 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sul
phates, alcohol, quality
 Schema: fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH, sul
phates, alcohol, quality
Expected: fixed_acidity but found: fixed acidity
CSV file: file:///src/tarea3Jupyter/winequality-red.csv
```

En el siguiente paso eliminamos la variable Quality para realizar el proceso de clasificación en base a nuestra vartiable sintética llamada "BinaryQuality"

In [57]:

```
WineQualityDF = WineQualityDF.drop("quality")
WineQualityDF.show(n=5)
```

```
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+
-------------+
|fixed_acidity|volatile_acidity|citric_acid|residual_sugar|chlorides|free_sulfur_dioxide|total_sulfur_dioxide|density|  pH|sulphates|alcohol|
BinaryQuality|
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+
```

```
-------------+
|         7.4|             0.7|        0.0|           1.9|    0.076|               11.0|                34.0| 0.9978|3.51|     0.56|    9.4|
0|
|         7.8|            0.88|        0.0|           2.6|    0.098|               25.0|                67.0| 0.9968| 3.2|     0.68|    9.8|
0|
|         7.8|            0.76|       0.04|           2.3|    0.092|               15.0|                54.0|  0.997|3.26|     0.65|    9.8|
0|
|        11.2|            0.28|       0.56|           1.9|    0.075|               17.0|                60.0|  0.998|3.16|     0.58|    9.8|
1|
|         7.4|             0.7|        0.0|           1.9|    0.076|               11.0|                34.0| 0.9978|3.51|     0.56|    9.4|
0|
+------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+
-------------+
only showing top 5 rows

21/11/04 04:23:31 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sul
phates, alcohol, quality
 Schema: fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH, sul
phates, alcohol, quality
Expected: fixed_acidity but found: fixed acidity
CSV file: file:///src/tarea3Jupyter/winequality-red.csv
```

Limpieza de datos

Busquedas de valores "NaN": Not a Number: No se identifican valores NaN

In [58]:

```
WineQualityDF.select([count(when(isnan(c), c)).alias(c) for c in WineQualityDF.columns]).show()
```

```
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+---+---------+-------+-
------------+
|fixed_acidity|volatile_acidity|citric_acid|residual_sugar|chlorides|free_sulfur_dioxide|total_sulfur_dioxide|density| pH|sulphates|alcohol|B
inaryQuality|
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+---+---------+-------+-
------------+
|            0|               0|          0|             0|        0|                  0|                   0|      0|  0|        0|      0|
0|
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+---+---------+-------+-
------------+

21/11/04 04:23:31 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sul
phates, alcohol, quality
 Schema: fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH, sul
phates, alcohol, quality
Expected: fixed_acidity but found: fixed acidity
CSV file: file:///src/tarea3Jupyter/winequality-red.csv
```

Busqueda de valores "Null": Se identifican varios valores Null

In [59]:

```
WineQualityDF.select([count(when(col(c).isNull(), c)).alias(c) for c in WineQualityDF.columns]).show()
```

```
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+---+---------+-------+-
------------+
|fixed_acidity|volatile_acidity|citric_acid|residual_sugar|chlorides|free_sulfur_dioxide|total_sulfur_dioxide|density| pH|sulphates|alcohol|B
inaryQuality|
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+---+---------+-------+-
------------+
|            0|               0|          0|             0|        0|                  0|                   0|      0|  0|        0|      0|
0|
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+---+---------+-------+-
------------+

21/11/04 04:23:31 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sul
phates, alcohol, quality
 Schema: fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH, sul
phates, alcohol, quality
Expected: fixed_acidity but found: fixed acidity
CSV file: file:///src/tarea3Jupyter/winequality-red.csv
```

No se encontraron datos NaN, tampoco Nulls, por lo que no fue necesario limpiar los datos. (Para términos de este ejercicio omitiremos la presencia de outliers, puesto que no tengo el conocimiento técnico para mantener/descartar en caso de que existiera un outlier)

Gráficos y estadísdisticas descriptivas

Estadísticas descriptivas - Features y Variable a predecir

In [60]:

```
WineQualityDF.describe(WineQualityDF.schema.names[0:4]).show()
```

```
21/11/04 04:23:31 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: fixed acidity, volatile acidity, citric acid, residual sugar
 Schema: fixed_acidity, volatile_acidity, citric_acid, residual_sugar
Expected: fixed_acidity but found: fixed acidity
CSV file: file:///src/tarea3Jupyter/winequality-red.csv
+-------+------------------+-------------------+-------------------+------------------+
|summary|     fixed_acidity|   volatile_acidity|        citric_acid|    residual_sugar|
+-------+------------------+-------------------+-------------------+------------------+
|  count|              1599|               1599|               1599|              1599|
|   mean|  8.31963727204333| 0.5278205118742565|0.27097560946082344|2.5388054955072743|
| stddev|1.7410963179910275|0.17905970357107073|0.19480113735645493|1.4099280590834145|
```

```
|    min|             4.6|              0.12|               0.0|               0.9|
|    max|            15.9|              1.58|               1.0|              15.5|
+-------+------------------+-------------------+-------------------+------------------+
```

In [61]:

```
WineQualityDF.describe(WineQualityDF.schema.names[4:8]).show()
```

```
21/11/04 04:23:31 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: chlorides, free sulfur dioxide, total sulfur dioxide, density
 Schema: chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density
Expected: free_sulfur_dioxide but found: free sulfur dioxide
CSV file: file:///src/tarea3Jupyter/winequality-red.csv
+-------+--------------------+-------------------+--------------------+--------------------+
|summary|           chlorides|free_sulfur_dioxide|total_sulfur_dioxide|             density|
+-------+--------------------+-------------------+--------------------+--------------------+
|  count|                1599|               1599|                1599|                1599|
|   mean|0.08746654185244558| 15.874921826141339|   46.46779237023139| 0.9967466800044371|
| stddev|0.04706530186883863|  10.46015696980971|   32.89532447829907|0.001887335252223...|
|    min|               0.012|                1.0|                 6.0|             0.99007|
|    max|               0.611|               72.0|               289.0|             1.00369|
+-------+--------------------+-------------------+--------------------+--------------------+
```

In [62]:

```
WineQualityDF.describe(WineQualityDF.schema.names[8:11]).show()
```

```
+-------+-------------------+-------------------+------------------+
|summary|                 pH|          sulphates|           alcohol|
+-------+-------------------+-------------------+------------------+
|  count|               1599|               1599|              1599|
|   mean| 3.3111131965107585| 0.6581488421292809|10.422983095003262|
| stddev|0.15438646318792024|0.16950698014899196|1.0656675859276652|
|    min|               2.74|               0.33|               8.4|
|    max|               4.01|                2.0|              14.9|
+-------+-------------------+-------------------+------------------+
```

In [63]:

```
WineQualityDF.groupBy("BinaryQuality").count().show(truncate=False)
```

```
+-------------+-----+
|BinaryQuality|count|
+-------------+-----+
|1            |855  |
|0            |744  |
+-------------+-----+
```

Matriz correlación Pearson y Vectorización

In [64]:

```
from pyspark.ml.feature import VectorAssembler

assembler = VectorAssembler(
    inputCols=WineQualityDF.drop("BinaryQuality").schema.names,
    outputCol='features')

vector_df = assembler.transform(WineQualityDF)
vector_df = vector_df.select(['features', "BinaryQuality"])
vector_df.show(n=15)
```

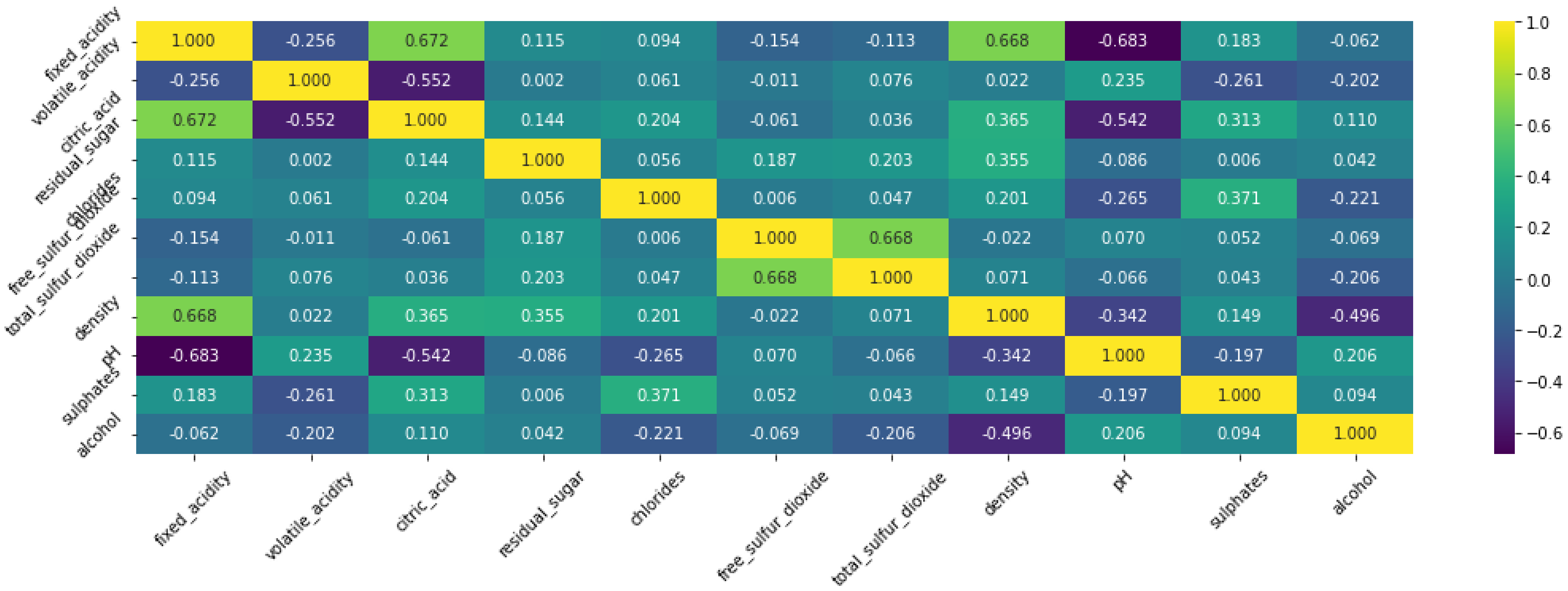
```
+--------------------+-------------+
|            features|BinaryQuality|
+--------------------+-------------+
|[7.40000009536743...|            0|
|[7.80000019073486...|            0|
|[7.80000019073486...|            0|
|[11.1999998092651...|            1|
|[7.40000009536743...|            0|
|[7.40000009536743...|            0|
|[7.90000009536743...|            0|
|[7.30000019073486...|            1|
|[7.80000019073486...|            1|
|[7.5,0.5,0.360000...|            0|
|[6.69999980926513...|            0|
|[7.5,0.5,0.360000...|            0|
|[5.59999990463256...|            0|
|[7.80000019073486...|            0|
|[8.89999961853027...|            0|
+--------------------+-------------+
only showing top 15 rows

21/11/04 04:23:32 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sul
phates, alcohol, quality
 Schema: fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH, sul
phates, alcohol, quality
Expected: fixed_acidity but found: fixed acidity
CSV file: file:///src/tarea3Jupyter/winequality-red.csv
```

In [65]:

```
from pyspark.ml.stat import Correlation
import seaborn as sns
import matplotlib.pyplot as plt

pearson_matrix = Correlation.corr(vector_df, 'features').collect()[0][0]
fig, ax = plt.subplots(figsize=(18,5))
```

```
sns.heatmap(pearson_matrix.toArray(), annot=True, fmt=".3f", cmap='viridis', ax=ax)
ax.set_xticklabels(WineQualityDF.drop("BinaryQuality").schema.names)
ax.set_yticklabels(WineQualityDF.drop("BinaryQuality").schema.names)
ax.tick_params(labelrotation=45)
```

```
21/11/04 04:23:32 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sul
phates, alcohol
 Schema: fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH, sul
phates, alcohol
Expected: fixed_acidity but found: fixed acidity
CSV file: file:///src/tarea3Jupyter/winequality-red.csv
21/11/04 04:23:32 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sul
phates, alcohol
 Schema: fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH, sul
phates, alcohol
Expected: fixed_acidity but found: fixed acidity
CSV file: file:///src/tarea3Jupyter/winequality-red.csv
21/11/04 04:23:32 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sul
phates, alcohol
 Schema: fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH, sul
phates, alcohol
Expected: fixed_acidity but found: fixed acidity
CSV file: file:///src/tarea3Jupyter/winequality-red.csv
21/11/04 04:23:32 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sul
phates, alcohol
 Schema: fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH, sul
phates, alcohol
Expected: fixed_acidity but found: fixed acidity
CSV file: file:///src/tarea3Jupyter/winequality-red.csv
```



De esta matriz se observa que si existe variable con una fuerte correlación tanto positiva ("Density" vs "free_sulfur_dioxide") y como correlación negativa ("pH" vs "fixed_acidity")

Normalización de Datos

Se utilizó el método de normalización por la siguientes razones:

- Para un conjunto de datos con múltiples Features que abarcan diversos grados de magnitud, rango y unidades, el método de normalización "mapea" los features a distancias ahora son más comparables de lo que eran antes de que se aplicara la normalización.
- La estandarización no tiene un rango límite, por lo que si el dataset tiene valores atípicos en sus datos, no se verán afectados por la estandarización.

In [66]:

```
from pyspark.ml.feature import StandardScaler

standard_scaler = StandardScaler(inputCol='features', outputCol='scaledFeatures')
scale_model = standard_scaler.fit(vector_df)

scaled_df = scale_model.transform(vector_df)
scaled_df.show()
```

```
+--------------------+-------------+--------------------+
|            features|BinaryQuality|      scaledFeatures|
+--------------------+-------------+--------------------+
|[7.40000009536743...|            0|[4.25019570652240...|
|[7.80000019073486...|            0|[4.47993606679665...|
|[7.80000019073486...|            0|[4.47993606679665...|
|[11.1999998092651...|            1|[6.43272844444832...|
|[7.40000009536743...|            0|[4.25019570652240...|
|[7.40000009536743...|            0|[4.25019570652240...|
|[7.90000009536743...|            0|[4.53737108839727...|
|[7.30000019073486...|            1|[4.19276068492178...|
|[7.80000019073486...|            1|[4.47993606679665...|
|[7.5,0.5,0.360000...|            0|[4.30763072812302...|
|[6.69999980926513...|            0|[3.84815000757451...|
|[7.5,0.5,0.360000...|            0|[4.30763072812302...|
```

```
|[5.59999990463256...|           0|[3.21636422222416...|
|[7.80000019073486...|           0|[4.47993606679665...|
|[8.89999961853027...|           0|[5.11172157827522...|
|[8.89999961853027...|           0|[5.11172157827522...|
|[8.5,0.2800000011...|           1|[4.88198149187275...|
|[8.10000038146972...|           0|[4.65224140547029...|
|[7.40000009536743...|           0|[4.25019570652240...|
|[7.90000009536743...|           1|[4.53737108839727...|
+--------------------+------------+--------------------+
only showing top 20 rows

21/11/04 04:23:33 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sul
phates, alcohol
 Schema: fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH, sul
phates, alcohol
Expected: fixed_acidity but found: fixed acidity
CSV file: file:///src/tarea3Jupyter/winequality-red.csv
21/11/04 04:23:33 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sul
phates, alcohol, quality
 Schema: fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH, sul
phates, alcohol, quality
Expected: fixed_acidity but found: fixed acidity
CSV file: file:///src/tarea3Jupyter/winequality-red.csv
```

In [67]:

```python
cleaned_df = scaled_df.select("scaledFeatures","BinaryQuality")
cleaned_df.show(truncate = False)
cleaned_df.printSchema()
```

```
+-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------+-------------+
|scaledFeatures
|BinaryQuality|
+-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------+-------------+
|[4.250195706522404,3.909310549043959,0.0,1.3475864700453726,1.6147776501679734,1.0516094578454602,1.033581535954439,528.681902902932,22.7351
53833991657,3.3036987733010235,8.820761504486935]                |0            |
|[4.479936066796656,4.914561890148259,0.0,1.84406565135161,2.082213298240522,2.3900214951033187,2.036763614969041,528.1520621530078,20.727206
139754976,4.0116342498394815,9.19611361004656]                   |0            |
|[4.479936066796656,4.244394329412059,0.2053376055640616,1.631288871441781,1.9547309063309342,1.4340128970619912,1.6415706747511676,528.25801
76704421,21.115840878452943,3.834650204887197,9.19611361004656]   |0            |
|[6.432728444448329,1.5637242529051651,2.874726554391083,1.3475864700453726,1.5935306903850721,1.6252146166702566,1.8239674163901862,528.7878
900017428,20.46811631395633,3.4216879021789866,9.19611361004656]  |1            |
|[4.250195706522404,3.909310549043959,0.0,1.3475864700453726,1.6147776501679734,1.0516094578454602,1.033581535954439,528.681902902932,22.7351
53833991657,3.3036987733010235,8.820761504486935]                |0            |
|[4.250195706522404,3.685921584049102,0.0,1.2766608485587931,1.5935306903850721,1.2428111774537256,1.2159782775934573,528.681902902932,22.735
153833991657,3.3036987733010235,8.820761504486935]               |0            |
|[4.537371088397272,3.350837803681002,0.30800640834609244,1.1348096901355436,1.4660481401724392,1.4340128970619912,1.7935679594503497,527.940
1195367628,21.37493070425159,2.7137526014581987,8.820761504486935]|0            |
|[4.1927606849217876,3.6300740099245727,0.0,0.851107288739135,1.381059826131699,1.4340128970619912,0.6383885957365651,526.9863935543485,21.95
788435659572,2.7727471658971803,9.383789215372433]               |1            |
|[4.479936066796656,3.2391429883077585,0.1026688027820308,1.418512091531952,1.5510364542131796,0.8604077382371946,0.5471902249170558,528.1520
621530078,21.76356544294956,3.362693337740005,8.914599754603811]  |1            |
|[4.30763072812302,2.79236472544223,1.8480385648178856,4.326461811532526,1.508542376344332,1.6252146166702566,3.1007446078633163,528.68190290
2932,21.698792986499896,4.71956972637794,9.852978676141054]       |0            |
|[3.848150007574517,3.2391429883077585,0.4106752111281232,1.2766608485587931,2.0609663384576207,1.4340128970619912,1.9759647010893684,527.675
1833711124,21.245385791352266,3.185709644423061,8.633085899161062]|0            |
|[4.30763072812302,2.79236472544223,1.8480385648178856,4.326461811532526,1.508542376344332,1.6252146166702566,3.1007446078633163,528.68190290
2932,21.698792986499896,4.71956972637794,9.852978676141054]       |0            |
|[3.2163642222241644,3.434608665554073,0.0,1.1348096901355436,1.8909897103761402,1.5296137568661239,1.7935679594503497,526.8274444875088,23.1
8856102913929,3.0677201639097573,9.289950965255557]              |0            |
|[4.479936066796656,3.406685044929716,1.4886976307776691,1.1348096901355436,2.4221665544034825,0.8604077382371946,0.8815842512552566,528.4699
602866871,21.115840878452943,9.203160491729273,8.539248543952064] |0            |
|[5.111721578275222,3.4625322861784302,0.9240192824089428,2.695172940090745,3.7394851845803876,4.971244709814902,4.407921256276283,529.105788
135422,20.46811631395633,5.191526593525132,8.633085899161062]     |0            |
|[5.111721578275222,3.4625322861784302,0.9753536359910703,2.766098646127234,3.6120027926708,4.8756438500107695,4.4991196270957925,529.1057881
35422,20.53288877040599,5.486499767355379,8.633085899161062]      |0            |
|[4.881981491872756,1.5637242529051651,2.874726554391083,1.2766608485587931,1.9547309063309342,3.346030093144646,3.131144064803153,528.205055
7024131,21.37493070425159,4.424596552547692,9.852978676141054]    |1            |
|[4.65224140547029,3.1274485058103303,1.4373632771955416,1.205735311622123,7.818923625323737,1.5296137568661239,1.7023695886308405,528.152062
1530078,20.144252487410846,7.551311280896431,8.726924149277938]   |0            |
|[4.250195706522404,3.294990229556473,0.4106752111281232,3.1207266690102218,1.8272485144213462,0.5736051588247965,0.8815842512552566,528.4699
602866871,21.893111900146057,2.9497310350317947,8.44541029383519] |0            |
|[4.537371088397272,1.7871133843379294,2.618054480503563,1.2766608485587931,7.2452523868214564,1.6252146166702566,1.7023695886308405,528.2050
557024131,19.690845292263216,6.371419288846122,8.633085899161062] |1            |
+-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------+-------------+
only showing top 20 rows

root
 |-- scaledFeatures: vector (nullable = true)
 |-- BinaryQuality: integer (nullable = true)

21/11/04 04:23:34 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sul
phates, alcohol, quality
 Schema: fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH, sul
phates, alcohol, quality
Expected: fixed_acidity but found: fixed acidity
CSV file: file:///src/tarea3Jupyter/winequality-red.csv
```

Escritura a la base de datos

Regeneramos los nombres de las columnas de cada feature del dataset para escribir a la base de datos

In [68]:
```
from pyspark.ml.functions import vector_to_array

features_col_names = WineQualityDF.drop("BinaryQuality").schema.names


ExpandedDFtoDB = (cleaned_df.withColumn("xs", vector_to_array("scaledFeatures")))\
    .select([col("xs")[i].alias(features_col_names[i]) for i in range(11)]+["BinaryQuality"])


ExpandedDFtoDB.show()
```

```
21/11/04 04:23:34 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sul
phates, alcohol, quality
 Schema: fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH, sul
phates, alcohol, quality
Expected: fixed_acidity but found: fixed acidity
CSV file: file:///src/tarea3Jupyter/winequality-red.csv
+------------------+------------------+-------------------+------------------+------------------+-------------------+--------------------+---
-------------+-----------------+-----------------+------------------+-------------+
|     fixed_acidity|  volatile_acidity|        citric_acid|    residual_sugar|         chlorides|free_sulfur_dioxide|total_sulfur_dioxide|
density|                pH|         sulphates|          alcohol|BinaryQuality|
+------------------+------------------+-------------------+------------------+------------------+-------------------+--------------------+---
-------------+-----------------+-----------------+------------------+-------------+
| 4.250195706522404| 3.909310549043959|                0.0|1.3475864700453726|1.6147776501679734| 1.0516094578454602|   1.033581535954439| 52
8.681902902932|22.735153833991657|3.3036987733010235|8.820761504486935|            0|
| 4.479936066796656| 4.914561890148259|                0.0|  1.84406565135161| 2.082213298240522| 2.3900214951033187|   2.036763614969041|52
8.1520621530078|20.727206139754976|4.0116342498394815| 9.19611361004656|            0|
| 4.479936066796656| 4.244394329412059| 0.2053376055640616| 1.631288871441781|1.9547309063309342| 1.4340128970619912|   1.6415706747511676|52
8.2580176704421|21.115840878452943| 3.834650204887197| 9.19611361004656|            0|
| 6.432728444448329|1.5637242529051651|  2.874726554391083|1.3475864700453726|1.5935306903850721| 1.6252146166702566|   1.8239674163901862|52
8.7878900017428| 20.46811631395633|3.4216879021789866| 9.19611361004656|            1|
| 4.250195706522404| 3.909310549043959|                0.0|1.3475864700453726|1.6147776501679734| 1.0516094578454602|   1.033581535954439| 52
8.681902902932|22.735153833991657|3.3036987733010235|8.820761504486935|            0|
| 4.250195706522404| 3.685921584049102|                0.0|1.2766608485587931|1.5935306903850721| 1.2428111774537256|   1.2159782775934573| 52
8.681902902932|22.735153833991657|3.3036987733010235|8.820761504486935|            0|
| 4.537371088397272| 3.350837803681002|0.30800640834609244|1.1348096901355436|1.4660481401724392| 1.4340128970619912|   1.7935679594503497|52
7.9401195367628| 21.37493070425159|2.7137526014581987|8.820761504486935|            0|
|4.1927606849217876|3.6300740099245727|                0.0| 0.851107288739135| 1.381059826131699| 1.4340128970619912|   0.6383885957365651|52
6.9863935543485| 21.95788435659572|2.7727471658971803|9.383789215372433|            1|
| 4.479936066796656|3.2391429883077585| 0.1026688027820308| 1.418512091531952|1.5510364542131796| 0.8604077382371946|   0.5471902249170558|52
8.1520621530078| 21.76356544294956| 3.362693337740005|8.914599754603811|            1|
|  4.30763072812302|  2.79236472544223| 1.8480385648178856| 4.326461811532526| 1.508542376344332| 1.6252146166702566|   3.1007446078633163| 52
8.681902902932|21.698792986499896|  4.71956972637794|9.852978676141054|            0|
| 3.848150007574517|3.2391429883077585| 0.4106752111281232|1.2766608485587931|2.0609663384576207| 1.4340128970619912|   1.9759647010893684|52
7.6751833711124|21.245385791352266| 3.185709644423061|8.633085899161062|            0|
|  4.30763072812302|  2.79236472544223| 1.8480385648178856| 4.326461811532526| 1.508542376344332| 1.6252146166702566|   3.1007446078633163| 52
8.681902902932|21.698792986499896|  4.71956972637794|9.852978676141054|            0|
|3.2163642222241644| 3.434608665554073|                0.0|1.1348096901355436|1.8909897103761402| 1.5296137568661239|   1.7935679594503497|52
6.8274444875088| 23.18856102913929|3.0677201639097573|9.289950965255557|            0|
| 4.479936066796656| 3.406685044929716| 1.4886976307776691|1.1348096901355436|2.4221665544034825| 0.8604077382371946|   0.8815842512552566|52
8.4699602866871|21.115840878452943| 9.203160491729273|8.539248543952064|            0|
| 5.111721578275222|3.4625322861784302| 0.9240192824089428| 2.695172940090745|3.7394851845803876|  4.971244709814902|   4.407921256276283| 52
9.105788135422| 20.46811631395633| 5.191526593525132|8.633085899161062|            0|
| 5.111721578275222|3.4625322861784302| 0.9753536359910703| 2.766098646127234|   3.6120027926708| 4.8756438500107695|   4.4991196270957925| 52
9.105788135422|  20.53288877040599| 5.486499767355379|8.633085899161062|            0|
| 4.881981491872756|1.5637242529051651|  2.874726554391083|1.2766608485587931|1.9547309063309342|  3.346030093144646|    3.131144064803153|52
8.2050557024131| 21.37493070425159| 4.424596552547692|9.852978676141054|            1|
|  4.65224140547029|3.1274485058103303| 1.4373632771955416| 1.205735311622123| 7.818923625323737| 1.5296137568661239|   1.7023695886308405|52
8.1520621530078|20.144252487410846| 7.551311280896431|8.726924149277938|            0|
| 4.250195706522404| 3.294990229556473| 0.4106752111281232|3.1207266690102218|1.8272485144213462| 0.5736051588247965|   0.8815842512552566|52
8.4699602866871|21.893111900146057|2.9497310350317947| 8.44541029383519|            0|
| 4.537371088397272|1.7871133843379294|  2.618054480503563|1.2766608485587931|7.2452523868214564| 1.6252146166702566|   1.7023695886308405|52
8.2050557024131|19.690845292263216| 6.371419288846122|8.633085899161062|            1|
+------------------+------------------+-------------------+------------------+------------------+-------------------+--------------------+---
-------------+-----------------+-----------------+------------------+-------------+
only showing top 20 rows
```

Escribimos con overwrite la tabla **tarea3** a la base de datos

In [69]:
```
# Almacenar el conjunto de datos limpio en la base de datos
ExpandedDFtoDB \
    .write \
    .format("jdbc") \
    .mode('overwrite') \
    .option("url", "jdbc:postgresql://host.docker.internal:5433/postgres") \
    .option("user", "postgres") \
    .option("password", "testPassword") \
    .option("dbtable", "tarea3") \
    .save()
```

```
21/11/04 04:23:34 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sul
phates, alcohol, quality
 Schema: fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH, sul
phates, alcohol, quality
Expected: fixed_acidity but found: fixed acidity
CSV file: file:///src/tarea3Jupyter/winequality-red.csv
```

Entrenamiento de modelos

Lectura desde la base de datos

Leemos la tabla **tarea3** desde la base de datos

In [70]:

```python
# Cargar el conjunto de datos. Esta vez desde la base de datos

# Reading single DataFrame in Spark by retrieving all rows from a DB table.
df = spark \
    .read \
    .format("jdbc") \
    .option("url", "jdbc:postgresql://host.docker.internal:5433/postgres") \
    .option("user", "postgres") \
    .option("password", "testPassword") \
    .option("dbtable", "tarea3") \
    .load()

df.show(n=3)
```

```
+-----------------+-----------------+------------------+------------------+------------------+-------------------+--------------------+------
-----------+------------------+------------------+-----------------+-------------+
|    fixed_acidity| volatile_acidity|       citric_acid|    residual_sugar|         chlorides|free_sulfur_dioxide|total_sulfur_dioxide|
density|                pH|         sulphates|          alcohol|BinaryQuality|
+-----------------+-----------------+------------------+------------------+------------------+-------------------+--------------------+------
-----------+------------------+------------------+-----------------+-------------+
|4.250195706522404|3.909310549043959|               0.0|1.3475864700453726|1.6147776501679734| 1.0516094578454602|   1.033581535954439| 528.6
81902902932|22.735153833991657|3.3036987733010235|8.820761504486935|            0|
|4.479936066796656|4.914561890148259|               0.0|  1.84406565135161| 2.082213298240522| 2.3900214951033187|   2.036763614969041|528.15
20621530078|20.727206139754976|4.0116342498394815| 9.19611361004656|            0|
|4.479936066796656|4.244394329412059|0.2053376055640616| 1.631288871441781|1.9547309063309342| 1.4340128970619912|  1.6415706747511676|528.25
80176704421|21.115840878452943| 3.834650204887197| 9.19611361004656|            0|
+-----------------+-----------------+------------------+------------------+------------------+-------------------+--------------------+------
-----------+------------------+------------------+-----------------+-------------+
only showing top 3 rows
```

Vectorizamos los features para poder utilizar las funciones de Machine Learning de Spark (spark.ML)

In [71]:

```python
# Para realizar operaciones más detalladas es necesario expresar las filas originales en vectores
from pyspark.ml.feature import VectorAssembler

assembler = VectorAssembler(
    inputCols=WineQualityDF.drop("BinaryQuality").schema.names,
    outputCol='features')

vector_df = assembler.transform(df)
vector_df = vector_df.select(['features', "BinaryQuality"])
vector_df.printSchema()
vector_df.show(n=3)
```

```
root
 |-- features: vector (nullable = true)
 |-- BinaryQuality: integer (nullable = true)

+--------------------+-------------+
|            features|BinaryQuality|
+--------------------+-------------+
|[4.25019570652240...|            0|
|[4.47993606679665...|            0|
|[4.47993606679665...|            0|
+--------------------+-------------+
only showing top 3 rows
```

Entrenamiento de modelos

Dividimos el dataset en Training (90%) y Testing (10%). Se utiliza esta partición para de 90% para training con el objetivo de aplicar el protocolo de validación K Fold. Se utiliza el 10% restante para validar el modelo

In [72]:

```python
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.regression import LinearRegression
from pyspark.ml.tuning import ParamGridBuilder, TrainValidationSplit

# Prepare train/test and validation data.

print("****************************")
trainDF, testDF = vector_df.randomSplit([0.9,0.1], seed = 21)
print('trainDF--> Qty Filas: {}  Cantidad Columnas: {}'.format(trainDF.count(), len(trainDF.columns)))
print('testDF--> Filas: {}  Cantidad Columnas: {}'.format(testDF.count(), len(testDF.columns)))
print("****************************")
```

```
****************************
trainDF--> Qty Filas: 1437  Cantidad Columnas: 2
testDF--> Filas: 162  Cantidad Columnas: 2
****************************
```

Clasificador - Arbol de Decision

El siguiente modelo utiliza el protocolo de validación K-Fold, además que se logra un entrenamiento entrenar un modelo predictivo

In [73]:

```python
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.mllib.evaluation import BinaryClassificationMetrics
#from mmlspark import ComputeModelStatistics

# Crear el model inicial de arbol de decision
dt = DecisionTreeClassifier(labelCol="BinaryQuality", featuresCol="features", maxDepth=8)

# crear grilla para probar el modelo
dtparamGrid = (ParamGridBuilder()
               .addGrid(dt.maxDepth, [4])
               .build())

# Evaluar el modelo
dtevaluator = BinaryClassificationEvaluator()
dtevaluator.setRawPredictionCol("prediction")
dtevaluator.setLabelCol("BinaryQuality")


# Create 5-fold CrossValidator
dtcv = CrossValidator(estimator = dt, estimatorParamMaps = dtparamGrid,evaluator = dtevaluator,numFolds = 5)#

# Run cross validations
dtcvModel = dtcv.fit(trainDF)
predictions = dtcvModel.transform(testDF)

print(dtcvModel)

print("****Evaluar Underfitting / Overfitting del modelo****")
print("dtpredictionsTrain")
dtpredictionsTrain = dtcvModel.transform(trainDF)
print("areaUnderROC Train",dtevaluator.evaluate(dtpredictionsTrain, {dtevaluator.metricName: "areaUnderROC"}))
print("*******************************************************")
print("dtpredictionsTest")
dtpredictionsTest = dtcvModel.transform(testDF)
print("areaUnderROC Test",dtevaluator.evaluate(dtpredictionsTest, {dtevaluator.metricName: "areaUnderROC"}))
print("*******************************************************")


predictionsDF1 = dtpredictionsTest
```

```
CrossValidatorModel_f98dbcb2fdaa
****Evaluar Underfitting / Overfitting del modelo****
dtpredictionsTrain
areaUnderROC Train 0.7556208527116699
*******************************************************
dtpredictionsTest
areaUnderROC Test 0.6858108108108107
*******************************************************
```

Se observa que el modelo se entrenó de manera correcta puesto que los valores de areaUnderROC para Training y Validación son muy cercanos, lo cual significa que no hubo overfitting ni underfitting.

Escritura de la tabla **modelo1** a la base de datos

In [74]:

```python
from pyspark.ml.functions import vector_to_array

features_col_names = WineQualityDF.drop("BinaryQuality").schema.names
print(features_col_names)


ExpandedDFtoDB = (predictionsDF1.drop("rawPrediction","probability").withColumn("xs", vector_to_array("features")))\
    .select([col("xs")[i].alias(features_col_names[i]) for i in range(11)]+["BinaryQuality"]+["prediction"])


ExpandedDFtoDB.show(n=3)

# Almacenar el conjunto de datos limpio en la base de datos
ExpandedDFtoDB \
    .write \
    .format("jdbc") \
    .mode('overwrite') \
    .option("url", "jdbc:postgresql://host.docker.internal:5433/postgres") \
    .option("user", "postgres") \
    .option("password", "testPassword") \
    .option("dbtable", "modelo1") \
    .save()
```

```
['fixed_acidity', 'volatile_acidity', 'citric_acid', 'residual_sugar', 'chlorides', 'free_sulfur_dioxide', 'total_sulfur_dioxide', 'density',
'pH', 'sulphates', 'alcohol']
+------------------+------------------+------------------+------------------+------------------+-------------------+--------------------+-----
------------+------------------+------------------+------------------+-------------+----------+
|     fixed_acidity|  volatile_acidity|       citric_acid|    residual_sugar|         chlorides|free_sulfur_dioxide|total_sulfur_dioxide|
density|                pH|         sulphates|           alcohol|BinaryQuality|prediction|
+------------------+------------------+------------------+------------------+------------------+-------------------+--------------------+-----
------------+------------------+------------------+------------------+-------------+----------+
|3.2163642222241644|3.685921584049102|               0.0|  1.77314011441494|1.4023069442176455| 0.6692060186289293| 0.45599185409754656| 525.
905520951013|22.79992629044132|3.4216879021789866|12.105087729867288|            0|       1.0|
|3.5035396040990325|3.937234169668316|0.5133440330337092|1.9859168943247691|1.7210130822946599| 1.2428111774537256|  0.8511847943154203|527.8
924316585733|23.31810594203861| 3.893645120961519| 9.571464820698306|            0|       1.0|
| 3.560974625699649|2.178044405954744|2.2073793458696604| 1.418512091531952| 1.508542376344332| 1.3384120372578585|  0.7295869665560745|526.8
168331450772|22.34651909529369|  5.13253202908615| 10.50984374223555|            1|       1.0|
+------------------+------------------+------------------+------------------+------------------+-------------------+--------------------+-----
------------+------------------+------------------+------------------+-------------+----------+
```

```
only showing top 3 rows
```

Clasificador - LogisticRegression

El siguiente model utilizar el protocolo de validación K-Fold, además que se logra un entrenamiento entrenar un modelo predictivo

In [75]:

```python
#https://gist.github.com/colbyford/7758088502211daa90dbc1b51c408762

from pyspark.ml.classification import LogisticRegression
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.mllib.evaluation import BinaryClassificationMetrics
#from mmlspark import ComputeModelStatistics

# Crear el model inicial de regresion logistiva
lr = LogisticRegression(labelCol="BinaryQuality", featuresCol="features")
# crear grilla para probar el modelo
lrparamGrid = (ParamGridBuilder()
             .addGrid(lr.maxIter, [15])
             .build())

# Evaluar el model
dtevaluator = BinaryClassificationEvaluator()
dtevaluator.setRawPredictionCol("prediction")
dtevaluator.setLabelCol("BinaryQuality")


# Create 5-fold CrossValidator
dtcv = CrossValidator(estimator = lr, estimatorParamMaps = lrparamGrid,evaluator = dtevaluator,numFolds = 5)#

# Run cross validations
dtcvModel = dtcv.fit(trainDF)
predictions = dtcvModel.transform(testDF)

print(dtcvModel)

print("****Evaluar Underfitting / Overfitting del modelo****")
print("dtpredictionsTrain")
dtpredictionsTrain = dtcvModel.transform(trainDF)
print("areaUnderROC Train",dtevaluator.evaluate(dtpredictionsTrain, {dtevaluator.metricName: "areaUnderROC"}))
print("*******************************************************")

print("dtpredictionsTest")
dtpredictionsTest = dtcvModel.transform(testDF)
print("areaUnderROC Test",dtevaluator.evaluate(dtpredictionsTest, {dtevaluator.metricName: "areaUnderROC"}))
print("*******************************************************")


predictionsDF2 = dtpredictionsTest
```

```
CrossValidatorModel_a15194f49ad1
****Evaluar Underfitting / Overfitting del modelo****
dtpredictionsTrain
areaUnderROC Train 0.7449775243729204
*******************************************************
dtpredictionsTest
areaUnderROC Test 0.7249692874692876
*******************************************************
```

Se observa que el modelo se entrenó de manera correcta puesto que los valores de areaUnderROC para Training y Validación son muy cercanos, lo cual significa que no hubo overfitting ni underfitting.

Escritura de la tabla **modelo2** a la base de datos

In [76]:

```python
from pyspark.ml.functions import vector_to_array

features_col_names = WineQualityDF.drop("BinaryQuality").schema.names
print(features_col_names)


ExpandedDFtoDB = (predictionsDF2.drop("rawPrediction","probability").withColumn("xs", vector_to_array("features")))\
    .select([col("xs")[i].alias(features_col_names[i]) for i in range(11)]+["BinaryQuality"]+["prediction"])


ExpandedDFtoDB.show(n=3)

# Almacenar el conjunto de datos limpio en la base de datos
ExpandedDFtoDB \
    .write \
    .format("jdbc") \
    .mode('overwrite') \
    .option("url", "jdbc:postgresql://host.docker.internal:5433/postgres") \
    .option("user", "postgres") \
    .option("password", "testPassword") \
    .option("dbtable", "modelo2") \
    .save()
```

```
['fixed_acidity', 'volatile_acidity', 'citric_acid', 'residual_sugar', 'chlorides', 'free_sulfur_dioxide', 'total_sulfur_dioxide', 'density',
'pH', 'sulphates', 'alcohol']
+------------------+-----------------+-------------------+------------------+-------------------+-------------------+--------------------+-----
------------+-----------------+------------------+------------------+-------------+----------+
|     fixed_acidity| volatile_acidity|        citric_acid|    residual_sugar|          chlorides|free_sulfur_dioxide|total_sulfur_dioxide|
```

```
density|                pH|         sulphates|           alcohol|BinaryQuality|prediction|
+------------------+----------------+------------------+------------------+-------------------+--------------------+--------------------+-----
------------+----------------+------------------+------------------+-------------+----------+
|3.2163642222241644|3.685921584049102|                0.0|  1.77314011441494|1.4023069442176455| 0.6692060186289293| 0.45599185409754656| 525.
905520951013|22.79992629044132|3.4216879021789866|12.105087729867288|            0|       1.0|
|3.5035396040990325|3.937234169668316|0.5133440330337092|1.9859168943247691|1.7210130822946599| 1.2428111774537256|  0.8511847943154203|527.8
924316585733|23.31810594203861| 3.893645120961519| 9.571464820698306|            0|       0.0|
| 3.560974625699649|2.178044405954744|2.2073793458696604| 1.418512091531952| 1.508542376344332| 1.3384120372578585|  0.7295869665560745|526.8
168331450772|22.34651909529369|  5.13253202908615| 10.50984374223555|            1|       1.0|
+------------------+----------------+------------------+------------------+-------------------+--------------------+--------------------+-----
------------+----------------+------------------+------------------+-------------+----------+
only showing top 3 rows
```

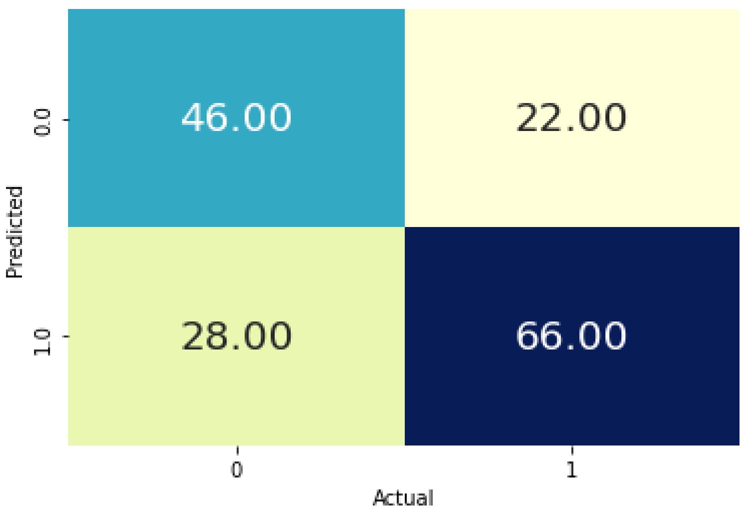
Análisis de resultados

Basado en la matriz de decisión generada en las siguientes secciones de código se observa que el modelo de Regresión logística tiene más Falsos negativos en comparación con el modelo de arbol de decisión. Ambos modelos tiene una taza de acierto similar.

Matriz de confusión Arbol de decisión

In [77]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
pandasDF1 = predictionsDF1.drop("features","rawPrediction","probability").toPandas()
pandasDF1.head()
print("size" ,len(pandasDF1.index))

y_true = pandasDF1["BinaryQuality"]
y_pred = pandasDF1["prediction"]


confusion_matrix = pd.crosstab(y_true, y_pred, rownames=['Actual'], colnames=['Predicted'])

plt.clf()

ax = fig.add_subplot(111)

ax.set_aspect(1)

res = sn.heatmap(confusion_matrix.T, annot=True, fmt='.2f', cmap="YlGnBu", cbar=False,annot_kws={"fontsize":20})
ax.legend( fontsize=20)



plt.show()
```

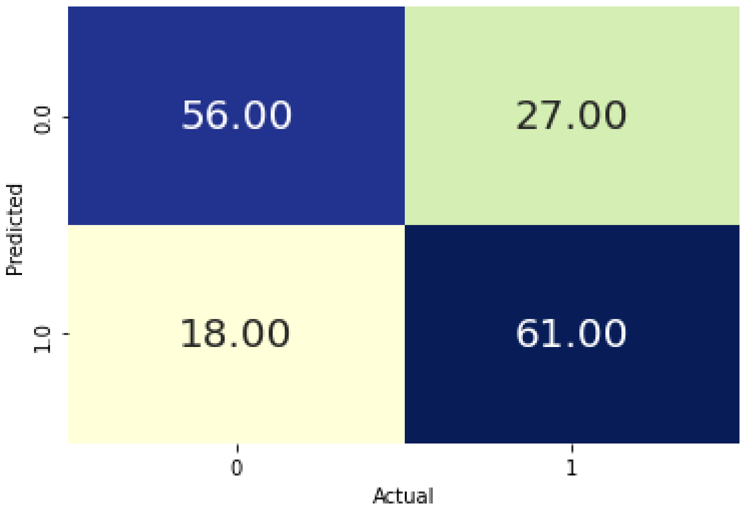
```
No handles with labels found to put in legend.
size 162
```



Matriz de confusión Regresión logística

In [78]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
pandasDF2 = predictionsDF2.drop("features","rawPrediction","probability").toPandas()
pandasDF2.head()
print("size" ,len(pandasDF1.index))

y_true = pandasDF2["BinaryQuality"]
y_pred = pandasDF2["prediction"]


confusion_matrix = pd.crosstab(y_true, y_pred, rownames=['Actual'], colnames=['Predicted'])

plt.clf()

ax = fig.add_subplot(111)

ax.set_aspect(1)

res = sn.heatmap(confusion_matrix.T, annot=True, fmt='.2f', cmap="YlGnBu", cbar=False,annot_kws={"fontsize":20})
ax.legend( fontsize=20)



plt.show()
```

```
No handles with labels found to put in legend.
size 162
```



In [ ]: