Classifying Diabetes with PySpark ML

Yu chien (Calvin) Ma

Installing Dependencies & Run Spark Session

```
#install pyspark
import findspark
findspark.init()
findspark.find()
```

```
'C:\\Users\\calvi\\anaconda3\\lib\\site-packages\\pyspark'
```

```
#create a sparksession
from pyspark.sql import SparkSession

spark = SparkSession.builder \
.appName('spark') \
.master('local[*]') \
.config('spark.sql.execution.arrow.pysupark.enabled', True) \
.config('spark.sql.session.timeZone', 'UTC') \
.config('spark.driver.memory','16G') \
.config('spark.ui.showConsoleProgress', True) \
.config('spark.sql.repl.eagerEval.enabled', True) \
.getOrCreate()
```

Cloning and exploring the dataset

```
#clone the diabetes dataset from the github repository
! git clone
https://github.com/calvinma888/PySparkML_DiabetesClassification.git
```

```
Cloning into 'diabetes_dataset'...
```

```
#check if the dataset exists
path = r"C:\Users\calvi\Documents\Portfolio Projects\diabetes_dataset\
diabetes.csv"


#create spark dataframe
df = spark.read.csv(path, header=True, inferSchema= True)

#display the dataframe
df.show(10)
```

```
+-----------+-------+-------------+-------------+-------+----
+-----------------------+---+-------+
|Pregnancies|Glucose|BloodPressure|SkinThickness|Insulin| BMI|
```

```
DiabetesPedigreeFunction|Age|Outcome|
+-----------+-------+-------------+-------------+-------+----
+------------------------+---+-------+
|           2|    138|           62|           35|      0|33.6|
0.127| 47|      1|
|           0|     84|           82|           31|    125|38.2|
0.233| 23|      0|
|           0|    145|            0|            0|      0|44.2|
0.63| 31|      1|
|           0|    135|           68|           42|    250|42.3|
0.365| 24|      1|
|           1|    139|           62|           41|    480|40.7|
0.536| 21|      0|
|           0|    173|           78|           32|    265|46.5|
1.159| 58|      0|
|           4|     99|           72|           17|      0|25.6|
0.294| 28|      0|
|           8|    194|           80|            0|      0|26.1|
0.551| 67|      0|
|           2|     83|           65|           28|     66|36.8|
0.629| 24|      0|
|           2|     89|           90|           30|      0|33.5|
0.292| 42|      0|
+-----------+-------+-------------+-------------+-------+----
+------------------------+---+-------+
only showing top 10 rows
```

```
#print the schema
df.printSchema()
```

```
root
 |-- Pregnancies: integer (nullable = true)
 |-- Glucose: integer (nullable = true)
 |-- BloodPressure: integer (nullable = true)
 |-- SkinThickness: integer (nullable = true)
 |-- Insulin: integer (nullable = true)
 |-- BMI: double (nullable = true)
 |-- DiabetesPedigreeFunction: double (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Outcome: integer (nullable = true)
```

```
#print the number of rows and columns
print(("Rows:", df.count(), "Columns", len(df.columns)))
```

```
('Rows:', 2000, 'Columns', 9)
```

```
#count the total number of diabetic and non-diabetic class
df.groupBy("Outcome").count().show()
```

```
+-------+-----+
|Outcome|count|
+-------+-----+
|      1|  684|
|      0| 1316|
+-------+-----+
```

```
#get the summary statistics
df.describe()
```

```
+-------+-----------------+------------------+------------------
+-----------------+-----------------+------------------
+-----------------------+------------------+------------------+
|summary|      Pregnancies|           Glucose|     BloodPressure|
SkinThickness|          Insulin|               BMI|
DiabetesPedigreeFunction|               Age|           Outcome|
+-------+-----------------+------------------+------------------
+-----------------+-----------------+------------------
+-----------------------+------------------+------------------+
|  count|             2000|              2000|              2000|
2000|             2000|              2000|                  2000|
2000|              2000|
|   mean|           3.7035|          121.1825|           69.1455|
20.935|           80.254|32.192999999999984|     0.47092999999999974|
33.0905|             0.342|
| stddev|3.306063032730656|32.068635649902916|19.188314815604098|
16.10324290992682|111.1805335457595| 8.149900701279762|
0.3235525586811429|11.786423106049496|0.4744982342297426|
|    min|                0|                 0|                 0|
0|                0|               0.0|                 0.078|
21|                  0|
|    max|               17|               199|               122|
110|              744|              80.6|                 2.42|
81|                  1|
+-------+-----------------+------------------+------------------
+-----------------+-----------------+------------------
+-----------------------+------------------+------------------+
```

Data Cleaning & Preparation

```
#check for null values
for col in df.columns:
    print(col+":", df[df[col].isNull()].count())
```

```
Pregnancies: 0
Glucose: 0
BloodPressure: 0
SkinThickness: 0
Insulin: 0
```

```
BMI: 0
DiabetesPedigreeFunction: 0
Age: 0
Outcome: 0
```

```
#look for the unnecessary values present
def count_zeroes():
    columns_list =
['Glucose','BloodPressure','SkinThickness','Insulin','BMI']
    for i in columns_list:
        print(i+":",df[df[i]==0].count())


count_zeroes()
```

```
Glucose: 13
BloodPressure: 90
SkinThickness: 573
Insulin: 956
BMI: 28
```

```
#impute the zero values by the mean value
from pyspark.sql.functions import *

for i in df.columns[1:6]:
    mean = df.agg({i:'mean'}).first()[0]
    print(f'mean value for {i} is {float(mean)}')

    #imputation
    df = df.withColumn(i, when(df[i]==0, int(mean)).otherwise(df[i]))
```

```
mean value for Glucose is 121.1825
mean value for BloodPressure is 69.1455
mean value for SkinThickness is 20.935
mean value for Insulin is 80.254
mean value for BMI is 32.192999999999984
```

```
#display the dataframe
df.show(10)
```

```
+-----------+-------+-------------+-------------+-------+----
+-------------------------+---+-------+
|Pregnancies|Glucose|BloodPressure|SkinThickness|Insulin| BMI|
DiabetesPedigreeFunction|Age|Outcome|
+-----------+-------+-------------+-------------+-------+----
+-------------------------+---+-------+
|          2|    138|           62|           35|     80|33.6|
0.127| 47|      1|
|          0|     84|           82|           31|    125|38.2|
0.233| 23|      0|
|          0|    145|           69|           20|     80|44.2|
```

```
0.63| 31|      1|
|           0|    135|           68|           42|    250|42.3|
0.365| 24|      1|
|           1|    139|           62|           41|    480|40.7|
0.536| 21|      0|
|           0|    173|           78|           32|    265|46.5|
1.159| 58|      0|
|           4|     99|           72|           17|     80|25.6|
0.294| 28|      0|
|           8|    194|           80|           20|     80|26.1|
0.551| 67|      0|
|           2|     83|           65|           28|     66|36.8|
0.629| 24|      0|
|           2|     89|           90|           30|     80|33.5|
0.292| 42|      0|
+------------+-------+-------------+-------------+-------+----
+------------------------+---+-------+
only showing top 10 rows
```

Correlation Analysis & Feature Selection

```
#find the correlation among the set of input & output variables
for i in df.columns:
    print(f"Correlation to the outcome for {i} is
{df.stat.corr('Outcome',i)}")
```

```
Correlation to the outcome for Pregnancies is 0.22443699263363961
Correlation to the outcome for Glucose is 0.48796646527321064
Correlation to the outcome for BloodPressure is 0.17171333286446713
Correlation to the outcome for SkinThickness is 0.1659010662889893
Correlation to the outcome for Insulin is 0.1711763270226193
Correlation to the outcome for BMI is 0.2827927569760082
Correlation to the outcome for DiabetesPedigreeFunction is
0.1554590791569403
Correlation to the outcome for Age is 0.23650924717620253
Correlation to the outcome for Outcome is 1.0
```

```
#feature selection
from pyspark.ml.feature import VectorAssembler
assembler =
VectorAssembler(inputCols=['Pregnancies','Glucose','BloodPressure','Sk
inThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']
                               ,outputCol='features')

output_data = assembler.transform(df)

#display dataframe
output_data.show(10)
```

```
+-----------+-------+-------------+-------------+-------+----
+------------------------+---+-------+--------------------+
|Pregnancies|Glucose|BloodPressure|SkinThickness|Insulin| BMI|
DiabetesPedigreeFunction|Age|Outcome|            features|
+-----------+-------+-------------+-------------+-------+----
+------------------------+---+-------+--------------------+
|          2|    138|           62|           35|     80|33.6|
0.127| 47|      1|[2.0,138.0,62.0,3...|
|          0|     84|           82|           31|    125|38.2|
0.233| 23|      0|[0.0,84.0,82.0,31...|
|          0|    145|           69|           20|     80|44.2|
0.63| 31|      1|[0.0,145.0,69.0,2...|
|          0|    135|           68|           42|    250|42.3|
0.365| 24|      1|[0.0,135.0,68.0,4...|
|          1|    139|           62|           41|    480|40.7|
0.536| 21|      0|[1.0,139.0,62.0,4...|
|          0|    173|           78|           32|    265|46.5|
1.159| 58|      0|[0.0,173.0,78.0,3...|
|          4|     99|           72|           17|     80|25.6|
0.294| 28|      0|[4.0,99.0,72.0,17...|
|          8|    194|           80|           20|     80|26.1|
0.551| 67|      0|[8.0,194.0,80.0,2...|
|          2|     83|           65|           28|     66|36.8|
0.629| 24|      0|[2.0,83.0,65.0,28...|
|          2|     89|           90|           30|     80|33.5|
0.292| 42|      0|[2.0,89.0,90.0,30...|
+-----------+-------+-------------+-------------+-------+----
+------------------------+---+-------+--------------------+
only showing top 10 rows
```

```
#print the schema
output_data.printSchema()
```

```
root
 |-- Pregnancies: integer (nullable = true)
 |-- Glucose: integer (nullable = true)
 |-- BloodPressure: integer (nullable = true)
 |-- SkinThickness: integer (nullable = true)
 |-- Insulin: integer (nullable = true)
 |-- BMI: double (nullable = true)
 |-- DiabetesPedigreeFunction: double (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Outcome: integer (nullable = true)
 |-- features: vector (nullable = true)
```

Splitting Dataset & Building the Model

```
#create final data
from pyspark.ml.classification import LogisticRegression
final_data = output_data.select('Outcome','features')

#print schema of final data
final_data.printSchema()
```

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

```
#split the dataset
train, test = final_data.randomSplit([0.7,0.3])

#build the model
model = LogisticRegression(labelCol='Outcome').fit(train)

#summary of the model
summary = model.summary

summary.predictions.describe()
```

```
+-------+-------------------+-------------------+
|summary|            Outcome|         prediction|
+-------+-------------------+-------------------+
|  count|               1415|               1415|
|   mean| 0.3469964664310954|0.27703180212014133|
| stddev|0.47618291252699946| 0.4476905484092893|
|    min|                0.0|                0.0|
|    max|                1.0|                1.0|
+-------+-------------------+-------------------+
```

```
summary.predictions.show(10)
```

```
+-------+--------------------+--------------------
+--------------------+----------+
|Outcome|            features|       rawPrediction|
probability|prediction|
+-------+--------------------+--------------------
+--------------------+----------+
|    0.0|[0.0,67.0,76.0,20...|[2.19680167434635...|
[0.89996193229190...|       0.0|
|    0.0|[0.0,67.0,76.0,20...|[2.19680167434635...|
[0.89996193229190...|       0.0|
|    0.0|[0.0,73.0,69.0,20...|[4.06019499217821...|
[0.98304671451051...|       0.0|
|    0.0|[0.0,74.0,52.0,10...|[3.56172547534394...|
[0.97239393406115...|       0.0|
|    0.0|[0.0,74.0,52.0,10...|[3.56172547534394...|
```

```
[0.97239393406115...|       0.0|
|    0.0|[0.0,78.0,88.0,29...|[2.41484992259258...|
[0.91795269763392...|       0.0|
|    0.0|[0.0,84.0,64.0,22...|[2.39725330567857...|
[0.91661761439255...|       0.0|
|    0.0|[0.0,84.0,64.0,22...|[2.39725330567857...|
[0.91661761439255...|       0.0|
|    0.0|[0.0,84.0,82.0,31...|[2.35324281764007...|
[0.91319163905189...|       0.0|
|    0.0|[0.0,84.0,82.0,31...|[2.35324281764007...|
[0.91319163905189...|       0.0|
+-------+--------------------+--------------------
+--------------------+----------+
only showing top 10 rows


summary.areaUnderROC

0.8412463300447003

summary.accuracy

0.768904593639576
```

Evaluating and Saving the Model

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
predictions = model.evaluate(test)

predictions.predictions.show(10)

+-------+--------------------+--------------------
+--------------------+----------+
|Outcome|            features|       rawPrediction|
probability|prediction|
+-------+--------------------+--------------------
+--------------------+----------+
|      0|[0.0,57.0,60.0,20...|[4.14482131532745...|
[0.98440092000597...|       0.0|
|      0|[0.0,57.0,60.0,20...|[4.14482131532745...|
[0.98440092000597...|       0.0|
|      0|[0.0,73.0,69.0,20...|[4.06019499217821...|
[0.98304671451051...|       0.0|
|      0|[0.0,74.0,52.0,10...|[3.56172547534394...|
[0.97239393406115...|       0.0|
|      0|[0.0,78.0,88.0,29...|[2.41484992259258...|
[0.91795269763392...|       0.0|
|      0|[0.0,84.0,64.0,22...|[2.39725330567857...|
[0.91661761439255...|       0.0|
|      0|[0.0,84.0,82.0,31...|[2.35324281764007...|
```

```
[0.91319163905189...|       0.0|
|      0|[0.0,84.0,82.0,31...|[2.35324281764007...|
[0.91319163905189...|       0.0|
|      0|[0.0,91.0,68.0,32...|[2.07394264547954...|
[0.88834462551579...|       0.0|
|      0|[0.0,91.0,80.0,20...|[2.24722913503398...|
[0.90441125785237...|       0.0|
+-------+--------------------+--------------------
+--------------------+----------+
only showing top 10 rows


evaluator =
BinaryClassificationEvaluator(rawPredictionCol='rawPrediction',labelCo
l= 'Outcome')

print(evaluator.explainParams())

labelCol: label column name. (default: label, current: Outcome)
metricName: metric name in evaluation (areaUnderROC|areaUnderPR)
(default: areaUnderROC)
numBins: Number of bins to down-sample the curves (ROC curve, PR
curve) in area computation. If 0, no down-sampling will occur. Must be
>= 0. (default: 1000)
rawPredictionCol: raw prediction (a.k.a. confidence) column name.
(default: rawPrediction, current: rawPrediction)
weightCol: weight column name. If this is not set or empty, we treat
all instance weights as 1.0. (undefined)

print(f"Area Under ROC: {evaluator.evaluate(model.transform(test))}")

Area Under ROC: 0.8371576609918576

# save model
# model.save("model")

# load saved model back to the environment
# from pyspark.ml.classification import LogisticRegressionModel
# model = LogisticRegressionModel.load("model")
```

Prediction on new data with the trained model

```
#create a new spark dataframe
path2 = r"C:\Users\calvi\Documents\Portfolio Projects\
diabetes_dataset\new_test.csv"
test_df = spark.read.csv(path, header=True,inferSchema=True)

#print the schema
test_df.printSchema()
```

```
root
 |-- Pregnancies: integer (nullable = true)
 |-- Glucose: integer (nullable = true)
 |-- BloodPressure: integer (nullable = true)
 |-- SkinThickness: integer (nullable = true)
 |-- Insulin: integer (nullable = true)
 |-- BMI: double (nullable = true)
 |-- DiabetesPedigreeFunction: double (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Outcome: integer (nullable = true)
```

```
#create an additional feature merged column
test_data = assembler.transform(test_df)
```

```
#print the schema
test_data.printSchema()
```

```
root
 |-- Pregnancies: integer (nullable = true)
 |-- Glucose: integer (nullable = true)
 |-- BloodPressure: integer (nullable = true)
 |-- SkinThickness: integer (nullable = true)
 |-- Insulin: integer (nullable = true)
 |-- BMI: double (nullable = true)
 |-- DiabetesPedigreeFunction: double (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Outcome: integer (nullable = true)
 |-- features: vector (nullable = true)
```

```
#use model to make predictions
results = model.transform(test_data)
results.printSchema()
```

```
root
 |-- Pregnancies: integer (nullable = true)
 |-- Glucose: integer (nullable = true)
 |-- BloodPressure: integer (nullable = true)
 |-- SkinThickness: integer (nullable = true)
 |-- Insulin: integer (nullable = true)
 |-- BMI: double (nullable = true)
 |-- DiabetesPedigreeFunction: double (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Outcome: integer (nullable = true)
 |-- features: vector (nullable = true)
 |-- rawPrediction: vector (nullable = true)
 |-- probability: vector (nullable = true)
 |-- prediction: double (nullable = false)
```

```
#display the predictions
results.select('features','prediction').show(10)
```

```
+--------------------+----------+
|            features|prediction|
+--------------------+----------+
|[2.0,138.0,62.0,3...|       0.0|
|[0.0,84.0,82.0,31...|       0.0|
|(8,[1,5,6,7],[145...|       1.0|
|[0.0,135.0,68.0,4...|       0.0|
|[1.0,139.0,62.0,4...|       0.0|
|[0.0,173.0,78.0,3...|       1.0|
|[4.0,99.0,72.0,17...|       0.0|
|[8.0,194.0,80.0,0...|       1.0|
|[2.0,83.0,65.0,28...|       0.0|
|[2.0,89.0,90.0,30...|       0.0|
+--------------------+----------+
only showing top 10 rows
```

```
print(f"Area Under ROC for new dataset:
{evaluator.evaluate(model.transform(test_data))}")
```

```
Area Under ROC for new dataset: 0.8295528271032212
```