IN-CLASS LAB ON MACHINE LEARNING WITH APACHE SPARK

1. Problem 1

The “diabetes.csv” dataset has historical data on individuals that eventually either developed diabetes or not. Diabetes is a condition where the body does not produce enough insulin to break down the food that you eat. Without medication, diabetes can lead to damage to cells and vital organs and eventual death.

In this problem, we want to predict whether a person is at risk of becoming diabetic based on the individual’s data. This information can then be used to begin preventative measures for the individual (example lifestyle change in diet and exercise). The features and label for the dataset are described below.

Features or independent Variables:

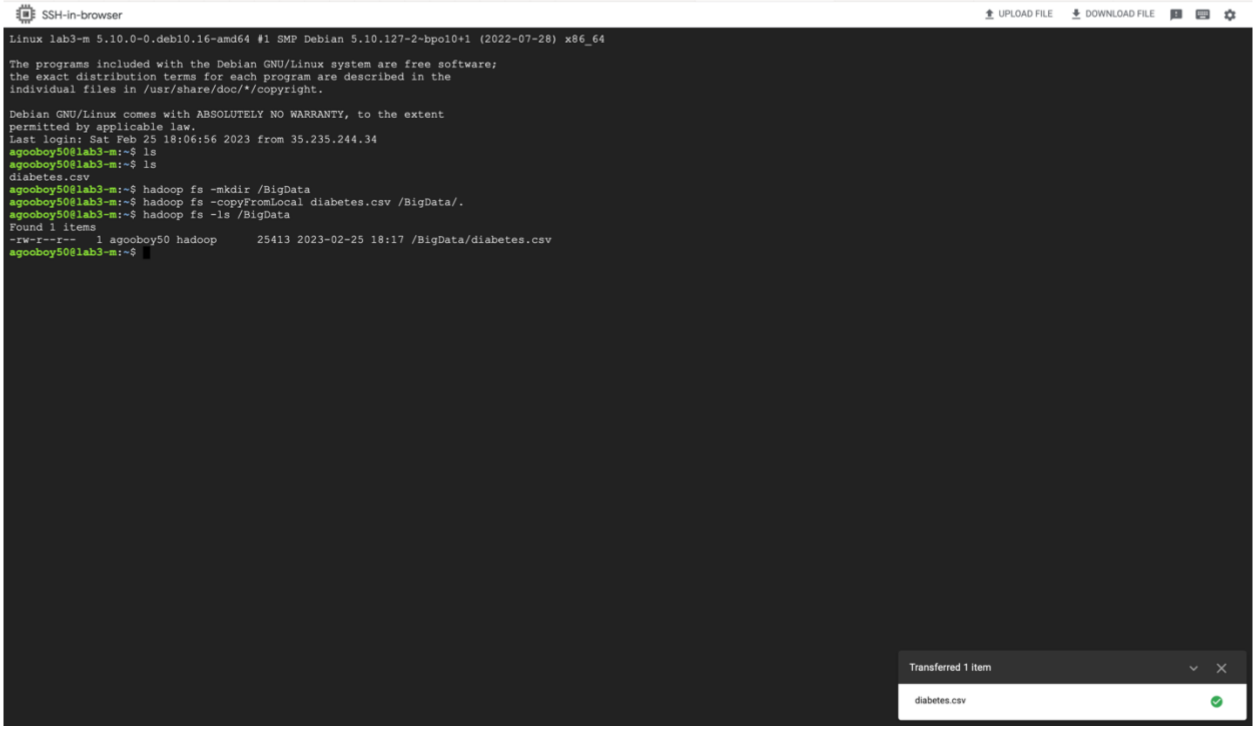
Pregnancies: Number of times pregnant
Glucose: Plasma glucose concentration, 2 hours in an oral glucose tolerance test
BloodPressure: Diastolic blood pressure (mm Hg)
SkinThickness: Triceps skin fold thickness (mm)
Insulin: 2-Hour serum insulin (mu U/ml)
BMI: Body mass index (weight in kg/(height in m)^2)
DiabetesPedigreeFunction: Diabetes pedigree function, a score based on genetic factor of a person (diabetes has a close relation to family history).

Age: Age (years)

Labels or Dependent Variable:

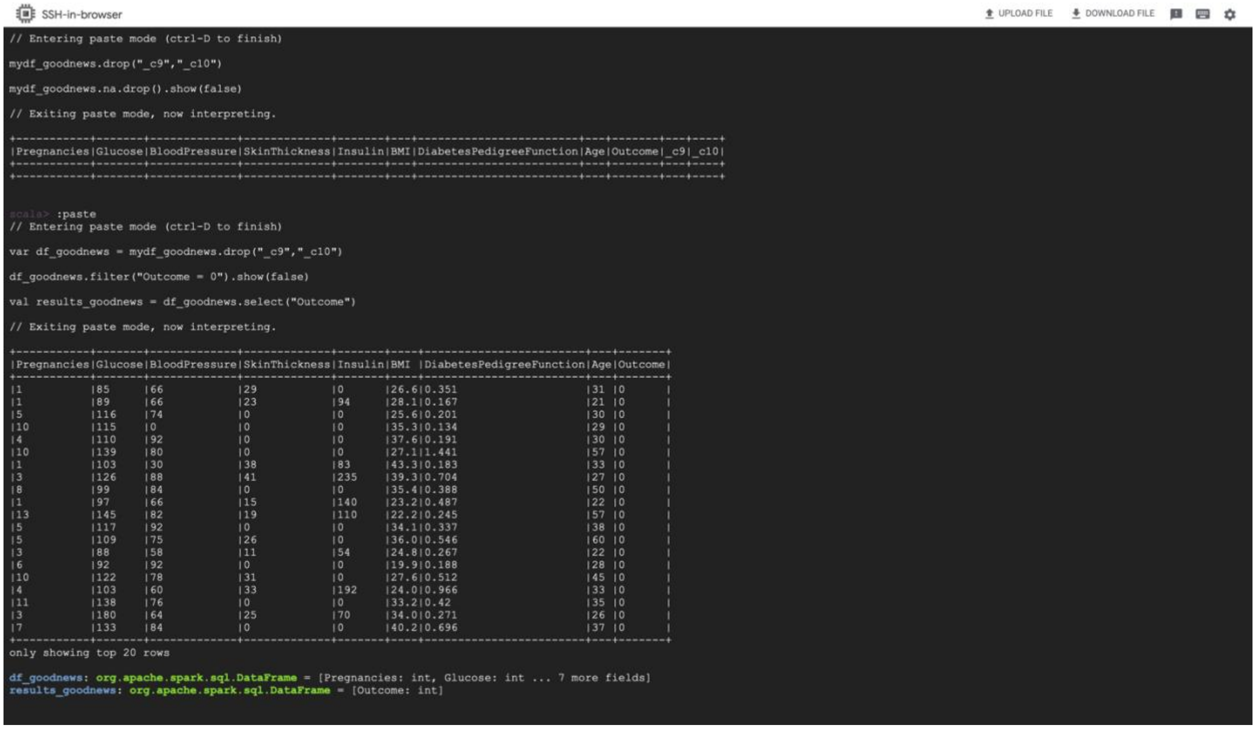
Outcome: No Diabetes = 0, Diabetes=1

Load into HDFS



```
Linux lab3-m 5.10.0-0.deb10.16-amd64 #1 SMP Debian 5.10.127-2~bpo10+1 (2022-07-28) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Feb 25 18:06:56 2023 from 35.235.244.34
agooboy50@lab3-m:~$ ls
agooboy50@lab3-m:~$ ls
diabetes.csv
agooboy50@lab3-m:~$ hadoop fs -mkdir /BigData
agooboy50@lab3-m:~$ hadoop fs -copyFromLocal diabetes.csv /BigData/.
agooboy50@lab3-m:~$ hadoop fs -ls /BigData
Found 1 items
-rw-r--r--   1 agooboy50 hadoop      25413 2023-02-25 18:17 /BigData/diabetes.csv
agooboy50@lab3-m:~$ 
```

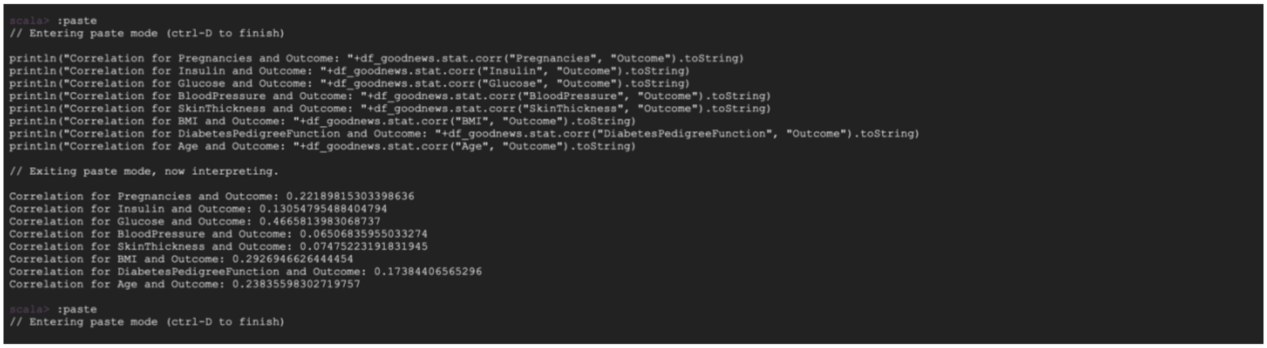
Removing "_c9","_c10" column from dataset.



```
// Entering paste mode (ctrl-D to finish)

mydf_goodnews.drop("_c9","_c10")

mydf_goodnews.na.drop().show(false)

// Exiting paste mode, now interpreting.

+-----------+-------+-------------+-------------+-------+---+------------------------+---+-------+---+----+
|Pregnancies|Glucose|BloodPressure|SkinThickness|Insulin|BMI|DiabetesPedigreeFunction|Age|Outcome|_c9|_c10|
+-----------+-------+-------------+-------------+-------+---+------------------------+---+-------+---+----+
+-----------+-------+-------------+-------------+-------+---+------------------------+---+-------+---+----+


scala> :paste
// Entering paste mode (ctrl-D to finish)

var df_goodnews = mydf_goodnews.drop("_c9","_c10")

df_goodnews.filter("Outcome = 0").show(false)

val results_goodnews = df_goodnews.select("Outcome")

// Exiting paste mode, now interpreting.

+-----------+-------+-------------+-------------+-------+----+------------------------+---+-------+
|Pregnancies|Glucose|BloodPressure|SkinThickness|Insulin|BMI |DiabetesPedigreeFunction|Age|Outcome|
+-----------+-------+-------------+-------------+-------+----+------------------------+---+-------+
|1          |85     |66           |29           |0      |26.6|0.351                   |31 |0      |
|1          |89     |66           |23           |94     |28.1|0.167                   |21 |0      |
|5          |116    |74           |0            |0      |25.6|0.201                   |30 |0      |
|10         |115    |0            |0            |0      |35.3|0.134                   |29 |0      |
|4          |110    |92           |0            |0      |37.6|0.191                   |30 |0      |
|10         |139    |80           |0            |0      |27.1|1.441                   |57 |0      |
|1          |103    |30           |38           |83     |43.3|0.183                   |33 |0      |
|3          |126    |88           |41           |235    |39.3|0.704                   |27 |0      |
|8          |99     |84           |0            |0      |35.4|0.388                   |50 |0      |
|1          |97     |66           |15           |140    |23.2|0.487                   |22 |0      |
|13         |145    |82           |19           |110    |22.2|0.245                   |57 |0      |
|5          |117    |92           |0            |0      |34.1|0.337                   |38 |0      |
|5          |109    |75           |26           |0      |36.0|0.546                   |60 |0      |
|3          |88     |58           |11           |54     |24.8|0.267                   |22 |0      |
|6          |92     |92           |0            |0      |19.9|0.188                   |28 |0      |
|10         |122    |78           |31           |0      |27.6|0.512                   |45 |0      |
|4          |103    |60           |33           |192    |24.0|0.966                   |33 |0      |
|11         |138    |76           |0            |0      |33.2|0.42                    |35 |0      |
|3          |180    |64           |25           |70     |34.0|0.271                   |26 |0      |
|7          |133    |84           |0            |0      |40.2|0.696                   |37 |0      |
+-----------+-------+-------------+-------------+-------+----+------------------------+---+-------+
only showing top 20 rows

df_goodnews: org.apache.spark.sql.DataFrame = [Pregnancies: int, Glucose: int ... 7 more fields]
results_goodnews: org.apache.spark.sql.DataFrame = [Outcome: int]
```

Selecting Features the dataset which can result to diabetes.

The correlation shows there's a significant weak correlation between SkinThickness, BloodPressure, and Outcome. We also experienced a weak correlation between insulin and outcome but we can consider it as cause of type 2 diabetes.



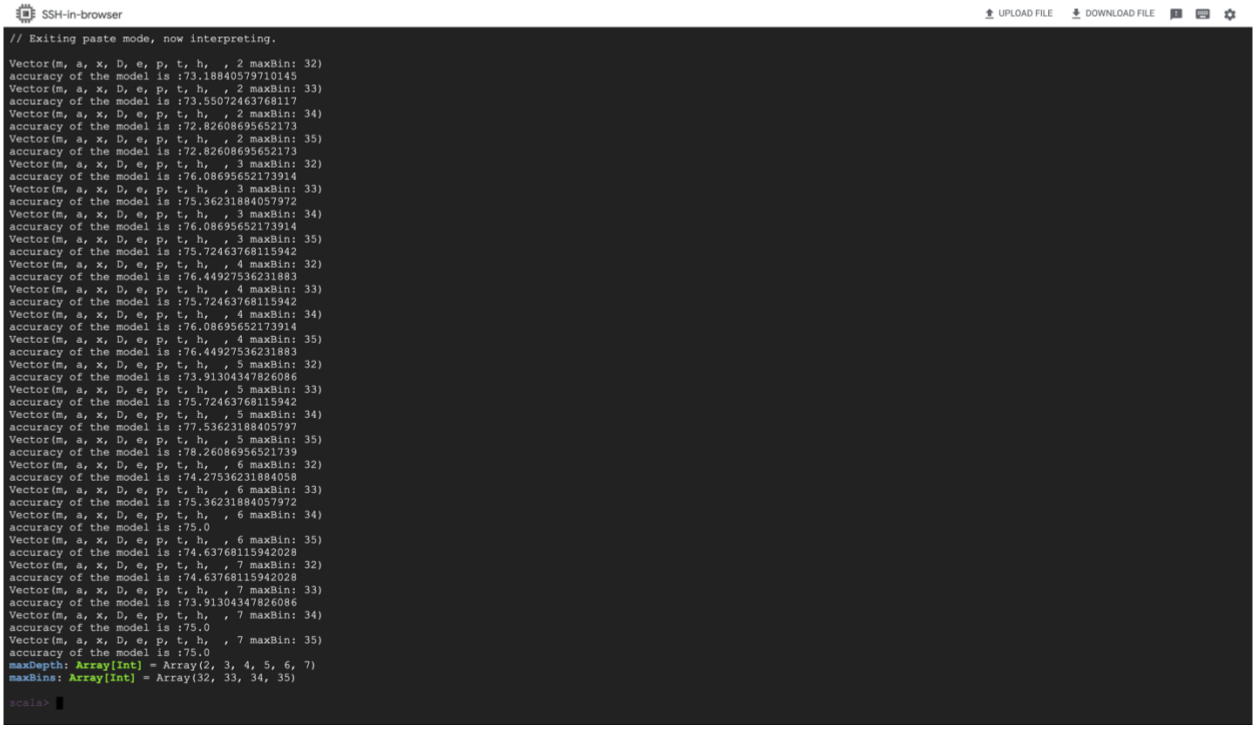
```
scala> :paste
// Entering paste mode (ctrl-D to finish)

println("Correlation for Pregnancies and Outcome: "+df_goodnews.stat.corr("Pregnancies", "Outcome").toString)
println("Correlation for Insulin and Outcome: "+df_goodnews.stat.corr("Insulin", "Outcome").toString)
println("Correlation for Glucose and Outcome: "+df_goodnews.stat.corr("Glucose", "Outcome").toString)
println("Correlation for BloodPressure and Outcome: "+df_goodnews.stat.corr("BloodPressure", "Outcome").toString)
println("Correlation for SkinThickness and Outcome: "+df_goodnews.stat.corr("SkinThickness", "Outcome").toString)
println("Correlation for BMI and Outcome: "+df_goodnews.stat.corr("BMI", "Outcome").toString)
println("Correlation for DiabetesPedigreeFunction and Outcome: "+df_goodnews.stat.corr("DiabetesPedigreeFunction", "Outcome").toString)
println("Correlation for Age and Outcome: "+df_goodnews.stat.corr("Age", "Outcome").toString)

// Exiting paste mode, now interpreting.

Correlation for Pregnancies and Outcome: 0.22189815303398636
Correlation for Insulin and Outcome: 0.13054795488404794
Correlation for Glucose and Outcome: 0.4665813983068737
Correlation for BloodPressure and Outcome: 0.06506835955033274
Correlation for SkinThickness and Outcome: 0.07475223191831945
Correlation for BMI and Outcome: 0.2926946626444454
Correlation for DiabetesPedigreeFunction and Outcome: 0.17384406565296
Correlation for Age and Outcome: 0.23835598302719757

scala> :paste
// Entering paste mode (ctrl-D to finish)
```

I will adopt Random Forest. Random Forest is a machine learning algorithm that is based on decision trees and is often used to avoid overfitting. Overfitting occurs when a model becomes too complex and starts to fit the noise in the data, rather than the underlying patterns. This can lead to poor performance on new data, as the model has essentially memorized the training data and cannot generalize well to new situations.

The parameters of a machine learning algorithm that can be tweaked to improve performance are called hyperparameters. The optimal values of hyperparameters can be difficult to predict in advance and tuning them to improve performance often requires a trial-and-error approach. One way to tune hyperparameters is to perform a grid search or random search over a range of values, train and evaluate the model with each combination of hyperparameters and select the combination that produces the best performance on a validation set.

maxDepth is a hyperparameter used in decision tree algorithms and other tree-based models, such as Random Forest. It specifies the maximum depth of a decision tree, which is the length of the longest path from the root node to a leaf node. In other words, it limits the number of nodes in a decision tree.

maxBins is an important hyperparameter that affects the accuracy and performance of decision tree algorithms. The default value of maxBin in spark is 32. and the value needs to be set >=2

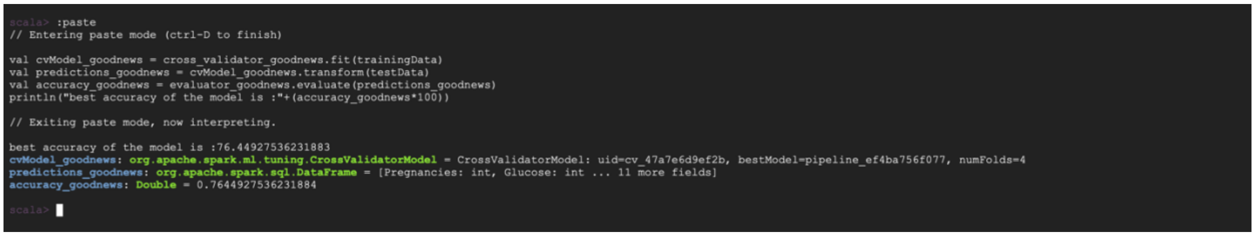


```
// Exiting paste mode, now interpreting.

Vector(m, a, x, D, e, p, t, h,  , 2 maxBin: 32)
accuracy of the model is :73.18840579710145
Vector(m, a, x, D, e, p, t, h,  , 2 maxBin: 33)
accuracy of the model is :73.55072463768117
Vector(m, a, x, D, e, p, t, h,  , 2 maxBin: 34)
accuracy of the model is :72.82608695652173
Vector(m, a, x, D, e, p, t, h,  , 2 maxBin: 35)
accuracy of the model is :72.82608695652173
Vector(m, a, x, D, e, p, t, h,  , 3 maxBin: 32)
accuracy of the model is :76.08695652173914
Vector(m, a, x, D, e, p, t, h,  , 3 maxBin: 33)
accuracy of the model is :75.36231884057972
Vector(m, a, x, D, e, p, t, h,  , 3 maxBin: 34)
accuracy of the model is :76.08695652173914
Vector(m, a, x, D, e, p, t, h,  , 3 maxBin: 35)
accuracy of the model is :75.72463768115942
Vector(m, a, x, D, e, p, t, h,  , 4 maxBin: 32)
accuracy of the model is :76.44927536231883
Vector(m, a, x, D, e, p, t, h,  , 4 maxBin: 33)
accuracy of the model is :75.72463768115942
Vector(m, a, x, D, e, p, t, h,  , 4 maxBin: 34)
accuracy of the model is :76.08695652173914
Vector(m, a, x, D, e, p, t, h,  , 4 maxBin: 35)
accuracy of the model is :76.44927536231883
Vector(m, a, x, D, e, p, t, h,  , 5 maxBin: 32)
accuracy of the model is :73.91304347826086
Vector(m, a, x, D, e, p, t, h,  , 5 maxBin: 33)
accuracy of the model is :75.72463768115942
Vector(m, a, x, D, e, p, t, h,  , 5 maxBin: 34)
accuracy of the model is :77.53623188405797
Vector(m, a, x, D, e, p, t, h,  , 5 maxBin: 35)
accuracy of the model is :78.26086956521739
Vector(m, a, x, D, e, p, t, h,  , 6 maxBin: 32)
accuracy of the model is :74.27536231884058
Vector(m, a, x, D, e, p, t, h,  , 6 maxBin: 33)
accuracy of the model is :75.36231884057972
Vector(m, a, x, D, e, p, t, h,  , 6 maxBin: 34)
accuracy of the model is :75.0
Vector(m, a, x, D, e, p, t, h,  , 6 maxBin: 35)
accuracy of the model is :74.63768115942028
Vector(m, a, x, D, e, p, t, h,  , 7 maxBin: 32)
accuracy of the model is :74.63768115942028
Vector(m, a, x, D, e, p, t, h,  , 7 maxBin: 33)
accuracy of the model is :73.91304347826086
Vector(m, a, x, D, e, p, t, h,  , 7 maxBin: 34)
accuracy of the model is :75.0
Vector(m, a, x, D, e, p, t, h,  , 7 maxBin: 35)
accuracy of the model is :75.0
maxDepth: Array[Int] = Array(2, 3, 4, 5, 6, 7)
maxBins: Array[Int] = Array(32, 33, 34, 35)

scala>
```

Based on the results, it appears that the combination of maxDepth=3 and maxBins=32 is yielding the best results, followed by maxDepth=4 and maxBins=33, 34, or 35, and maxDepth=6 and maxBins=33 or 34.

Accuracy:

A cross validator is a technique used in machine learning to evaluate and select the best hyperparameters for a model. The highest accuracy in which this model can predict whether the person has diabetes or not is: 76.45**%**



```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val cvModel_goodnews = cross_validator_goodnews.fit(trainingData)
val predictions_goodnews = cvModel_goodnews.transform(testData)
val accuracy_goodnews = evaluator_goodnews.evaluate(predictions_goodnews)
println("best accuracy of the model is :"+(accuracy_goodnews*100))

// Exiting paste mode, now interpreting.

best accuracy of the model is :76.44927536231883
cvModel_goodnews: org.apache.spark.ml.tuning.CrossValidatorModel = CrossValidatorModel: uid=cv_47a7e6d9ef2b, bestModel=pipeline_ef4ba756f077, numFolds=4
predictions_goodnews: org.apache.spark.sql.DataFrame = [Pregnancies: int, Glucose: int ... 11 more fields]
accuracy_goodnews: Double = 0.7644927536231884

scala>
```

Reference:

Kelleher, J. D., & Tierney, B. (2018). Data Science An Introduction. Boca Raton: Chapman and Hall/CRC.