

Cancer Data (With Logistic Regression)

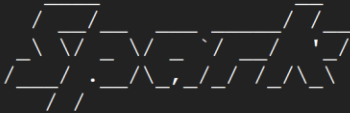
Creating a Hadoop directory and loading the data from the local files system to Hadoop

The screenshot shows a web browser window displaying a Google Cloud Shell session. The address bar at the top contains a long URL starting with "https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m7...". Below the address bar, there's a header area with a gear icon and the text "SSH-in-browser". To the right of this header are icons for uploading, downloading, chat, keyboard shortcuts, and settings. The main area is a dark-themed terminal window. It shows three lines of command input and output:

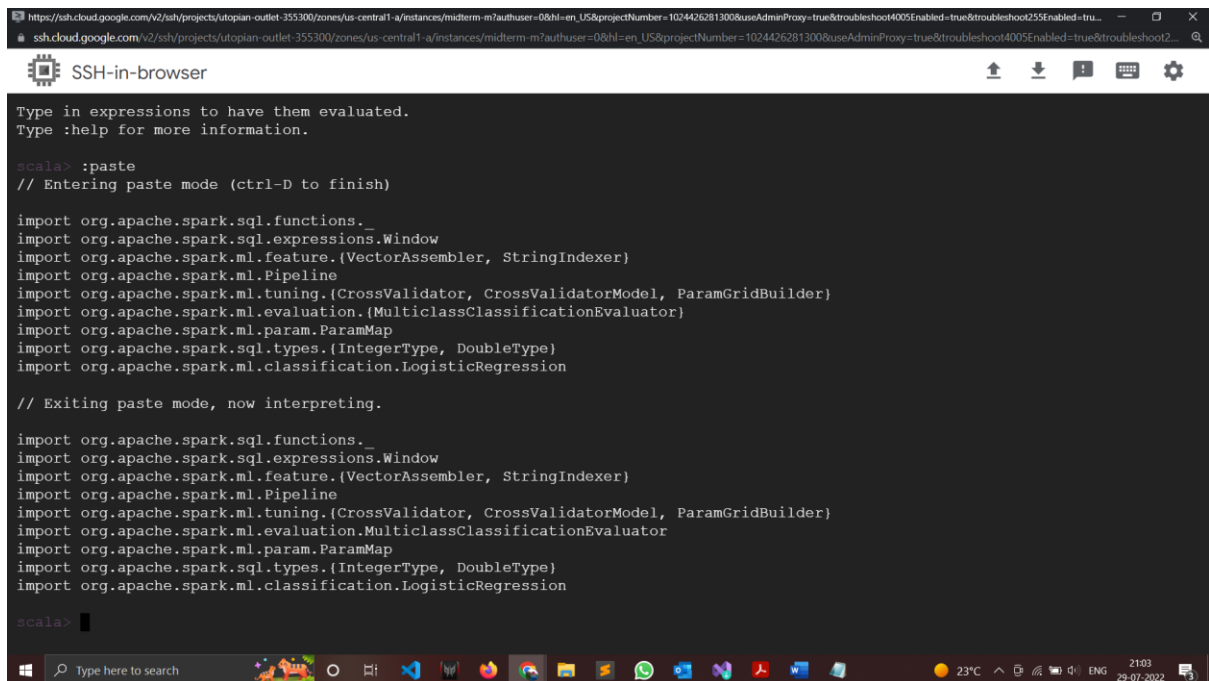
```
harsh88maggo@midterm-m:~$ hadoop fs -mkdir /MidTerm  
mkdir: `/MidTerm': File exists  
harsh88maggo@midterm-m:~$ hadoop fs -copyFromLocal cancer.csv /MidTerm/.  
harsh88maggo@midterm-m:~$
```


At the bottom of the screen, there's a Windows taskbar. It includes a search bar on the left, followed by several application icons like File Explorer, Edge, and various utility apps. On the right side of the taskbar, it displays system information: "23°C", network status, battery level, language set to "ENG", and the time "2100" along with the date "29-07-2022".

Running the Spark Shell

```
https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true...  
# ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true&troubleshoot2...  
harsh88maggo@midterm-m:~$ spark-shell --master yarn  
Setting default log level to "WARN".  
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(new  
22/07/30 00:36:24 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker  
22/07/30 00:36:24 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster  
22/07/30 00:36:24 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat  
22/07/30 00:36:24 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator  
Spark context Web UI available at http://midterm-m.us-centrall1-a.c.utopian-outlet-355300.  
Spark context available as 'sc' (master = yarn, app id = application_1659139248703_000000)  
Spark session available as 'spark'.  
Welcome to  
 version 3.1.3  
Using Scala version 2.12.14 (OpenJDK 64-Bit Server VM, Java 1.8.0_332)  
Type in expressions to have them evaluated.  
Type :help for more information.  
scala>
```

Import statements



The screenshot shows a terminal window titled "SSH-in-browser" with a dark background. The terminal displays Scala code for importing various Spark and MLlib libraries. The code is organized into two sections, each preceded by a comment: "// Entering paste mode (ctrl-D to finish)" and "// Exiting paste mode, now interpreting." The first section contains imports for Spark SQL functions, expressions, ML feature sets, pipelines, tuning, evaluation, and classification. The second section repeats these imports. The terminal window has a standard Linux-style taskbar at the bottom with various application icons and system status information like temperature and time.

```
https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true...
ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot2...

SSH-in-browser

Type in expressions to have them evaluated.
Type :help for more information.

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

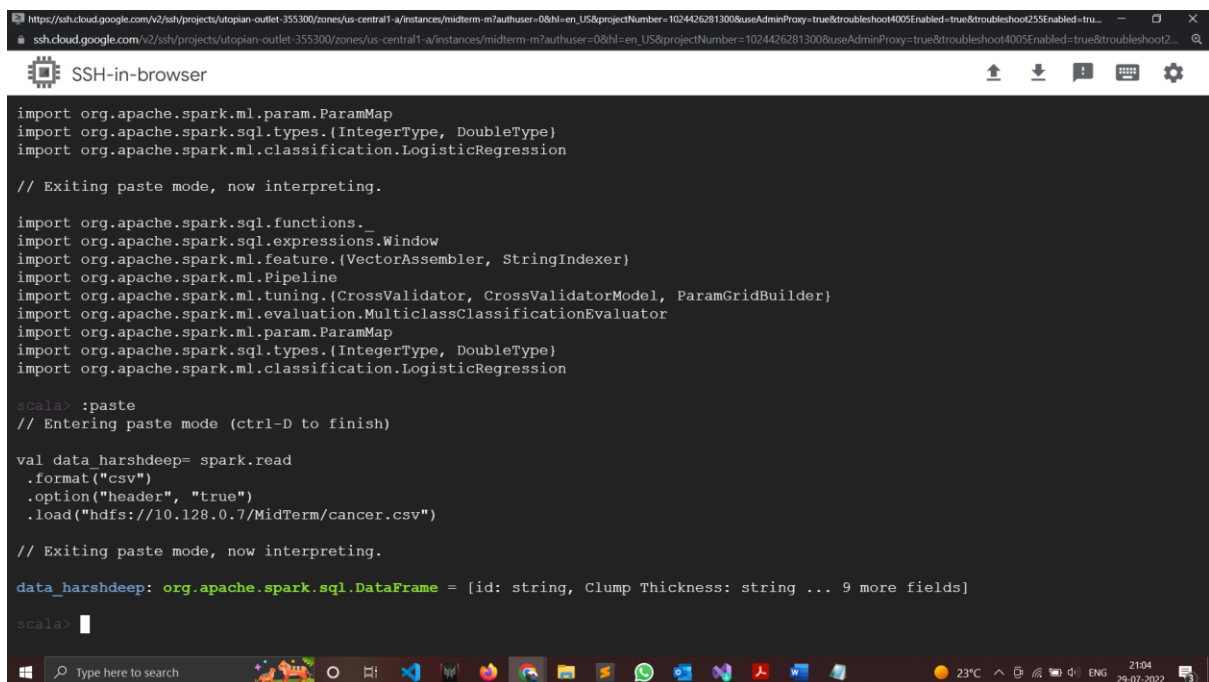
import org.apache.spark.sql.functions._
import org.apache.spark.sql.expressions.Window
import org.apache.spark.ml.feature.{VectorAssembler, StringIndexer}
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.tuning.{CrossValidator, CrossValidatorModel, ParamGridBuilder}
import org.apache.spark.ml.evaluation.{MulticlassClassificationEvaluator}
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.sql.types.{IntegerType, DoubleType}
import org.apache.spark.ml.classification.LogisticRegression

// Exiting paste mode, now interpreting.

import org.apache.spark.sql.functions._
import org.apache.spark.sql.expressions.Window
import org.apache.spark.ml.feature.{VectorAssembler, StringIndexer}
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.tuning.{CrossValidator, CrossValidatorModel, ParamGridBuilder}
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.sql.types.{IntegerType, DoubleType}
import org.apache.spark.ml.classification.LogisticRegression

scala>
```

Reading the CSV file



The screenshot shows a terminal window titled "SSH-in-browser" with a dark background. The terminal displays Scala code for reading a CSV file into a Spark DataFrame. The code is organized into two sections, each preceded by a comment: "// Entering paste mode (ctrl-D to finish)" and "// Exiting paste mode, now interpreting." The first section contains imports for Spark ML param, Spark SQL types, and Spark ML classification. The second section contains imports for Spark SQL functions, expressions, ML feature sets, pipelines, tuning, evaluation, and classification. The code then uses the `spark.read` method to load a CSV file from HDFS, specifying the format as "csv", the header as "true", and the file path as "hdfs://10.128.0.7/MidTerm/cancer.csv". The result is assigned to a variable named `data_harshdeep`. The terminal window has a standard Linux-style taskbar at the bottom with various application icons and system status information like temperature and time.

```
https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true...
ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot2...

SSH-in-browser

import org.apache.spark.ml.param.ParamMap
import org.apache.spark.sql.types.{IntegerType, DoubleType}
import org.apache.spark.ml.classification.LogisticRegression

// Exiting paste mode, now interpreting.

import org.apache.spark.sql.functions._
import org.apache.spark.sql.expressions.Window
import org.apache.spark.ml.feature.{VectorAssembler, StringIndexer}
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.tuning.{CrossValidator, CrossValidatorModel, ParamGridBuilder}
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.sql.types.{IntegerType, DoubleType}
import org.apache.spark.ml.classification.LogisticRegression

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

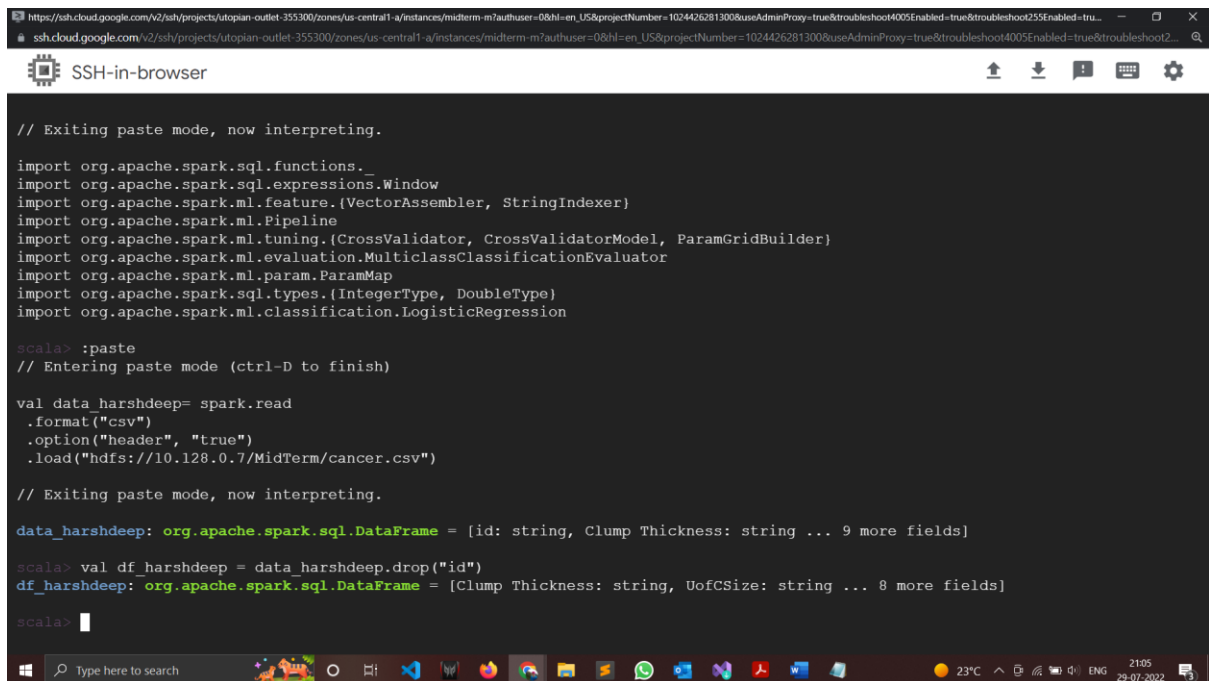
val data_harshdeep= spark.read
  .format("csv")
  .option("header", "true")
  .load("hdfs://10.128.0.7/MidTerm/cancer.csv")

// Exiting paste mode, now interpreting.

data_harshdeep: org.apache.spark.sql.DataFrame = [id: string, Clump Thickness: string ... 9 more fields]

scala>
```

Dropping the 'ID' column since it won't help us in prediction



```
https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true...
SSH-in-browser

// Exiting paste mode, now interpreting.

import org.apache.spark.sql.functions._
import org.apache.spark.sql.expressions.Window
import org.apache.spark.ml.feature.{VectorAssembler, StringIndexer}
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.tuning.{CrossValidator, CrossValidatorModel, ParamGridBuilder}
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.sql.types.{IntegerType, DoubleType}
import org.apache.spark.ml.classification.LogisticRegression

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

val data_harshdeep= spark.read
  .format("csv")
  .option("header", "true")
  .load("hdfs://10.128.0.7/MidTerm/cancer.csv")

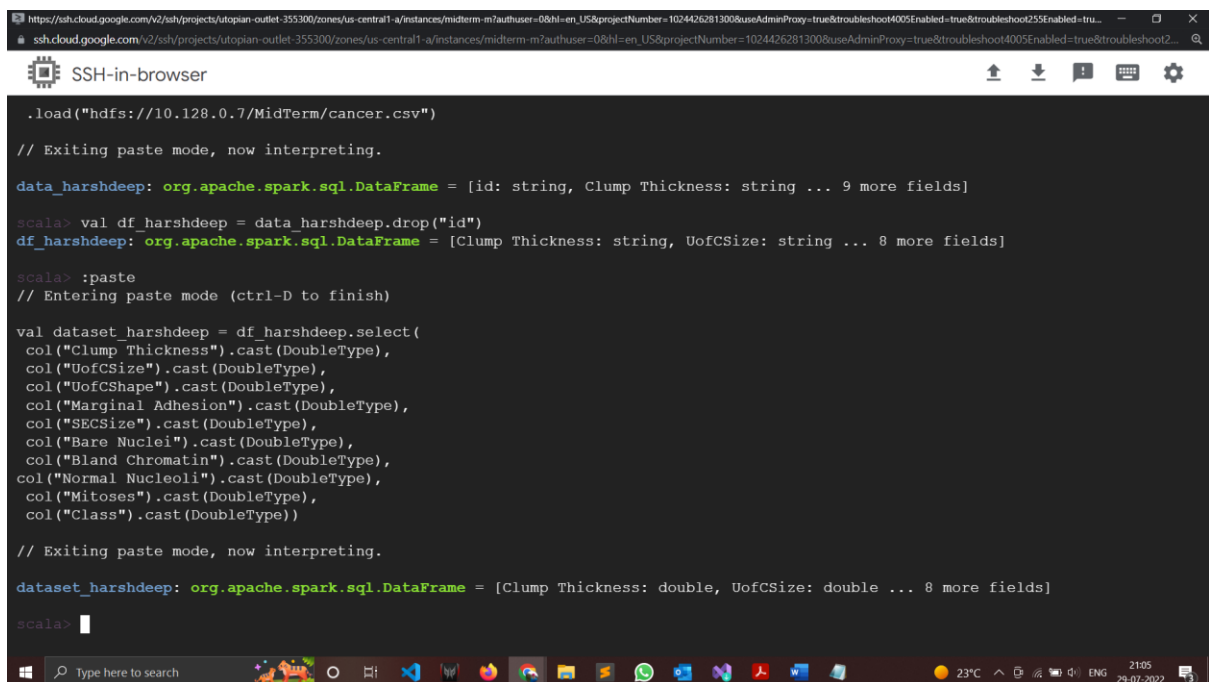
// Exiting paste mode, now interpreting.

data_harshdeep: org.apache.spark.sql.DataFrame = [id: string, Clump Thickness: string ... 9 more fields]

scala> val df_harshdeep = data_harshdeep.drop("id")
df_harshdeep: org.apache.spark.sql.DataFrame = [Clump Thickness: string, UofCSize: string ... 8 more fields]

scala>
```

Typecasting the data into type Double for our model training



```
https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true...
SSH-in-browser

.load("hdfs://10.128.0.7/MidTerm/cancer.csv")

// Exiting paste mode, now interpreting.

data_harshdeep: org.apache.spark.sql.DataFrame = [id: string, Clump Thickness: string ... 9 more fields]

scala> val df_harshdeep = data_harshdeep.drop("id")
df_harshdeep: org.apache.spark.sql.DataFrame = [Clump Thickness: string, UofCSize: string ... 8 more fields]

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

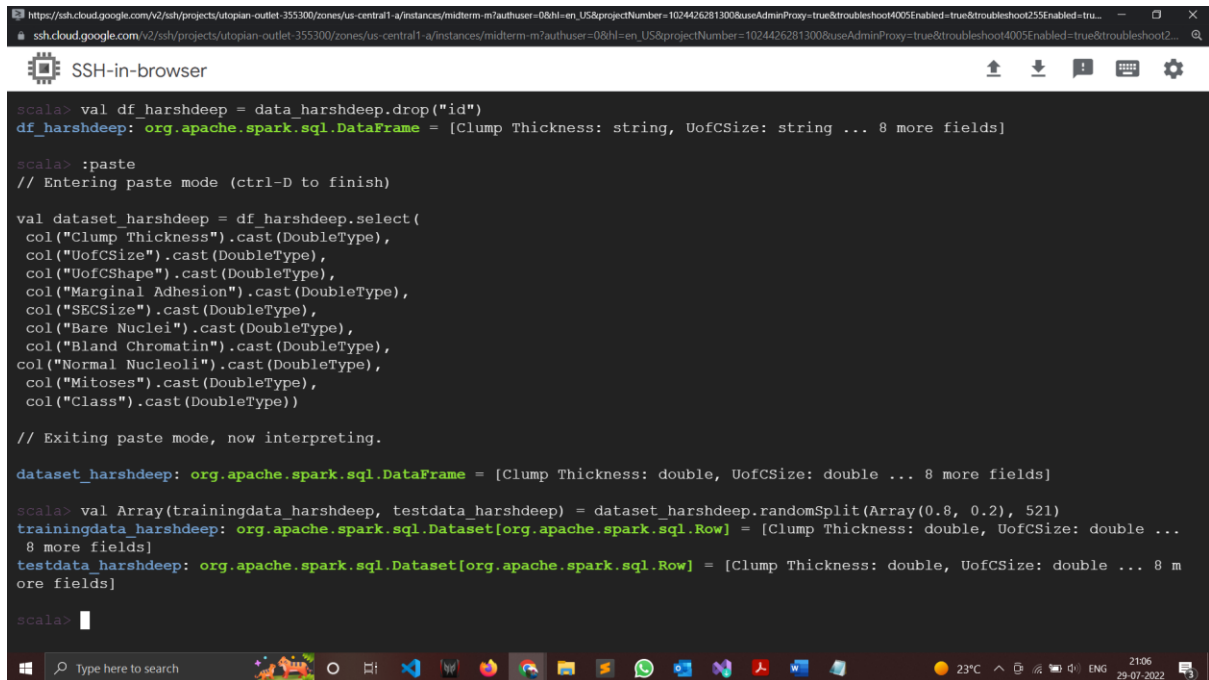
val dataset_harshdeep = df_harshdeep.select(
  col("Clump Thickness").cast(DoubleType),
  col("UofCSize").cast(DoubleType),
  col("UofCShape").cast(DoubleType),
  col("Marginal Adhesion").cast(DoubleType),
  col("SECSIZE").cast(DoubleType),
  col("Bare Nuclei").cast(DoubleType),
  col("Bland Chromatin").cast(DoubleType),
  col("Normal Nucleoli").cast(DoubleType),
  col("Mitoses").cast(DoubleType),
  col("Class").cast(DoubleType)
)

// Exiting paste mode, now interpreting.

dataset_harshdeep: org.apache.spark.sql.DataFrame = [Clump Thickness: double, UofCSize: double ... 8 more fields]

scala>
```

Split the dataset into train and test



```
scala> val df_harshdeep = data_harshdeep.drop("id")
df_harshdeep: org.apache.spark.sql.DataFrame = [Clump Thickness: string, UofCSize: string ... 8 more fields]

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

val dataset_harshdeep = df_harshdeep.select(
  col("Clump Thickness").cast(DoubleType),
  col("UofCSize").cast(DoubleType),
  col("UofCShape").cast(DoubleType),
  col("Marginal Adhesion").cast(DoubleType),
  col("SECSIZE").cast(DoubleType),
  col("Bare Nuclei").cast(DoubleType),
  col("Bland Chromatin").cast(DoubleType),
  col("Normal Nucleoli").cast(DoubleType),
  col("Mitoses").cast(DoubleType),
  col("Class").cast(DoubleType)
)

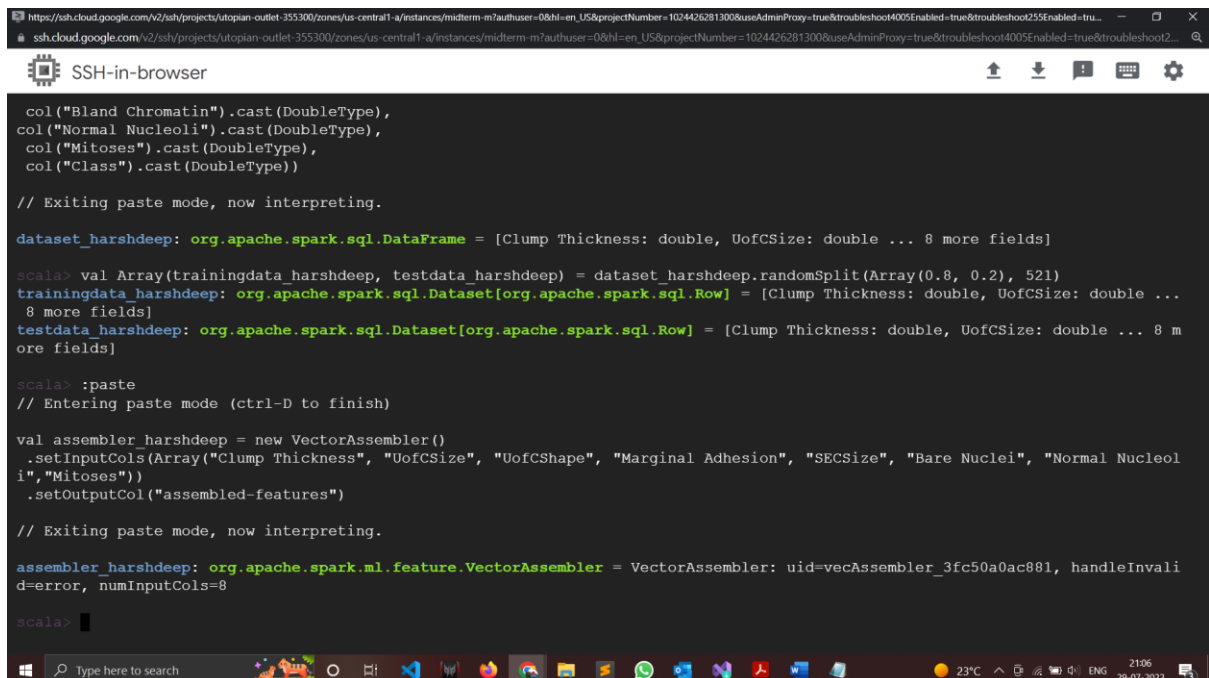
// Exiting paste mode, now interpreting.

dataset_harshdeep: org.apache.spark.sql.DataFrame = [Clump Thickness: double, UofCSize: double ... 8 more fields]

scala> val Array(trainingdata_harshdeep, testdata_harshdeep) = dataset_harshdeep.randomSplit(Array(0.8, 0.2), 521)
trainingdata_harshdeep: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Clump Thickness: double, UofCSize: double ... 8 more fields]
testdata_harshdeep: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Clump Thickness: double, UofCSize: double ... 8 more fields]

scala>
```

Assembling the features using VectorAssembler



```
col("Bland Chromatin").cast(DoubleType),
col("Normal Nucleoli").cast(DoubleType),
col("Mitoses").cast(DoubleType),
col("Class").cast(DoubleType)

// Exiting paste mode, now interpreting.

dataset_harshdeep: org.apache.spark.sql.DataFrame = [Clump Thickness: double, UofCSize: double ... 8 more fields]

scala> val Array(trainingdata_harshdeep, testdata_harshdeep) = dataset_harshdeep.randomSplit(Array(0.8, 0.2), 521)
trainingdata_harshdeep: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Clump Thickness: double, UofCSize: double ... 8 more fields]
testdata_harshdeep: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Clump Thickness: double, UofCSize: double ... 8 more fields]

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

val assembler_harshdeep = new VectorAssembler()
  .setInputCols(Array("Clump Thickness", "UofCSize", "UofCShape", "Marginal Adhesion", "SECSIZE", "Bare Nuclei", "Normal Nucleoli", "Mitoses"))
  .setOutputCol("assembled-features")

// Exiting paste mode, now interpreting.

assembler_harshdeep: org.apache.spark.ml.feature.VectorAssembler = VectorAssembler: uid=vecAssembler_3fc50a0ac881, handleInvalid=error, numInputCols=8

scala>
```

Creating the Logistic Regression Object and passing the features

```
https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true...
ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot2...

SSH-in-browser

8 more fields]
testdata_harshdeep: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Clump Thickness: double, UofCSize: double ... 8 m
ore fields]

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

val assembler_harshdeep = new VectorAssembler()
  .setInputCols(Array("Clump Thickness", "UofCSize", "UofCShape", "Marginal Adhesion", "SECSize", "Bare Nuclei", "Normal Nucleol
i", "Mitoses"))
  .setOutputCol("assembled-features")

// Exiting paste mode, now interpreting.

assembler_harshdeep: org.apache.spark.ml.feature.VectorAssembler = VectorAssembler: uid=vecAssembler_3fc50a0ac881, handleInvali
d-error, numInputCols=8

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

val lr_harshdeep = new LogisticRegression()
  .setFeaturesCol("assembled-features")
  .setLabelCol("Class")

// Exiting paste mode, now interpreting.

lr_harshdeep: org.apache.spark.ml.classification.LogisticRegression = logreg_0cc145354c9e

scala>
```

Creating the pipeling

```
https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true...
ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot2...

SSH-in-browser

.setOutputCol("assembled-features")

// Exiting paste mode, now interpreting.

assembler_harshdeep: org.apache.spark.ml.feature.VectorAssembler = VectorAssembler: uid=vecAssembler_3fc50a0ac881, handleInvali
d-error, numInputCols=8

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

val lr_harshdeep = new LogisticRegression()
  .setFeaturesCol("assembled-features")
  .setLabelCol("Class")

// Exiting paste mode, now interpreting.

lr_harshdeep: org.apache.spark.ml.classification.LogisticRegression = logreg_0cc145354c9e

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

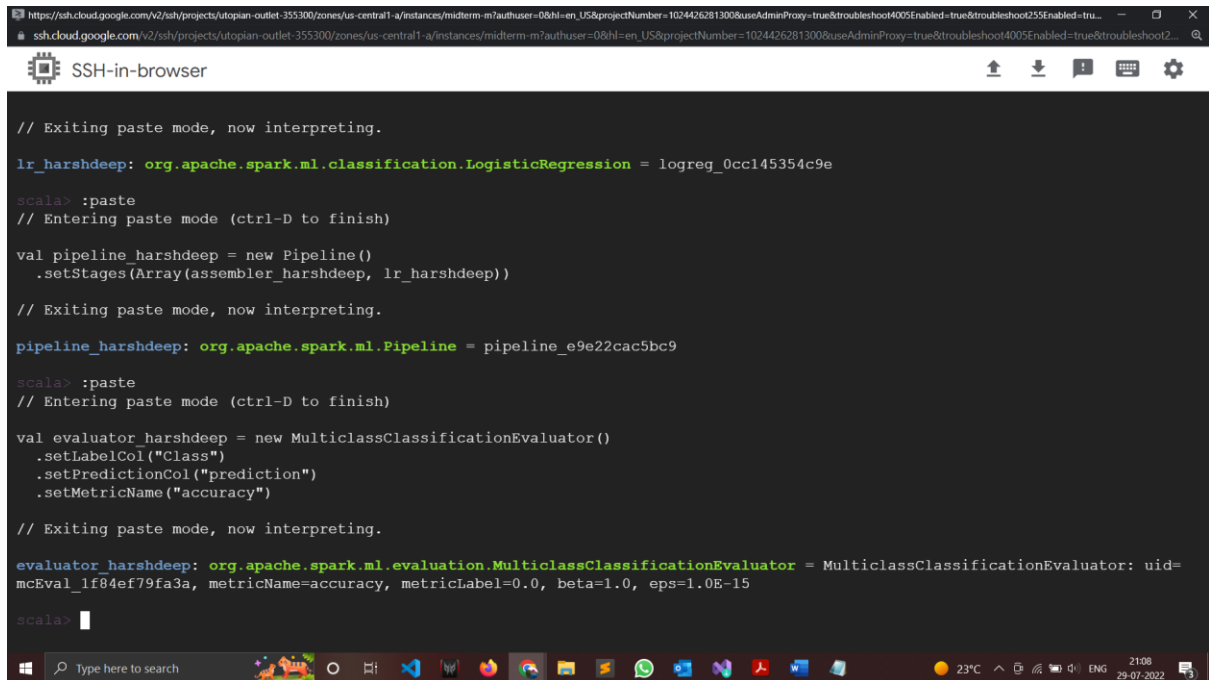
val pipeline_harshdeep = new Pipeline()
  .setStages(Array(assembler_harshdeep, lr_harshdeep))

// Exiting paste mode, now interpreting.

pipeline_harshdeep: org.apache.spark.ml.Pipeline = pipeline_e9e22cac5bc9

scala>
```

Evaluator for our model



```
https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m7authuser=08hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true...
ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m7authuser=08hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true...

SSH-in-browser

// Exiting paste mode, now interpreting.

lr_harshdeep: org.apache.spark.ml.classification.LogisticRegression = logreg_0cc145354c9e

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

val pipeline_harshdeep = new Pipeline()
  .setStages(Array(assembler_harshdeep, lr_harshdeep))

// Exiting paste mode, now interpreting.

pipeline_harshdeep: org.apache.spark.ml.Pipeline = pipeline_e9e22cac5bc9

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

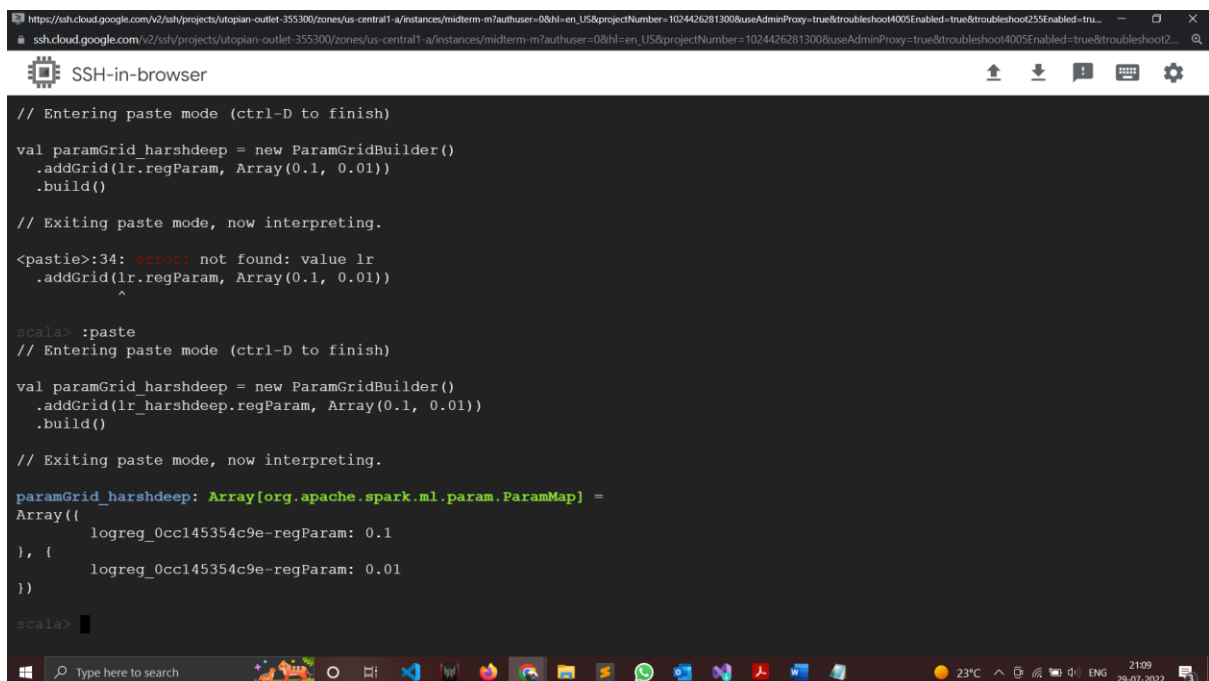
val evaluator_harshdeep = new MulticlassClassificationEvaluator()
  .setLabelCol("Class")
  .setPredictionCol("prediction")
  .setMetricName("accuracy")

// Exiting paste mode, now interpreting.

evaluator_harshdeep: org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator = MulticlassClassificationEvaluator: uid=mcEval_1f84ef79fa3a, metricName=accuracy, metricLabel=0.0, beta=1.0, eps=1.0E-15

scala> 
```

Setting the hyperparameters



```
https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m7authuser=08hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true...
ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m7authuser=08hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true...

SSH-in-browser

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

val paramGrid_harshdeep = new ParamGridBuilder()
  .addGrid(lr.regParam, Array(0.1, 0.01))
  .build()

// Exiting paste mode, now interpreting.

<paste>:34: error: not found: value lr
  .addGrid(lr.regParam, Array(0.1, 0.01))
          ^

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

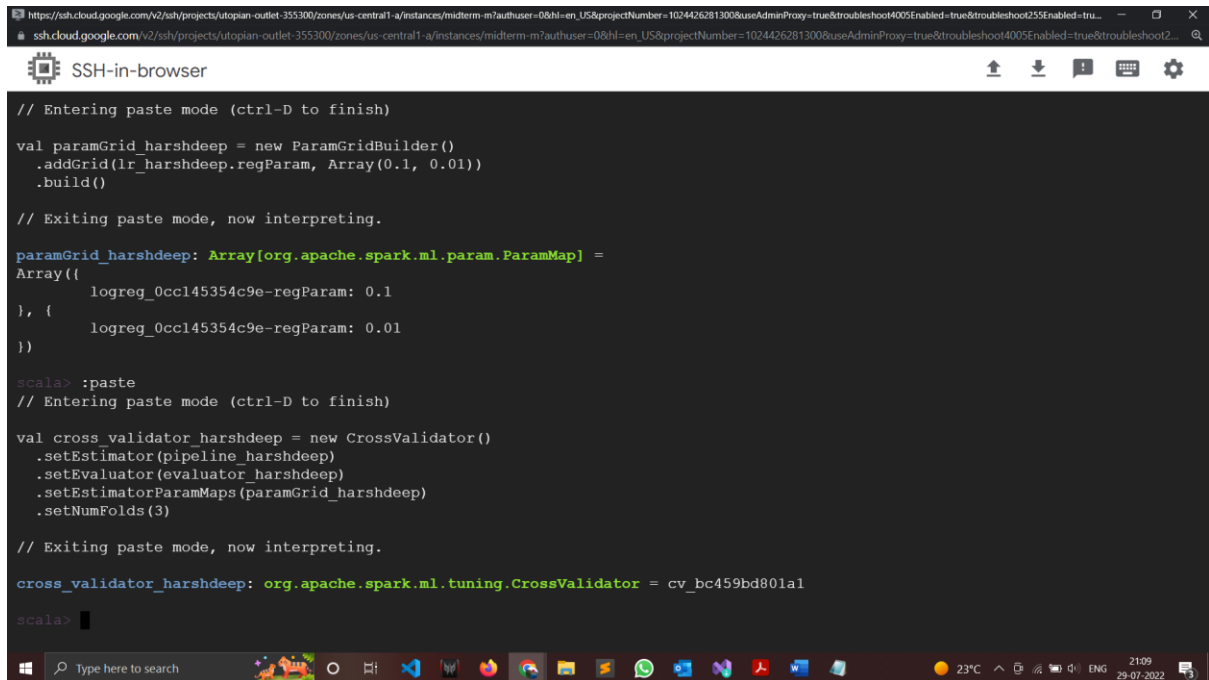
val paramGrid_harshdeep = new ParamGridBuilder()
  .addGrid(lr_harshdeep.regParam, Array(0.1, 0.01))
  .build()

// Exiting paste mode, now interpreting.

paramGrid_harshdeep: Array[org.apache.spark.ml.param.ParamMap] =
Array({
  logreg_0cc145354c9e-regParam: 0.1
}, {
  logreg_0cc145354c9e-regParam: 0.01
})

scala> 
```

Creating the Cross Validator



```
https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true...
ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true...

SSH-in-browser

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

val paramGrid_harshdeep = new ParamGridBuilder()
  .addGrid(lr_harshdeep.regParam, Array(0.1, 0.01))
  .build()

// Exiting paste mode, now interpreting.

paramGrid_harshdeep: Array[org.apache.spark.ml.param.ParamMap] =
Array((
  logreg_0cc145354c9e-regParam: 0.1
), {
  logreg_0cc145354c9e-regParam: 0.01
})

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

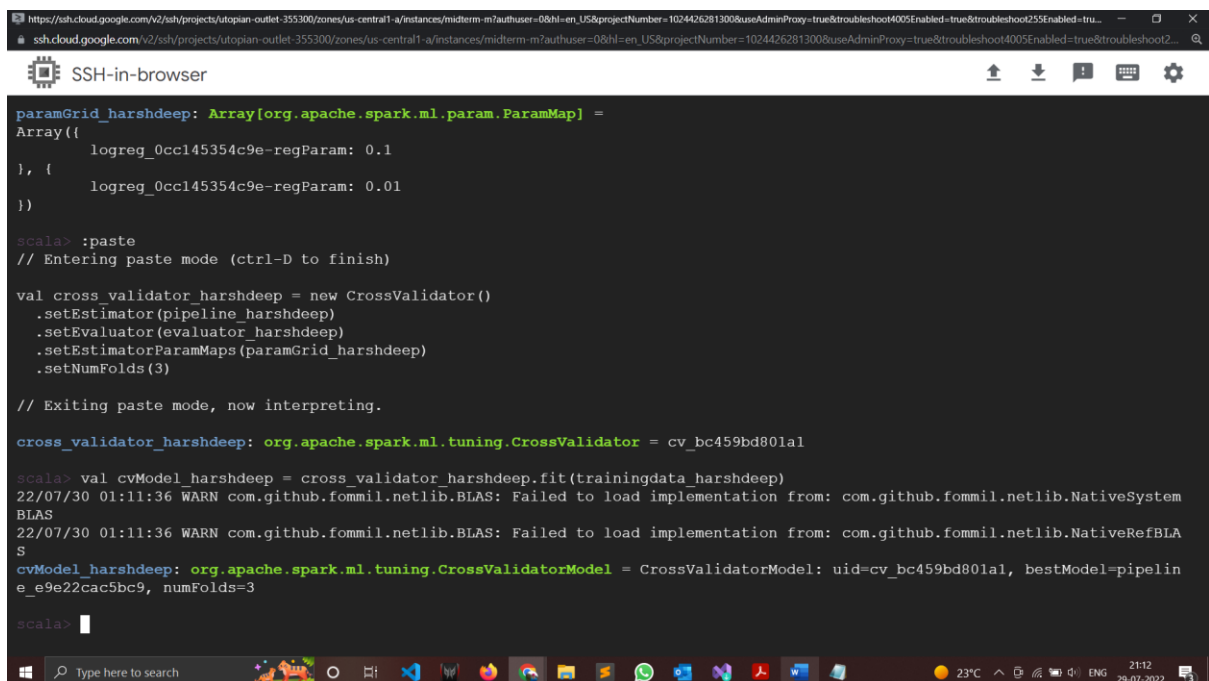
val cross_validator_harshdeep = new CrossValidator()
  .setEstimator(pipeline_harshdeep)
  .setEvaluator(evaluator_harshdeep)
  .setEstimatorParamMaps(paramGrid_harshdeep)
  .setNumFolds(3)

// Exiting paste mode, now interpreting.

cross_validator_harshdeep: org.apache.spark.ml.tuning.CrossValidator = cv_bc459bd801a1

scala>
```

Training our model



```
https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true...
ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true...

SSH-in-browser

paramGrid_harshdeep: Array[org.apache.spark.ml.param.ParamMap] =
Array((
  logreg_0cc145354c9e-regParam: 0.1
), {
  logreg_0cc145354c9e-regParam: 0.01
})

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

val cross_validator_harshdeep = new CrossValidator()
  .setEstimator(pipeline_harshdeep)
  .setEvaluator(evaluator_harshdeep)
  .setEstimatorParamMaps(paramGrid_harshdeep)
  .setNumFolds(3)

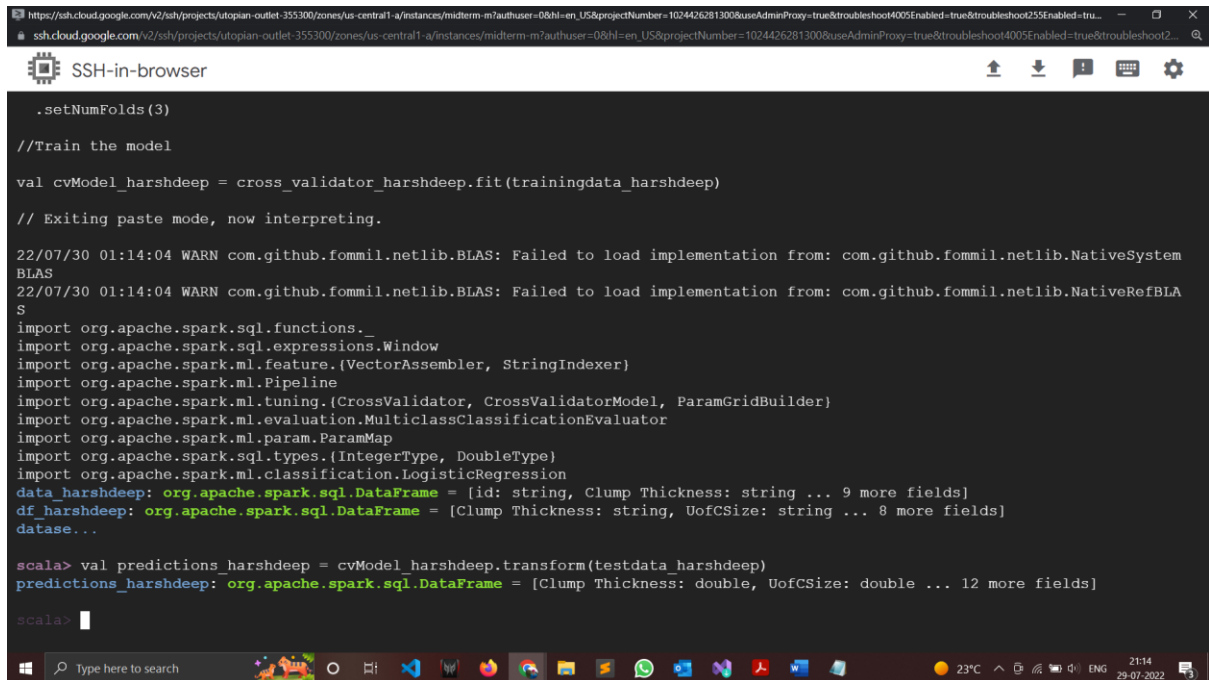
// Exiting paste mode, now interpreting.

cross_validator_harshdeep: org.apache.spark.ml.tuning.CrossValidator = cv_bc459bd801a1

scala> val cvModel_harshdeep = cross_validator_harshdeep.fit(trainingdata_harshdeep)
22/07/30 01:11:36 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystem
BLAS
22/07/30 01:11:36 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLA
S
cvModel_harshdeep: org.apache.spark.ml.tuning.CrossValidatorModel = CrossValidatorModel: uid=cv_bc459bd801a1, bestModel=pipelin
e_e9e22cac5bc9, numFolds=3

scala>
```


Prediction using testdata



```
https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true...
ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true...

SSH-in-browser

.setNumFolds(3)

//Train the model

val cvModel_harshdeep = cross_validator_harshdeep.fit(trainingdata_harshdeep)

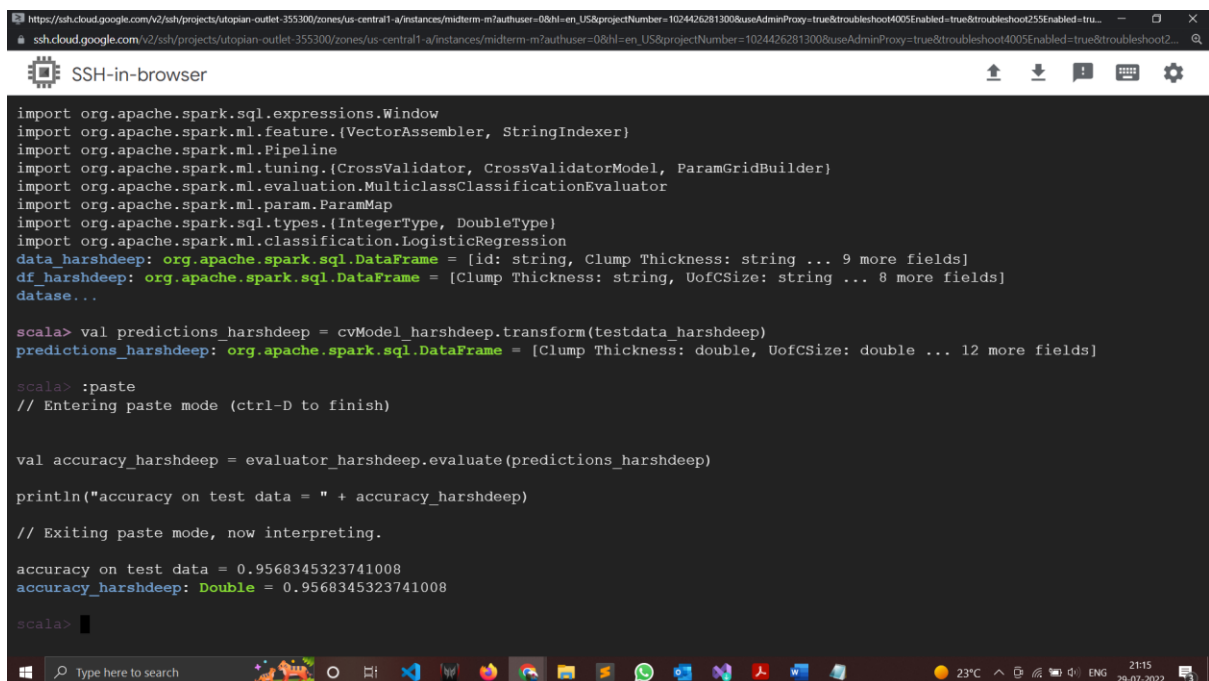
// Exiting paste mode, now interpreting.

22/07/30 01:14:04 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystem
BLAS
22/07/30 01:14:04 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLA
S
import org.apache.spark.sql.functions._
import org.apache.spark.sql.expressions.Window
import org.apache.spark.ml.feature.{VectorAssembler, StringIndexer}
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.tuning.{CrossValidator, CrossValidatorModel, ParamGridBuilder}
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.sql.types.{IntegerType, DoubleType}
import org.apache.spark.ml.classification.LogisticRegression
data_harshdeep: org.apache.spark.sql.DataFrame = [id: string, Clump Thickness: string ... 9 more fields]
df_harshdeep: org.apache.spark.sql.DataFrame = [Clump Thickness: string, UofCSize: string ... 8 more fields]
dataset...

scala> val predictions_harshdeep = cvModel_harshdeep.transform(testdata_harshdeep)
predictions_harshdeep: org.apache.spark.sql.DataFrame = [Clump Thickness: double, UofCSize: double ... 12 more fields]

scala>
```

Evaluating the performance of our model



```
https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true...
ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true...

SSH-in-browser

import org.apache.spark.sql.expressions.Window
import org.apache.spark.ml.feature.{VectorAssembler, StringIndexer}
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.tuning.{CrossValidator, CrossValidatorModel, ParamGridBuilder}
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.sql.types.{IntegerType, DoubleType}
import org.apache.spark.ml.classification.LogisticRegression
data_harshdeep: org.apache.spark.sql.DataFrame = [id: string, Clump Thickness: string ... 9 more fields]
df_harshdeep: org.apache.spark.sql.DataFrame = [Clump Thickness: string, UofCSize: string ... 8 more fields]
dataset...

scala> val predictions_harshdeep = cvModel_harshdeep.transform(testdata_harshdeep)
predictions_harshdeep: org.apache.spark.sql.DataFrame = [Clump Thickness: double, UofCSize: double ... 12 more fields]

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

val accuracy_harshdeep = evaluator_harshdeep.evaluate(predictions_harshdeep)

println("accuracy on test data = " + accuracy_harshdeep)

// Exiting paste mode, now interpreting.

accuracy on test data = 0.9568345323741008
accuracy_harshdeep: Double = 0.9568345323741008

scala>
```

ACCURACY: 95.68%