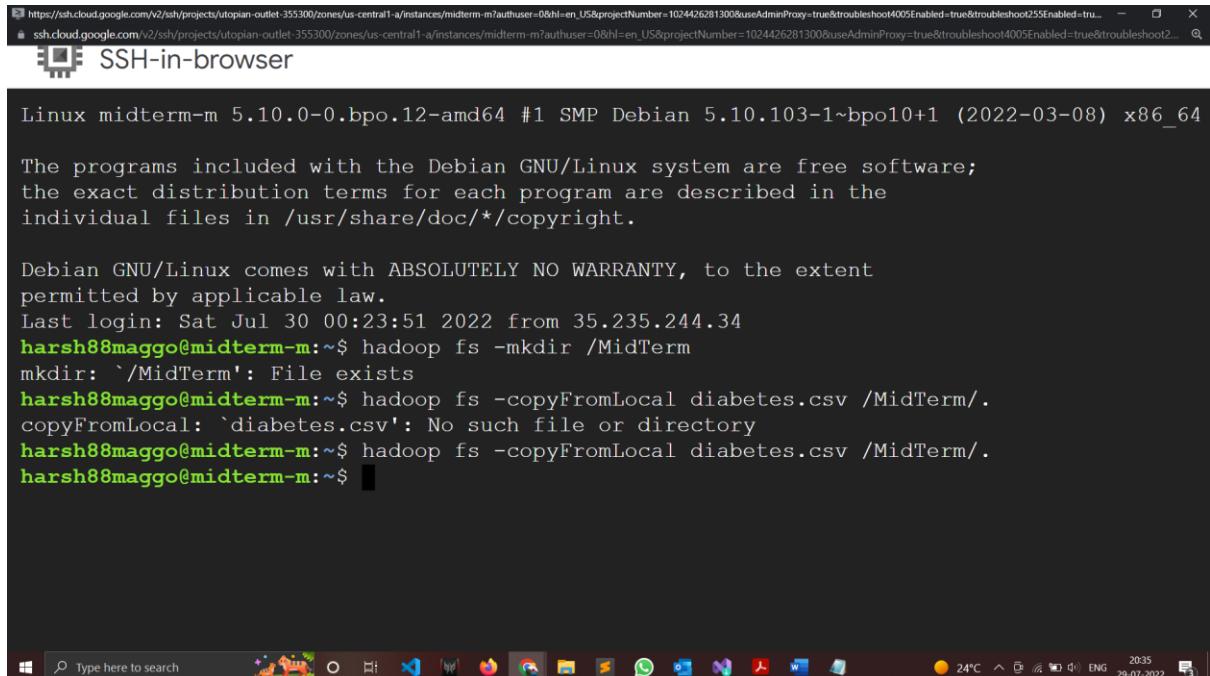


# PROJECT EXECUTION REPORT

## Problem 1: Diabetes Data (With Random Forest Classification)

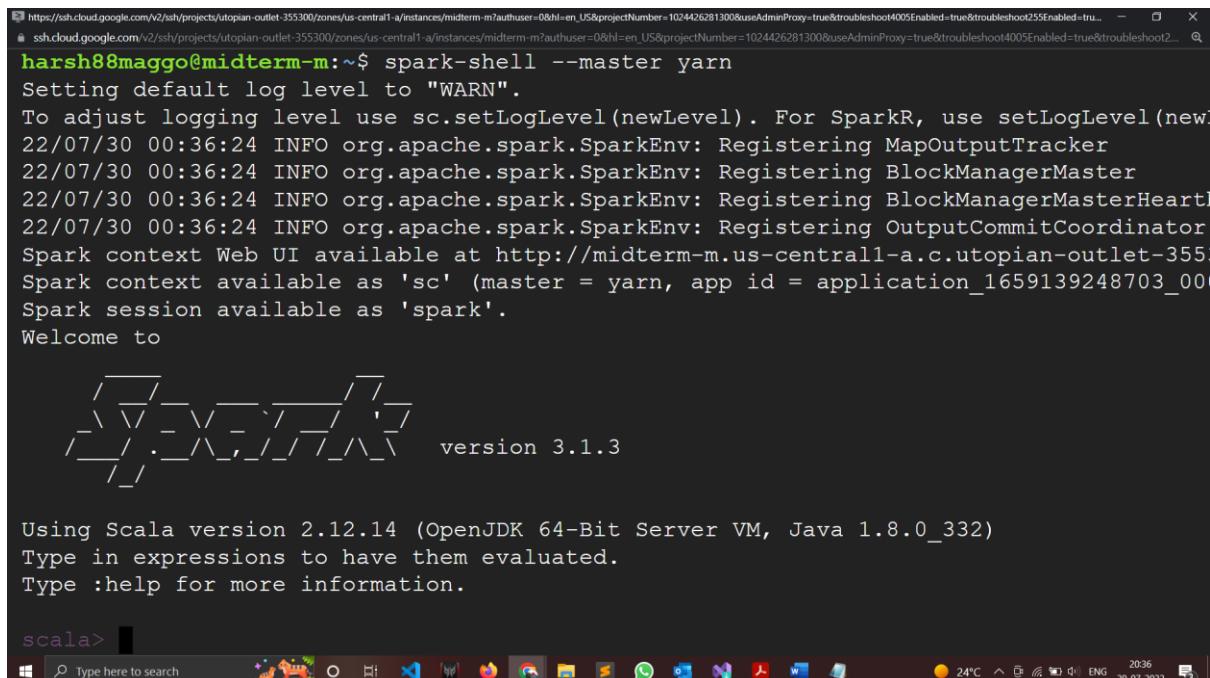
Creating a Hadoop directory and loading the data from the local files system to Hadoop



```
Linux midterm-m 5.10.0-0.bpo.12-amd64 #1 SMP Debian 5.10.103-1~bpo10+1 (2022-03-08) 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 Jul 30 00:23:51 2022 from 35.235.244.34
harsh88mago@midterm-m:~$ hadoop fs -mkdir /MidTerm
mkdir: `/MidTerm': File exists
harsh88mago@midterm-m:~$ hadoop fs -copyFromLocal diabetes.csv /MidTerm/.
copyFromLocal: `diabetes.csv': No such file or directory
harsh88mago@midterm-m:~$ hadoop fs -copyFromLocal diabetes.csv /MidTerm/.
harsh88mago@midterm-m:~$
```

Running the Spark Shell



```
harsh88mago@midterm-m:~$ spark-shell --master yarn
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
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-central1-a.c.utoptian-outlet-355300:4040
Spark context available as 'sc' (master = yarn, app id = application_1659139248703_0001).
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.

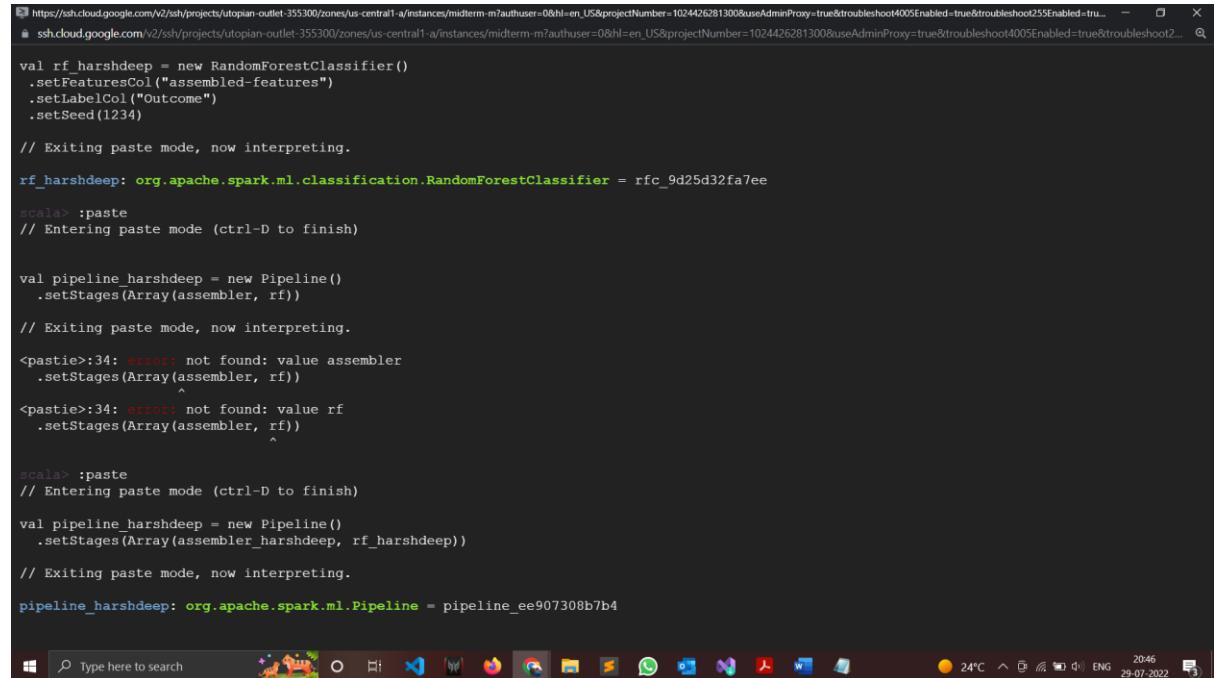
scala>
```





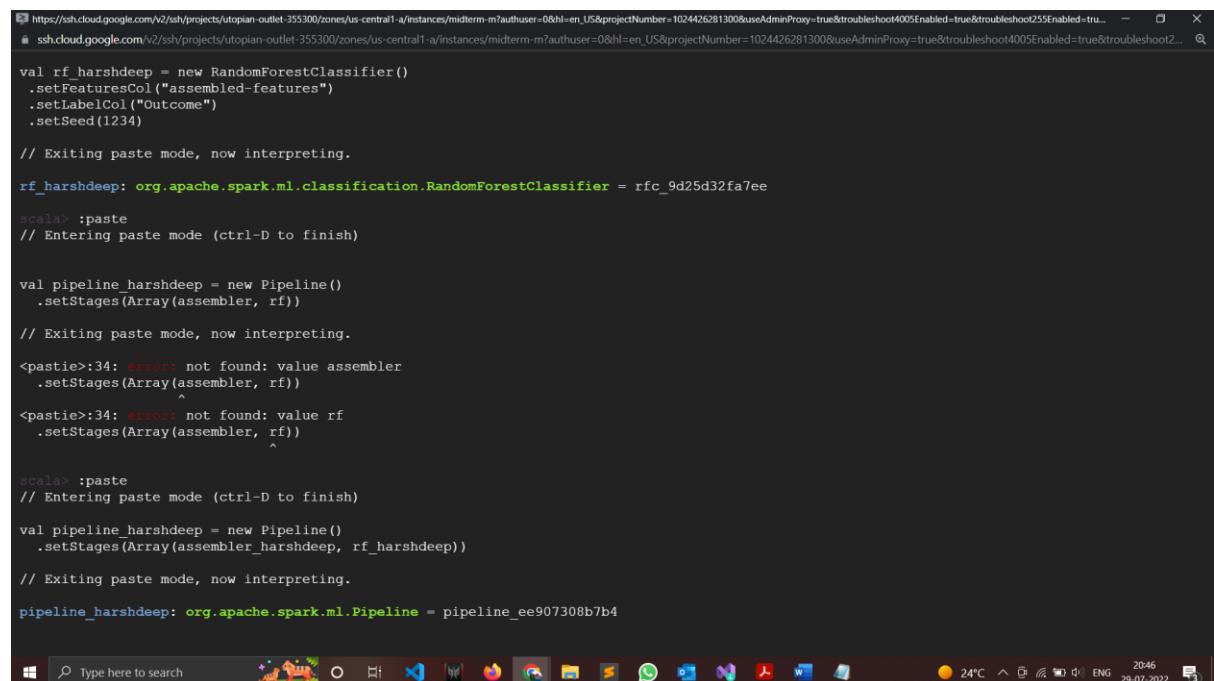


## Creating the Random Forest Object and passing the features



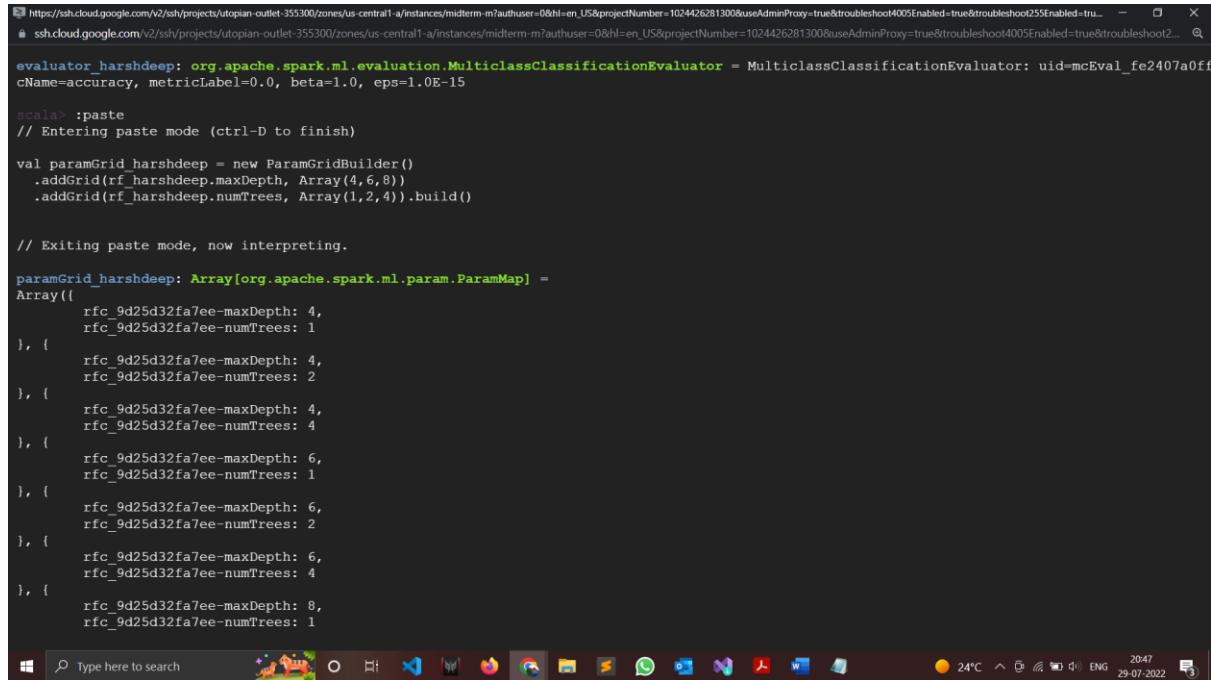
```
https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true  
ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot2...  
  
val rf_harshdeep = new RandomForestClassifier()  
.setFeaturesCol("assembled-features")  
.setLabelCol("Outcome")  
.setSeed(1234)  
  
// Exiting paste mode, now interpreting.  
  
rf_harshdeep: org.apache.spark.ml.classification.RandomForestClassifier = rfc_9d25d32fa7ee  
  
scala> :paste  
// Entering paste mode (ctrl-D to finish)  
  
val pipeline_harshdeep = new Pipeline()  
.setStages(Array(assembler, rf))  
  
// Exiting paste mode, now interpreting.  
  
<pastie>:34: error: not found: value assembler  
.setStages(Array(assembler, rf))  
^  
<pastie>:34: error: not found: value rf  
.setStages(Array(assembler, rf))  
^  
  
scala> :paste  
// Entering paste mode (ctrl-D to finish)  
  
val pipeline_harshdeep = new Pipeline()  
.setStages(Array(assembler_harshdeep, rf_harshdeep))  
  
// Exiting paste mode, now interpreting.  
  
pipeline_harshdeep: org.apache.spark.ml.Pipeline = pipeline_ee907308b7b4
```

## 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...  
  
val rf_harshdeep = new RandomForestClassifier()  
.setFeaturesCol("assembled-features")  
.setLabelCol("Outcome")  
.setSeed(1234)  
  
// Exiting paste mode, now interpreting.  
  
rf_harshdeep: org.apache.spark.ml.classification.RandomForestClassifier = rfc_9d25d32fa7ee  
  
scala> :paste  
// Entering paste mode (ctrl-D to finish)  
  
val pipeline_harshdeep = new Pipeline()  
.setStages(Array(assembler, rf))  
  
// Exiting paste mode, now interpreting.  
  
<pastie>:34: error: not found: value assembler  
.setStages(Array(assembler, rf))  
^  
<pastie>:34: error: not found: value rf  
.setStages(Array(assembler, rf))  
^  
  
scala> :paste  
// Entering paste mode (ctrl-D to finish)  
  
val pipeline_harshdeep = new Pipeline()  
.setStages(Array(assembler_harshdeep, rf_harshdeep))  
  
// Exiting paste mode, now interpreting.  
  
pipeline_harshdeep: org.apache.spark.ml.Pipeline = pipeline_ee907308b7b4
```

## Evaluator for our model



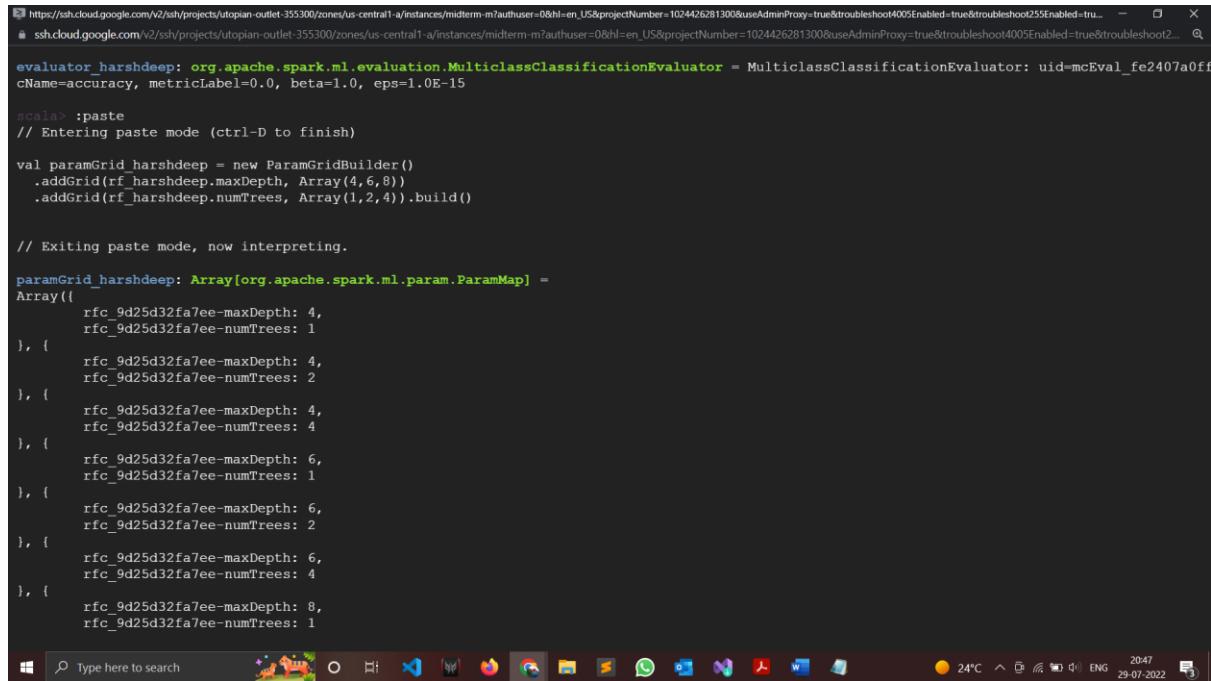
```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val paramGrid harshdeep = new ParamGridBuilder()
  .addGrid(rf_harshdeep.maxDepth, Array(4,6,8))
  .addGrid(rf_harshdeep.numTrees, Array(1,2,4)).build()

// Exiting paste mode, now interpreting.

paramGrid_harshdeep: Array[org.apache.spark.ml.param.ParamMap] =
Array([
  rfc_9d25d32fa7ee-maxDepth: 4,
  rfc_9d25d32fa7ee-numTrees: 1
), [
  rfc_9d25d32fa7ee-maxDepth: 4,
  rfc_9d25d32fa7ee-numTrees: 2
), [
  rfc_9d25d32fa7ee-maxDepth: 4,
  rfc_9d25d32fa7ee-numTrees: 4
), [
  rfc_9d25d32fa7ee-maxDepth: 6,
  rfc_9d25d32fa7ee-numTrees: 1
), [
  rfc_9d25d32fa7ee-maxDepth: 6,
  rfc_9d25d32fa7ee-numTrees: 2
), [
  rfc_9d25d32fa7ee-maxDepth: 6,
  rfc_9d25d32fa7ee-numTrees: 4
), [
  rfc_9d25d32fa7ee-maxDepth: 8,
  rfc_9d25d32fa7ee-numTrees: 1
```

## Setting the hyperparameters



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

val paramGrid harshdeep = new ParamGridBuilder()
  .addGrid(rf_harshdeep.maxDepth, Array(4,6,8))
  .addGrid(rf_harshdeep.numTrees, Array(1,2,4)).build()

// Exiting paste mode, now interpreting.

paramGrid_harshdeep: Array[org.apache.spark.ml.param.ParamMap] =
Array([
  rfc_9d25d32fa7ee-maxDepth: 4,
  rfc_9d25d32fa7ee-numTrees: 1
), [
  rfc_9d25d32fa7ee-maxDepth: 4,
  rfc_9d25d32fa7ee-numTrees: 2
), [
  rfc_9d25d32fa7ee-maxDepth: 4,
  rfc_9d25d32fa7ee-numTrees: 4
), [
  rfc_9d25d32fa7ee-maxDepth: 6,
  rfc_9d25d32fa7ee-numTrees: 1
), [
  rfc_9d25d32fa7ee-maxDepth: 6,
  rfc_9d25d32fa7ee-numTrees: 2
), [
  rfc_9d25d32fa7ee-maxDepth: 6,
  rfc_9d25d32fa7ee-numTrees: 4
), [
  rfc_9d25d32fa7ee-maxDepth: 8,
  rfc_9d25d32fa7ee-numTrees: 1
```

## 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
sh.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

}, {
    rfc_9d25d32fa7ee-maxDepth: 4,
    rfc_9d25d32fa7ee-numTrees: 4
}, {
    rfc_9d25d32fa7ee-maxDepth: 6,
    rfc_9d25d32fa7ee-numTrees: 1
}, {
    rfc_9d25d32fa7ee-maxDepth: 6,
    rfc_9d25d32fa7ee-numTrees: 2
}, {
    rfc_9d25d32fa7ee-maxDepth: 6,
    rfc_9d25d32fa7ee-numTrees: 4
}, {
    rfc_9d25d32fa7ee-maxDepth: 8,
    rfc_9d25d32fa7ee-numTrees: 1
}, {
    rfc_9d25d32fa7ee-maxDepth: 8,
    rfc_9d25d32fa7ee-numTrees: 2
}, {
    rfc_9d25d32fa7ee-maxDepth: 8,
    rfc_9d25d32fa7ee-numTrees: 4
})

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_44752ec6e46d

24°C ⚡ ENG 29-07-2022 20:49
```

## 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
sh.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

<console>:34: error: not found: value cross_validator
      val cvModel_harshdeep = cross_validator.harshdeep.fit(trainingdata_harshdeep)
                                         ^
scala> val cvModel_harshdeep = cross_validator.harshdeep.fit(trainingdata_harshdeep)
cvModel_harshdeep: org.apache.spark.ml.tuning.CrossValidatorModel = CrossValidatorModel: u
v_44752ec6e46d, bestModel=pipeline_ee907308b7b4, numFolds=3

scala> val predictions_harshdeep = cvModel_harshdeep.transform(testdata_harshdeep)
predictions_harshdeep: org.apache.spark.sql.DataFrame = [Pregnancies: double, Glucose: dou
... 11 more fields]

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

val accuracy_harshdeep = evaluator.evaluate(predictions_harshdeep)

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

// Exiting paste mode, now interpreting.

24°C ⚡ ENG 29-07-2022 20:54
```

## Prediction using testdata

```
<console>:34: error: not found: value cross_validator
           val cvModel_harshdeep = cross_validator.fit(trainingdata_harshdeep)
                                         ^
scala> val cvModel_harshdeep = cross_validator_harshdeep.fit(trainingdata_harshdeep)
cvModel_harshdeep: org.apache.spark.ml.tuning.CrossValidatorModel = CrossValidatorModel: u
v_44752ec6e46d, bestModel=pipeline_ee907308b7b4, numFolds=3

scala> val predictions_harshdeep = cvModel_harshdeep.transform(testdata_harshdeep)
predictions_harshdeep: org.apache.spark.sql.DataFrame = [Pregnancies: double, Glucose: dou
... 11 more fields]

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

val accuracy_harshdeep = evaluator.evaluate(predictions_harshdeep)

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

// Exiting paste mode, now interpreting.


```



## Evaluating the performance of our model

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

// Exiting paste mode, now interpreting.

<pastie>:34: error: not found: value evaluator
val accuracy_harshdeep = evaluator.evaluate(predictions_harshdeep)
                           ^
scala> :paste
// Entering paste mode (ctrl-D to finish)

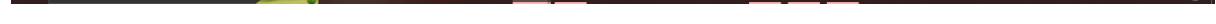
val accuracy_harshdeep = evaluator_harshdeep.evaluate(predictions_harshdeep)

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

// Exiting paste mode, now interpreting.

accuracy on test data = 0.8089171974522293
accuracy_harshdeep: Double = 0.8089171974522293

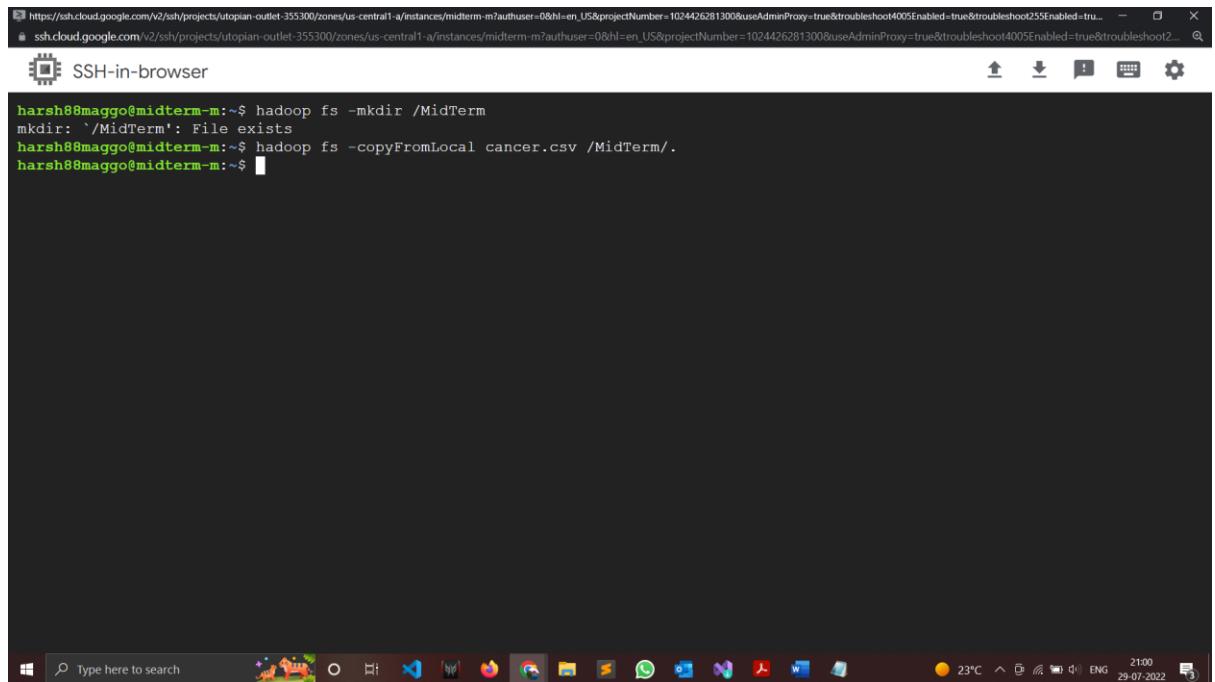
scala>
```



**ACCURACY: 80.89%**

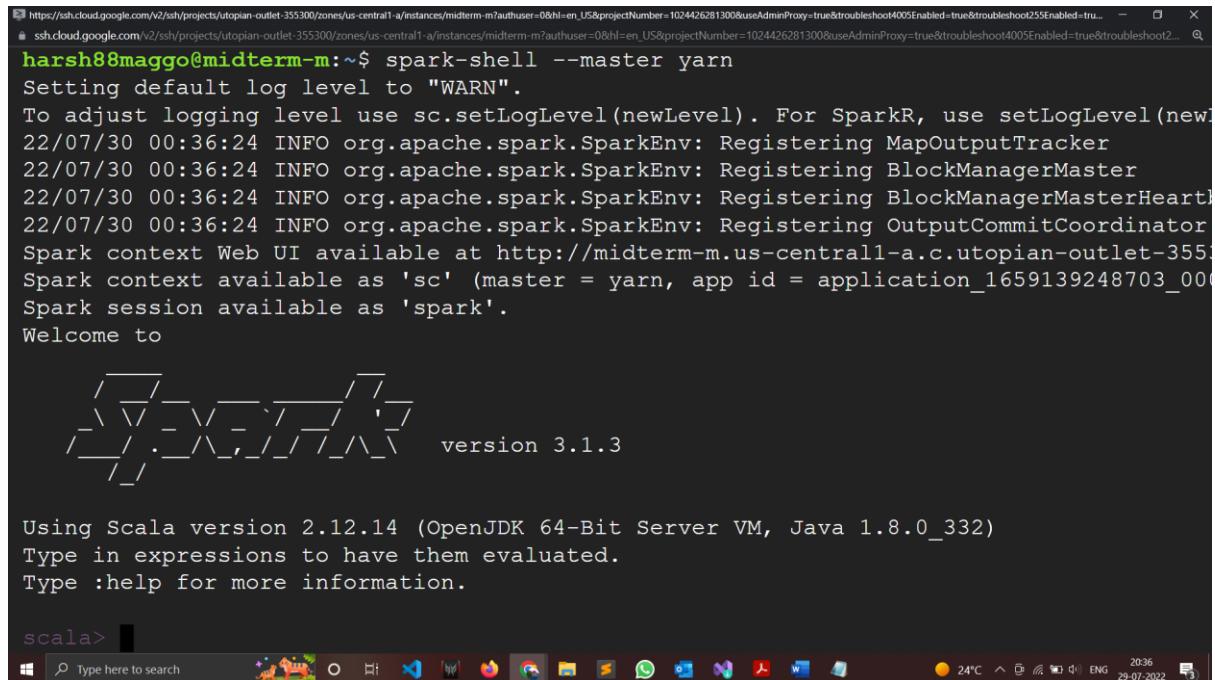
## Problem 2: Cancer Data (With Logistic Regression)

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



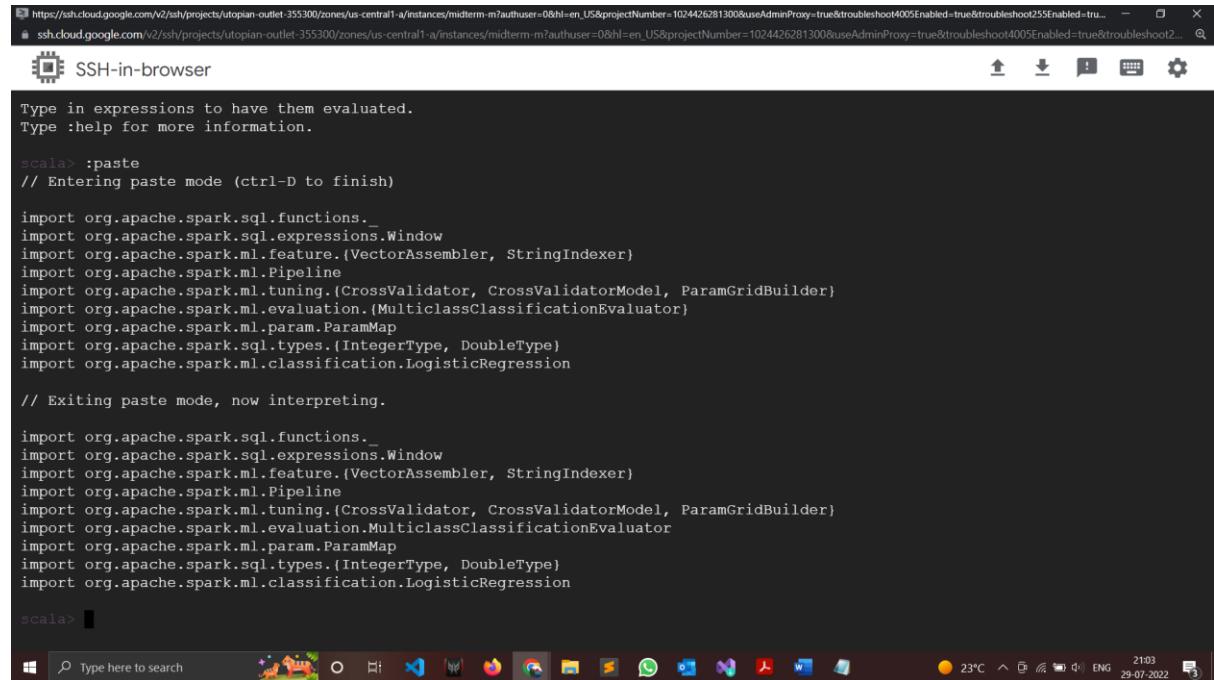
```
https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot400Enabled=true&troubleshoot255Enabled=true... - https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot400Enabled=true&troubleshoot255Enabled=true... x  
SSH-in-browser  
harsh88mago@midterm-m:~$ hadoop fs -mkdir /MidTerm  
mkdir: '/MidTerm': File exists  
harsh88mago@midterm-m:~$ hadoop fs -copyFromLocal cancer.csv /MidTerm/.  
harsh88mago@midterm-m:~$
```

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&troubleshoot400Enabled=true&troubleshoot255Enabled=true... - 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&troubleshoot400Enabled=true&troubleshoot255Enabled=true... x  
harsh88mago@midterm-m:~$ spark-shell --master yarn  
Setting default log level to "WARN".  
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).  
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-central1-a.c.utopian-outlet-355300.0000  
Spark context available as 'sc' (master = yarn, app id = application_1659139248703_0001).  
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



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

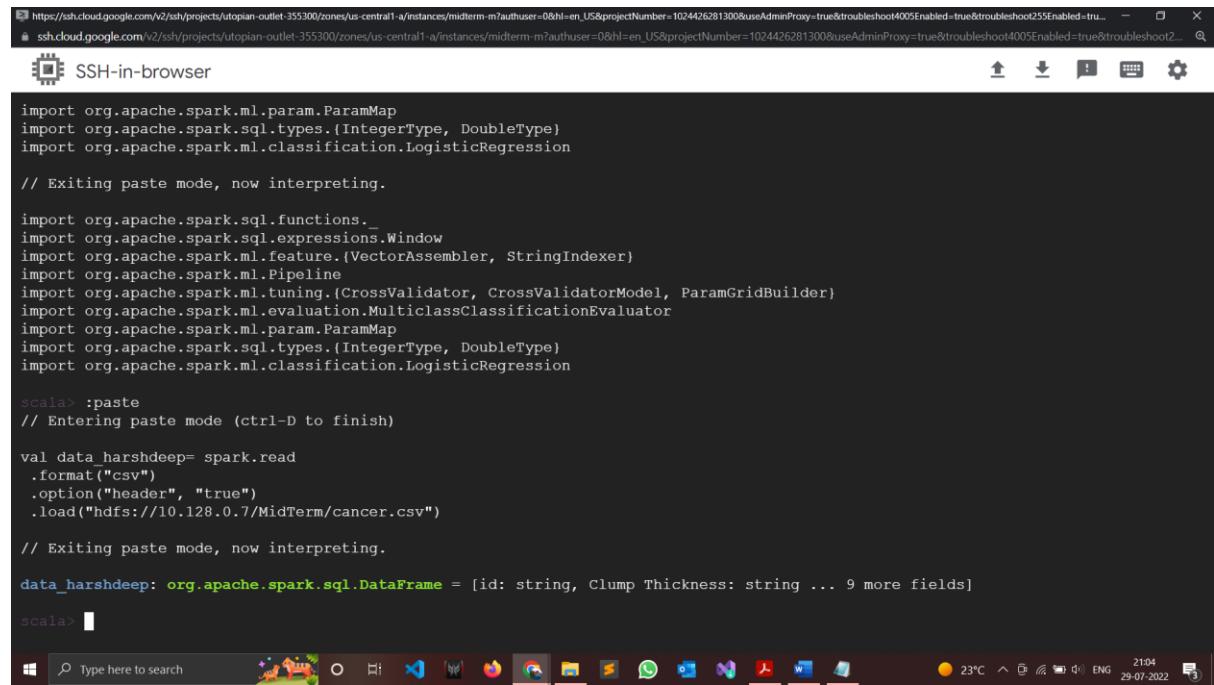
// Exiting paste mode, now interpreting.

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>
```

Windows taskbar at the bottom with various icons and system status.

## Reading the CSV file



```
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)

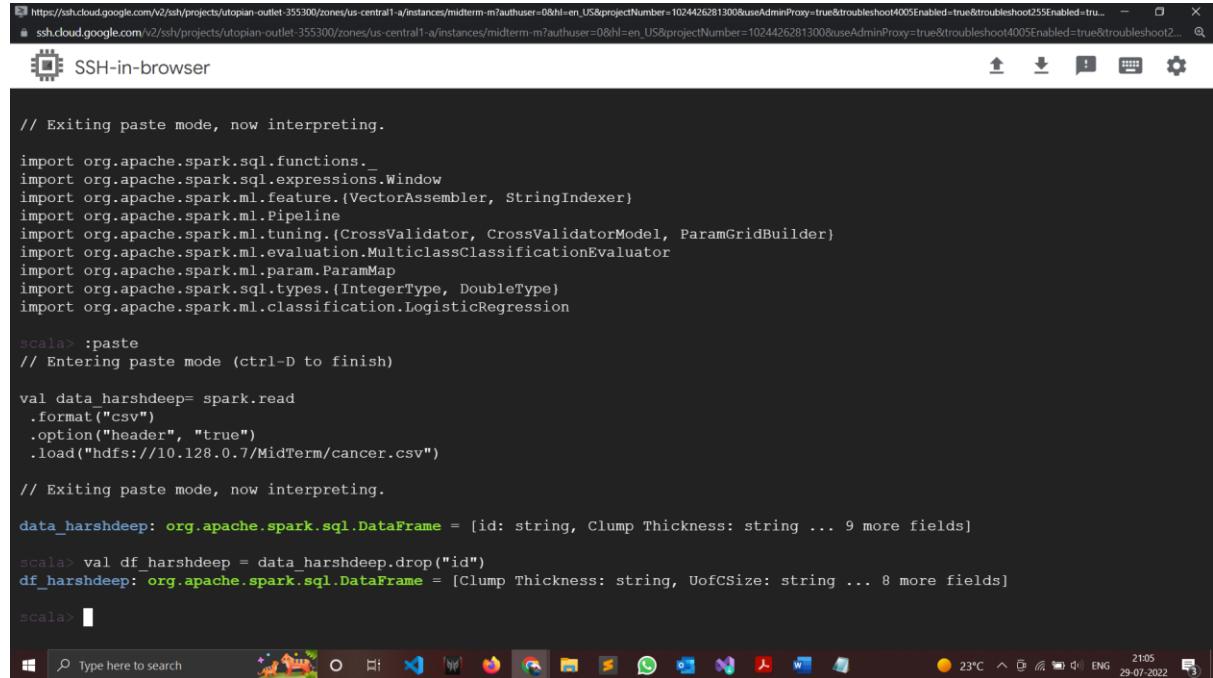
val data_harshdeep= spark.read
  .format("csv")
  .option("header", "true")
  .load("hdfs://10.128.0.7/MidTerm/cancer.csv")

// Exiting paste mode, now interpreting.

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

Windows taskbar at the bottom with various icons and system status.

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



```
// 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")

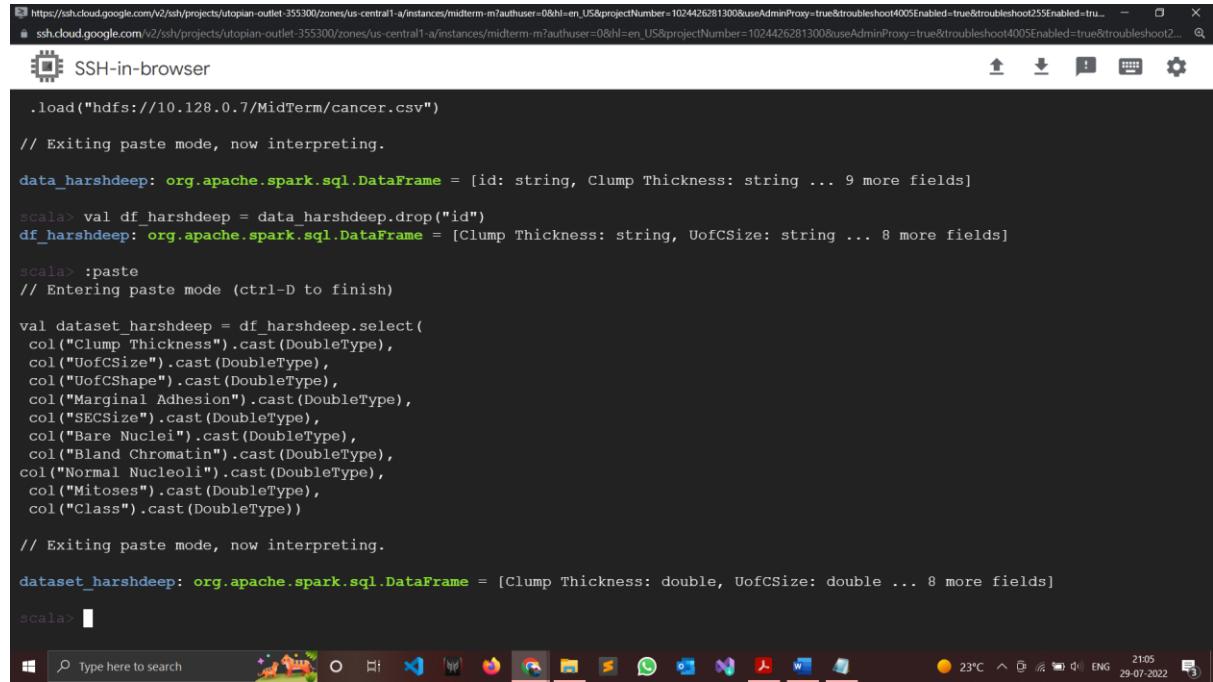
// Exiting paste mode, now interpreting.

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



```
.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

The screenshot shows a terminal window titled "SSH-in-browser". The terminal content is as follows:

```
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 m
ore fields]

scala> 
```

The terminal is running on a Windows operating system, as indicated by the taskbar at the bottom.

## Assembling the features using VectorAssembler

The screenshot shows a terminal window titled "SSH-in-browser". The terminal content is as follows:

```
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 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")

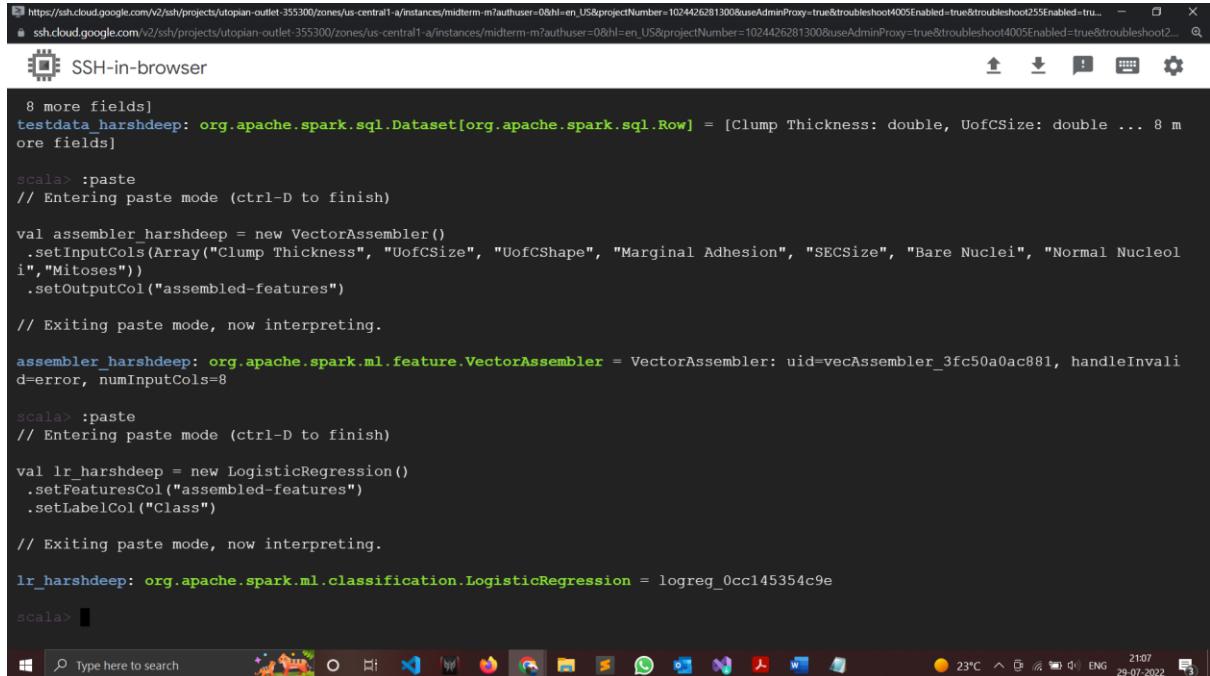
// Exiting paste mode, now interpreting.

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

scala> 
```

The terminal is running on a Windows operating system, as indicated by the taskbar at the bottom.

## Creating the Logistic Regression Object and passing the features



```
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 Nucleoli", "Mitoses"))
.setOutputCol("assembled-features")

// Exiting paste mode, now interpreting.

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

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

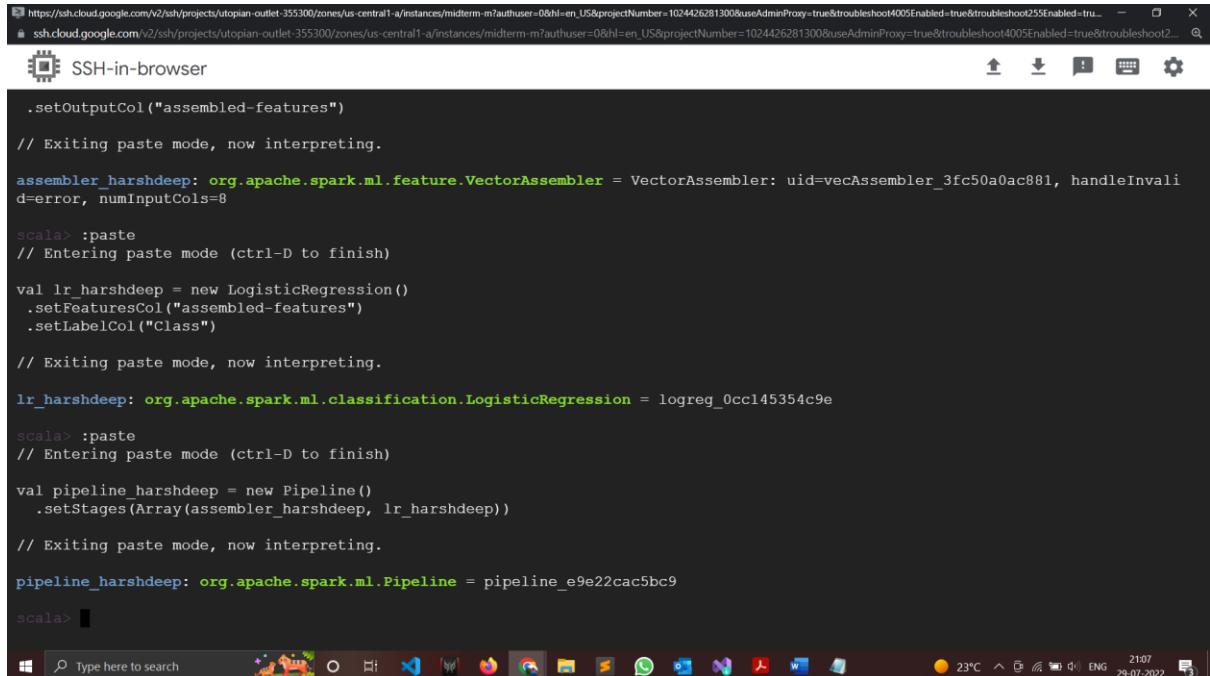
val lr_harshdeep = new LogisticRegression()
.setFeaturesCol("assembled-features")
.setLabelCol("Class")

// Exiting paste mode, now interpreting.

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

scala>
```

## Creating the pipeline



```
.setOutputCol("assembled-features")

// Exiting paste mode, now interpreting.

assembler_harshdeep: org.apache.spark.ml.feature.VectorAssembler = VectorAssembler: uid=vecAssembler_3fc50a0ac881, handleInvali
dError, 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)

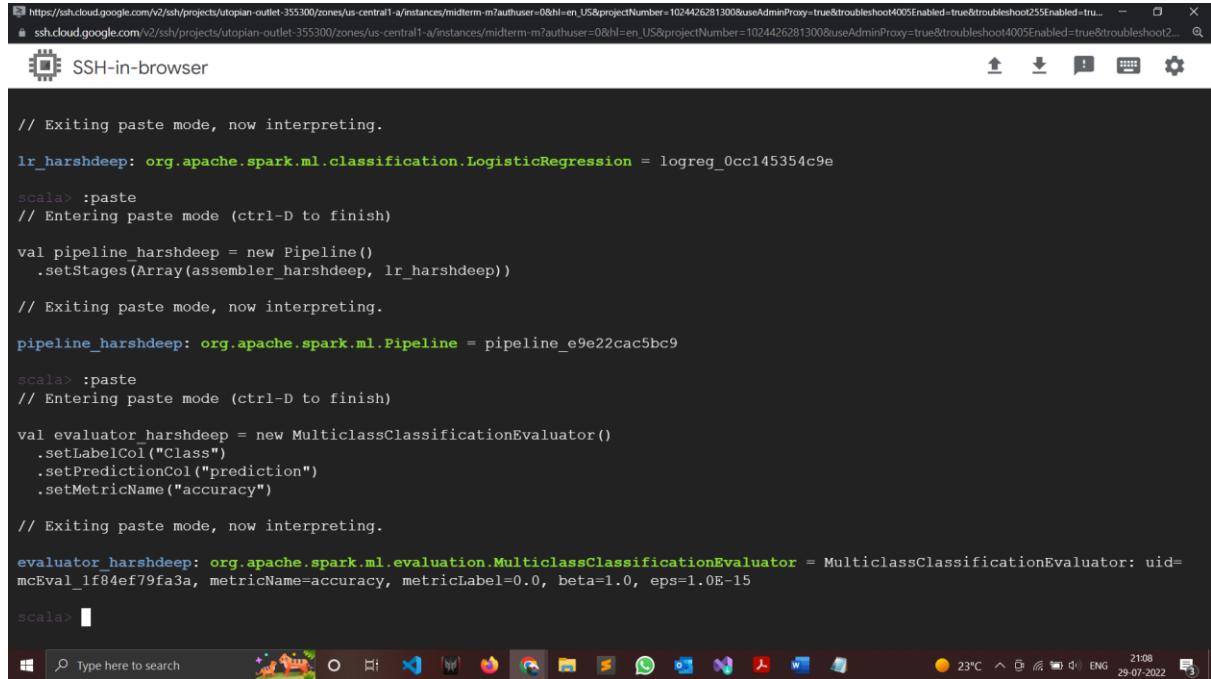
val pipeline_harshdeep = new Pipeline()
.setStages(Array(assembler_harshdeep, lr_harshdeep))

// Exiting paste mode, now interpreting.

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

scala>
```

## Evaluator for our model



```
// 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)

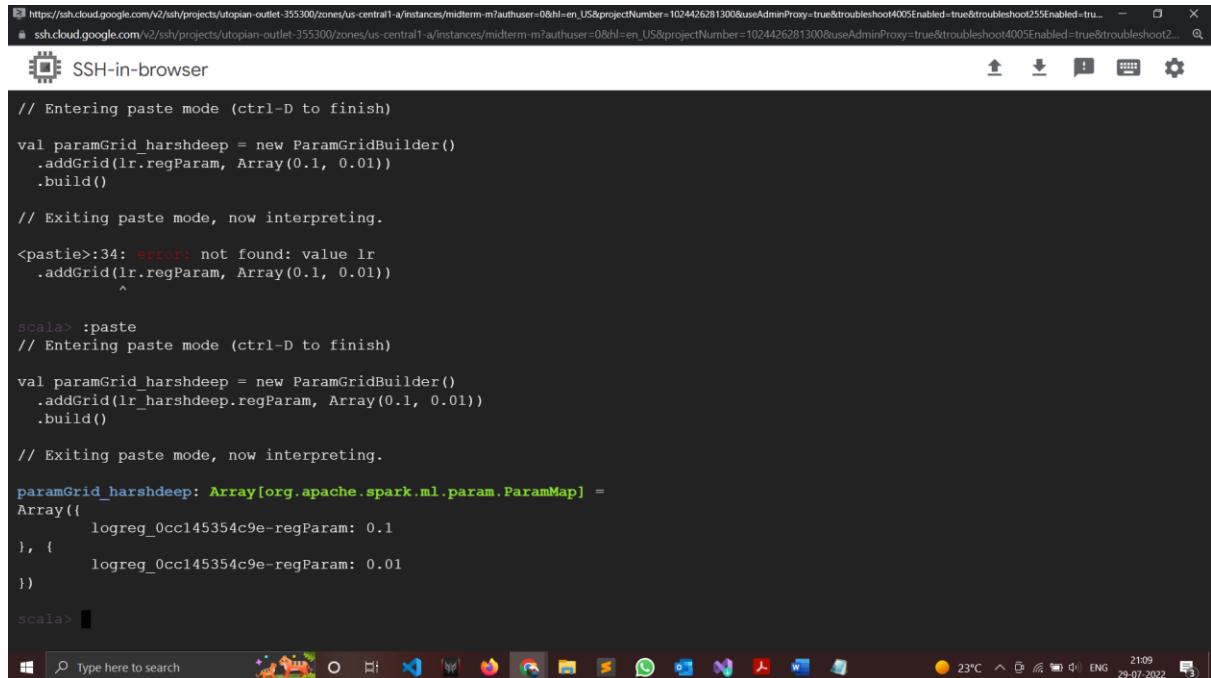
val evaluator_harshdeep = new MulticlassClassificationEvaluator()
  .setLabelCol("Class")
  .setPredictionCol("prediction")
  .setMetricName("accuracy")

// 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



```
// 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.

<pastie>:34: error: not found: value lr
      .addGrid(lr.regParam, Array(0.1, 0.01))
           ^
scala> :paste
// Entering paste mode (ctrl-D to finish)

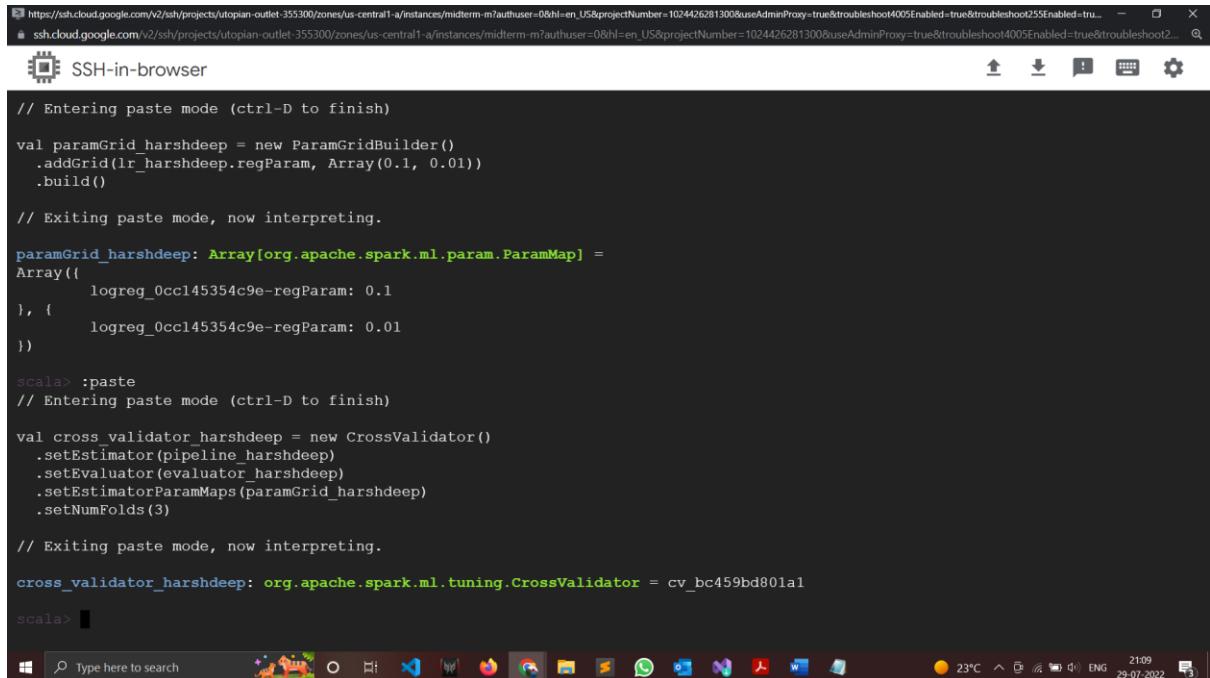
val paramGrid_harshdeep = new ParamGridBuilder()
  .addGrid(lr_harshdeep.regParam, Array(0.1, 0.01))
  .build()

// 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



```
// 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)

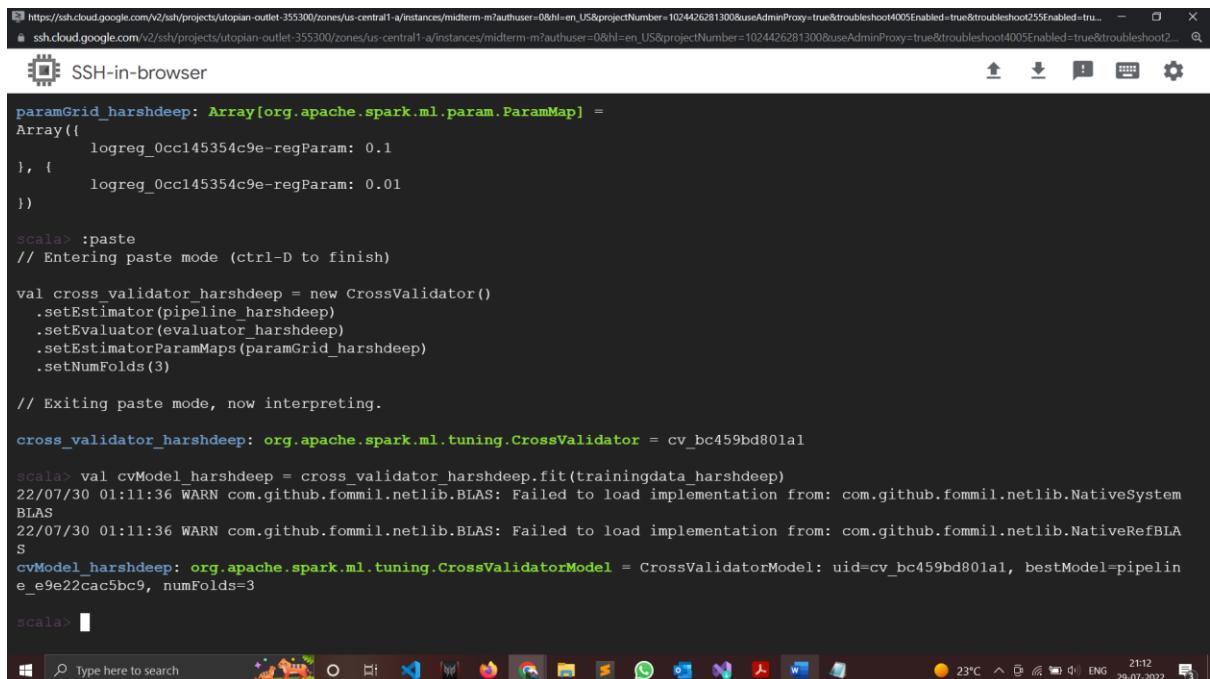
val cross_validator_harshdeep = new CrossValidator()
  .setEstimator(pipeline_harshdeep)
  .setEvaluator(evaluator_harshdeep)
  .setEstimatorParamMaps(paramGrid_harshdeep)
  .setNumFolds(3)

// Exiting paste mode, now interpreting.

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

scala>
```

## Training our model



```
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)

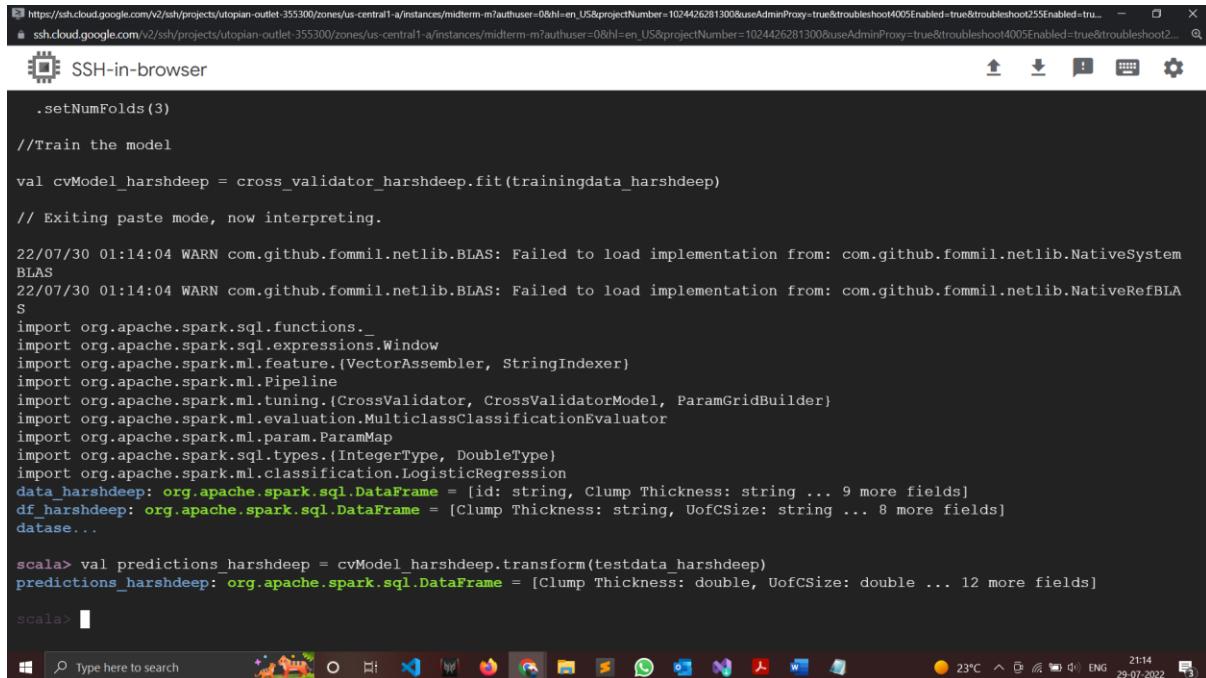
// Exiting paste mode, now interpreting.

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.NativeSystemBLAS
22/07/30 01:11:36 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
cvModel_harshdeep: org.apache.spark.ml.tuning.CrossValidatorModel = CrossValidatorModel: uid=cv_bc459bd801a1, bestModel=pipeliner_e9e22cac5bc9, numFolds=3

scala>
```

## Prediction using testdata



```
.setNumFolds(3)

//Train the model

val cvModel_harshdeep = cross_validator_harshdeep.fit(trainingdata_harshdeep)

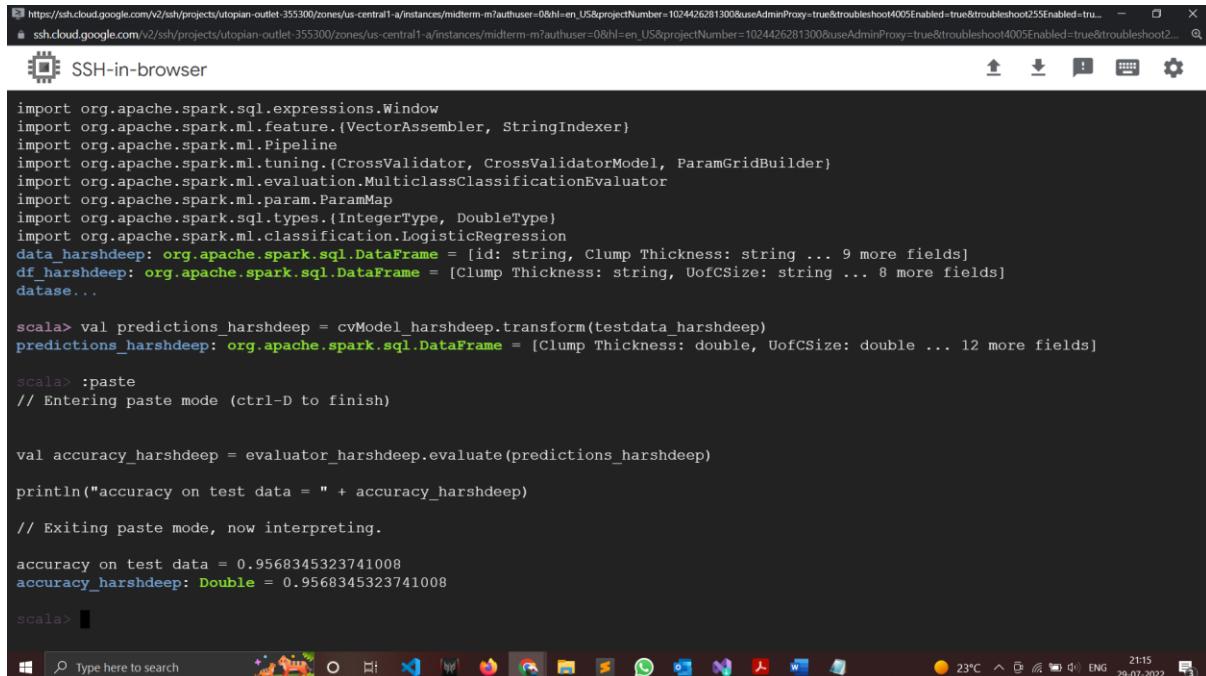
// Exiting paste mode, now interpreting.

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



```
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.

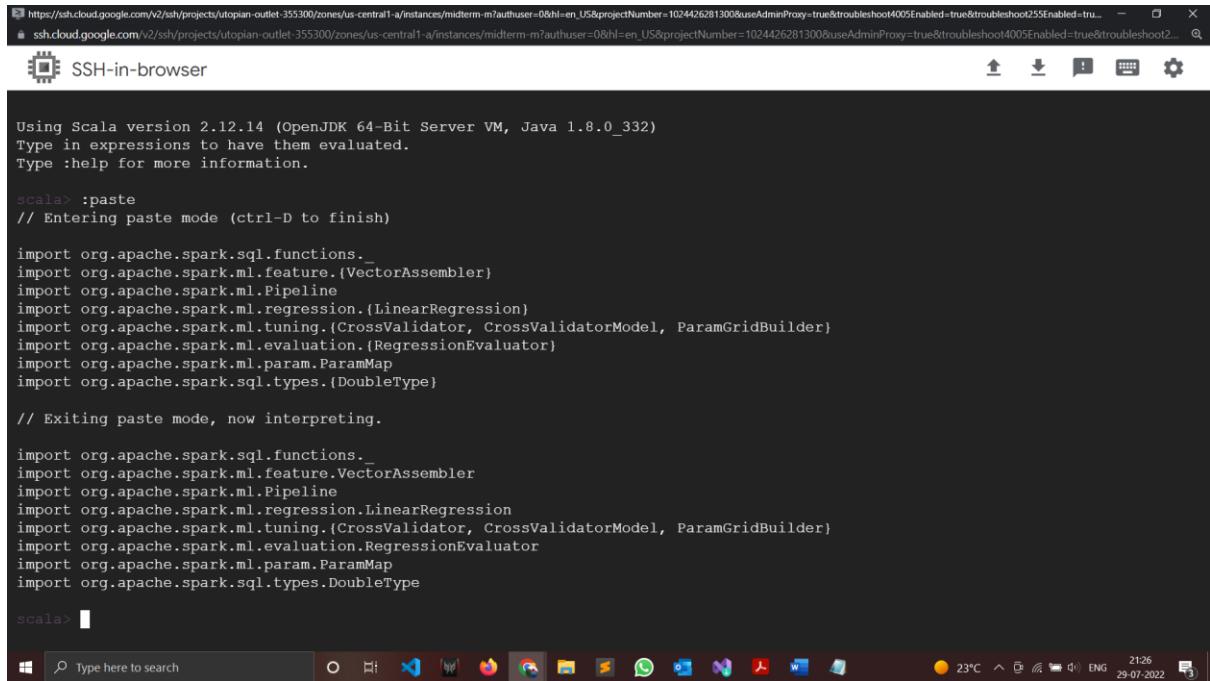
accuracy on test data = 0.9568345323741008
accuracy_harshdeep: Double = 0.9568345323741008

scala>
```

**ACCURACY: 95.68%**



## Import statements



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

import org.apache.spark.sql.functions._
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.regression.LinearRegression
import org.apache.spark.ml.tuning.CrossValidator, CrossValidatorModel, ParamGridBuilder
import org.apache.spark.ml.evaluation.RegressionEvaluator
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.sql.types.DoubleType

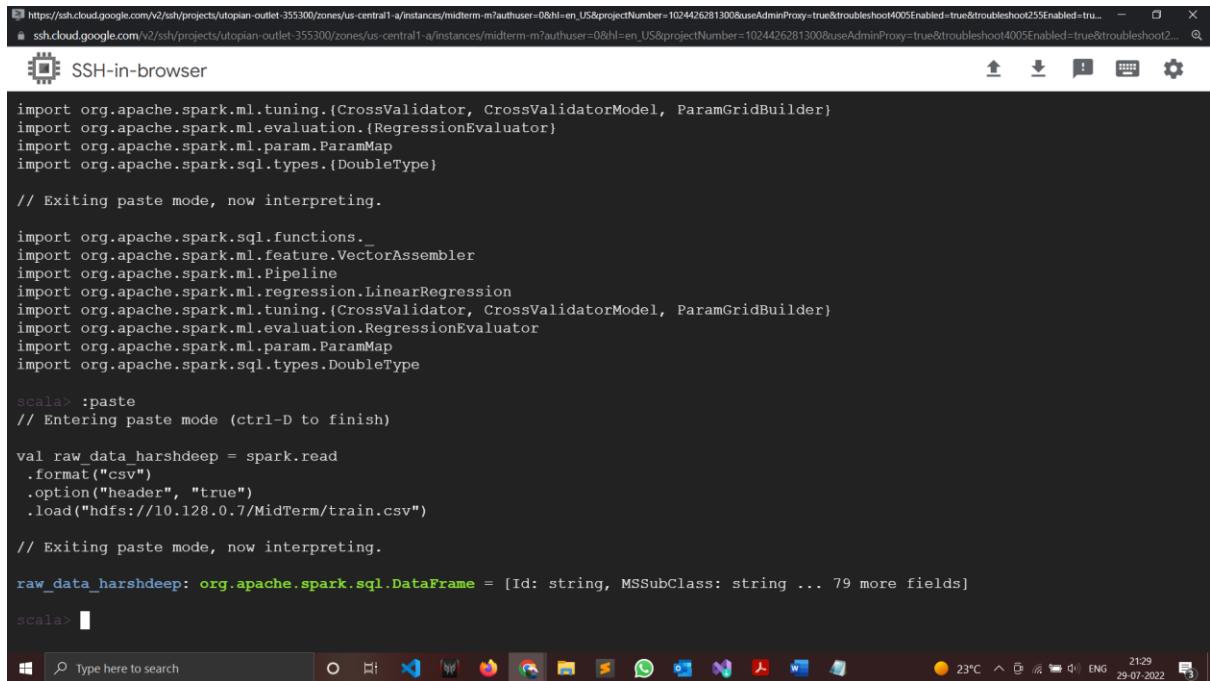
// Exiting paste mode, now interpreting.

import org.apache.spark.sql.functions._
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.regression.LinearRegression
import org.apache.spark.ml.tuning.CrossValidator, CrossValidatorModel, ParamGridBuilder
import org.apache.spark.ml.evaluation.RegressionEvaluator
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.sql.types.DoubleType

scala>
```

Type here to search    23°C 2126 29-07-2022

## Reading the CSV file



```
import org.apache.spark.ml.tuning.CrossValidator, CrossValidatorModel, ParamGridBuilder
import org.apache.spark.ml.evaluation.RegressionEvaluator
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.sql.types.DoubleType

// Exiting paste mode, now interpreting.

import org.apache.spark.sql.functions._
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.regression.LinearRegression
import org.apache.spark.ml.tuning.CrossValidator, CrossValidatorModel, ParamGridBuilder
import org.apache.spark.ml.evaluation.RegressionEvaluator
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.sql.types.DoubleType

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

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

// Exiting paste mode, now interpreting.

raw_data_harshdeep: org.apache.spark.sql.DataFrame = [Id: string, MSSubClass: string ... 79 more fields]

scala>
```

Type here to search    23°C 2129 29-07-2022

Dropping the NA values from the dataset, selecting columns for model training, and typecasting the data in type Double

```
//Drop NA values
val clean_harshdeep = raw_data_harshdeep.na.drop()

//Selecting columns for model training
val data_harshdeep = clean_harshdeep.select(
  col("MSSubClass").cast(DoubleType),
  col("LotArea").cast(DoubleType),
  col("OverallCond").cast(DoubleType),
  col("TotalBsmtSF").cast(DoubleType),
  col("1stFlrSF").cast(DoubleType),
  col("GrLivArea").cast(DoubleType),
  col("FullBath").cast(DoubleType),
  col("TotRmsAbvGrd").cast(DoubleType),
  col("GarageArea").cast(DoubleType),
  col("WoodDeckSF").cast(DoubleType),
  col("OpenPorchSF").cast(DoubleType),
  col("SalePrice").cast(DoubleType))

// Exiting paste mode, now interpreting.

import org.apache.spark.sql.functions._
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.regression.LinearRegression
import org.apache.spark.ml.tuning.{CrossValidator, CrossValidatorModel, ParamGridBuilder}
import org.apache.spark.ml.evaluation.RegressionEvaluator
import org.apache.spark.ml.param.ParamMap

scala> Type here to search  O 22°C  ENG 29-07-2022 21:32
```

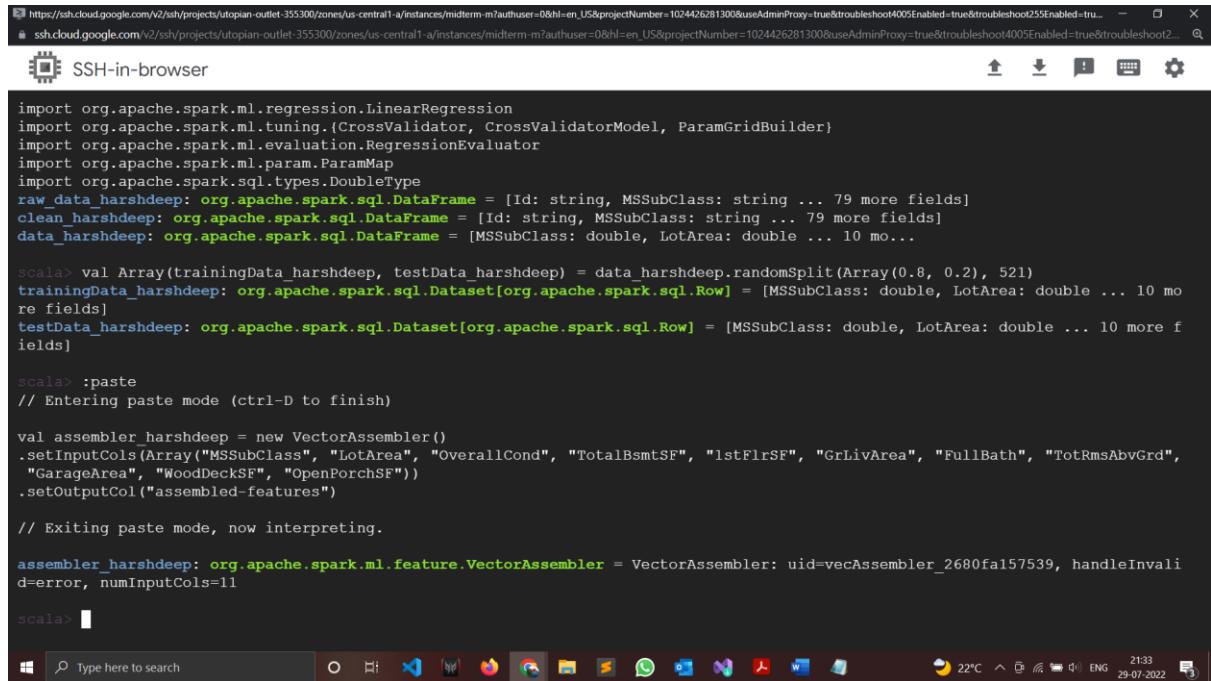
Split the dataset into train and test

```
col("GrLivArea").cast(DoubleType),
col("FullBath").cast(DoubleType),
col("TotRmsAbvGrd").cast(DoubleType),
col("GarageArea").cast(DoubleType),
col("WoodDeckSF").cast(DoubleType),
col("OpenPorchSF").cast(DoubleType),
col("SalePrice").cast(DoubleType))

// Exiting paste mode, now interpreting.

import org.apache.spark.sql.functions._
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.regression.LinearRegression
import org.apache.spark.ml.tuning.{CrossValidator, CrossValidatorModel, ParamGridBuilder}
import org.apache.spark.ml.evaluation.RegressionEvaluator
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.sql.types.DoubleType
raw_data_harshdeep: org.apache.spark.sql.DataFrame = [Id: string, MSSubClass: string ... 79 more fields]
clean_harshdeep: org.apache.spark.sql.DataFrame = [Id: string, MSSubClass: string ... 79 more fields]
data_harshdeep: org.apache.spark.sql.DataFrame = [MSSubClass: double, LotArea: double ... 10 mo...
re fields]
scala> val Array(trainingData_harshdeep, testData_harshdeep) = data_harshdeep.randomSplit(Array(0.8, 0.2), 521)
trainingData_harshdeep: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [MSSubClass: double, LotArea: double ... 10 mo...
re fields]
testData_harshdeep: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [MSSubClass: double, LotArea: double ... 10 more f...
ields]
scala> Type here to search  O 22°C  ENG 29-07-2022 21:33
```

## Assembling the features using VectorAssembler



```
import org.apache.spark.ml.regression.LinearRegression
import org.apache.spark.ml.tuning.{CrossValidator, CrossValidatorModel, ParamGridBuilder}
import org.apache.spark.ml.evaluation.RegressionEvaluator
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.sql.types.DoubleType
raw_data_harshdeep: org.apache.spark.sql.DataFrame = [Id: string, MSSubClass: string ... 79 more fields]
clean_harshdeep: org.apache.spark.sql.DataFrame = [Id: string, MSSubClass: string ... 79 more fields]
data_harshdeep: org.apache.spark.sql.DataFrame = [MSSubClass: double, LotArea: double ... 10 mo...
scala> val Array(trainingData_harshdeep, testData_harshdeep) = data_harshdeep.randomSplit(Array(0.8, 0.2), 521)
trainingData_harshdeep: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [MSSubClass: double, LotArea: double ... 10 mo...
re fields]
testData_harshdeep: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [MSSubClass: double, LotArea: double ... 10 more f...
ields]

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

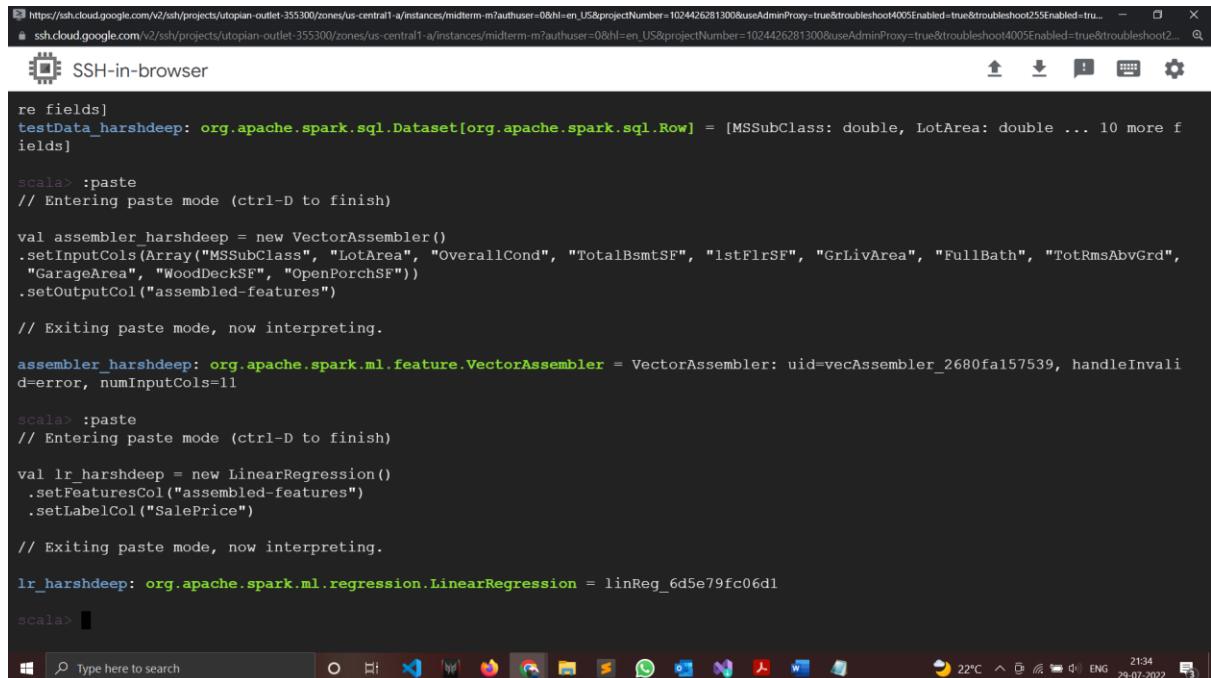
val assembler_harshdeep = new VectorAssembler()
.setInputCols(Array("MSSubClass", "LotArea", "OverallCond", "TotalBsmtSF", "1stFlrSF", "GrLivArea", "FullBath", "TotRmsAbvGrd",
"GarageArea", "WoodDeckSF", "OpenPorchSF"))
.setOutputCol("assembled-features")

// Exiting paste mode, now interpreting.

assembler_harshdeep: org.apache.spark.ml.feature.VectorAssembler = VectorAssembler: uid=vecAssembler_2680fa157539, handleInvali...
dError, numInputCols=11

scala>
```

## Creating the Linear Regression Object and passing the features



```
re fields]
testData_harshdeep: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [MSSubClass: double, LotArea: double ... 10 more f...
ields]

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

val assembler_harshdeep = new VectorAssembler()
.setInputCols(Array("MSSubClass", "LotArea", "OverallCond", "TotalBsmtSF", "1stFlrSF", "GrLivArea", "FullBath", "TotRmsAbvGrd",
"GarageArea", "WoodDeckSF", "OpenPorchSF"))
.setOutputCol("assembled-features")

// Exiting paste mode, now interpreting.

assembler_harshdeep: org.apache.spark.ml.feature.VectorAssembler = VectorAssembler: uid=vecAssembler_2680fa157539, handleInvali...
dError, numInputCols=11

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

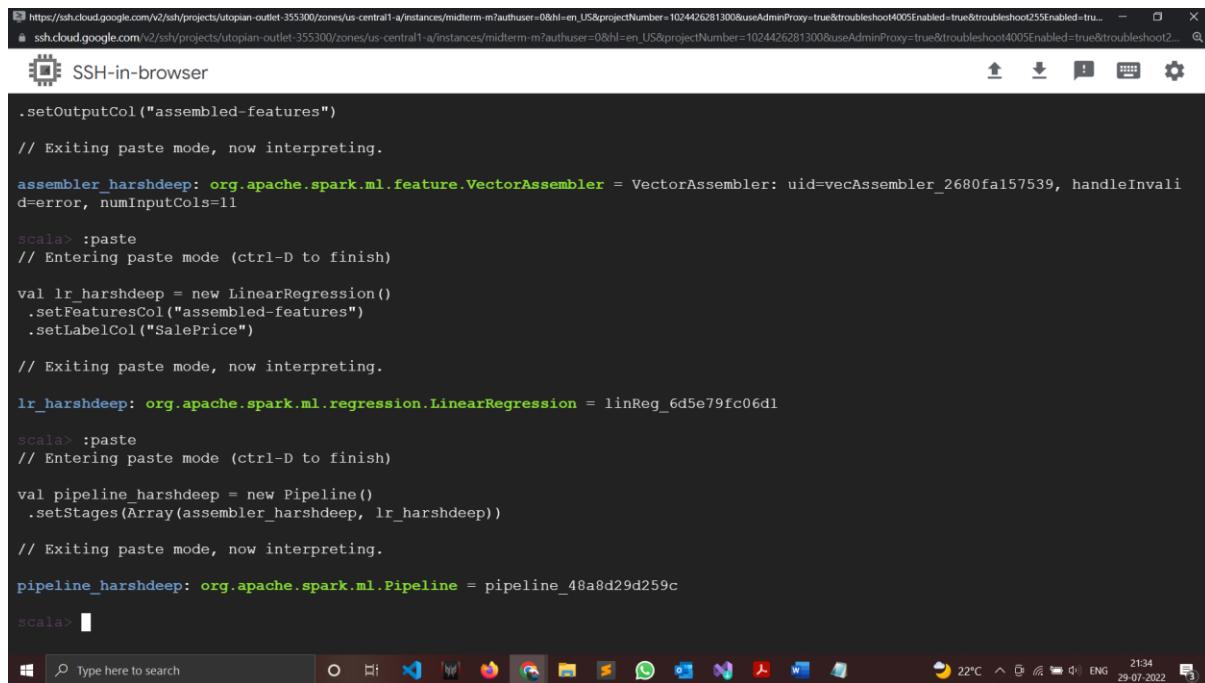
val lr_harshdeep = new LinearRegression()
.setFeaturesCol("assembled-features")
.setLabelCol("SalePrice")

// Exiting paste mode, now interpreting.

lr_harshdeep: org.apache.spark.ml.regression.LinearRegression = linReg_6d5e79fc06d1

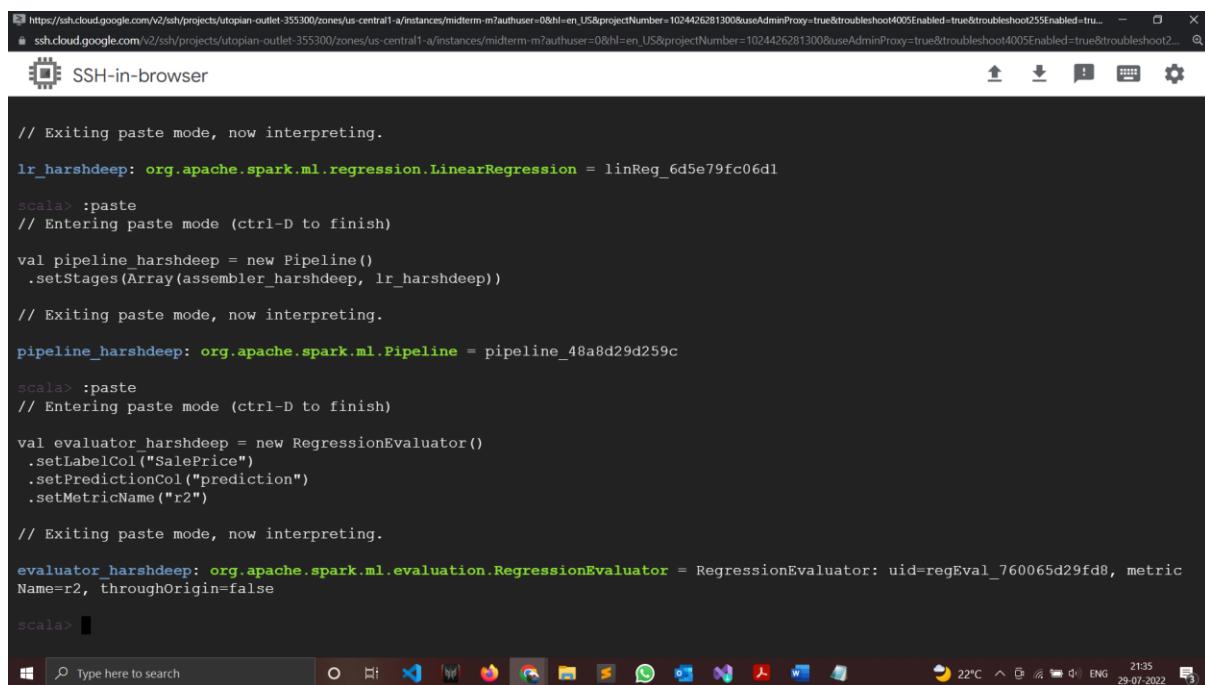
scala>
```

## Creating the pipeline



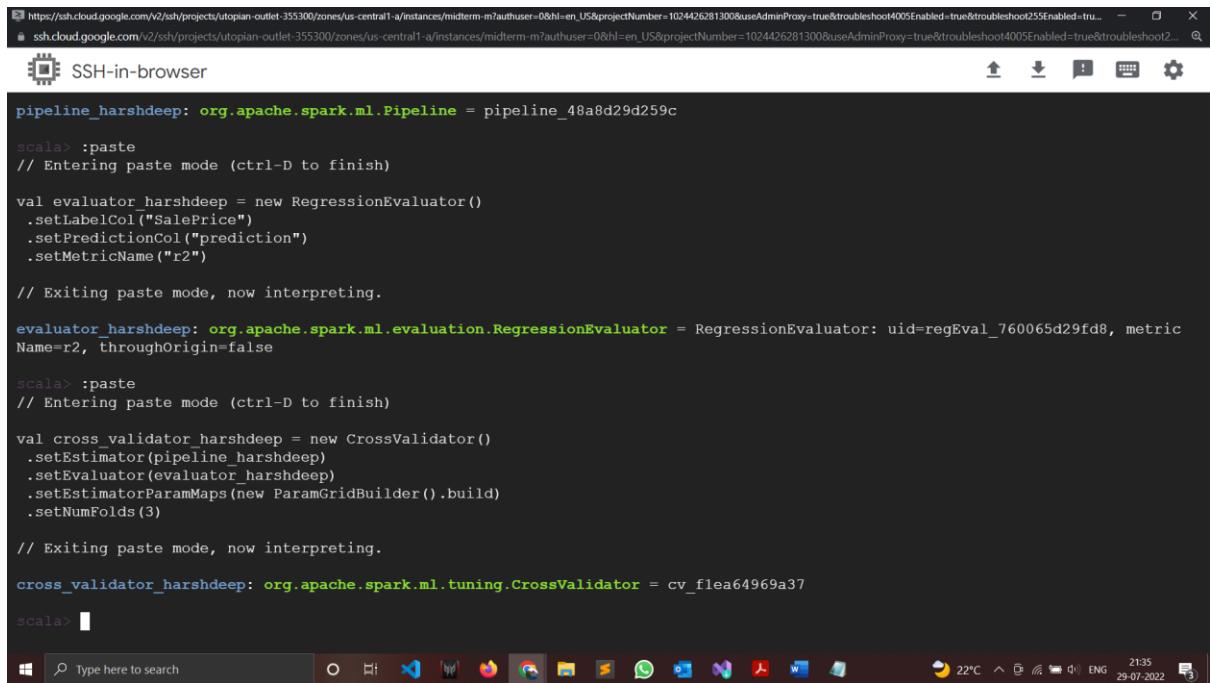
```
https://ssh.cloud.google.com/v2/sh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true&troubleshoot2...  
ssh.cloud.google.com/v2/sh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true&troubleshoot2...  
  
SSH-in-browser  
  
.setOutputCol("assembled-features")  
  
// Exiting paste mode, now interpreting.  
  
assembler_harshdeep: org.apache.spark.ml.feature.VectorAssembler = VectorAssembler: uid=vecAssembler_2680fa157539, handleInvalidError, numInputCols=11  
  
scala> :paste  
// Entering paste mode (ctrl-D to finish)  
  
val lr_harshdeep = new LinearRegression()  
.setFeaturesCol("assembled-features")  
.setLabelCol("SalePrice")  
  
// Exiting paste mode, now interpreting.  
  
lr_harshdeep: org.apache.spark.ml.regression.LinearRegression = linReg_6d5e79fc06d1  
  
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_48a8d29d259c  
  
scala>
```

## Evaluator for our model



```
https://ssh.cloud.google.com/v2/sh/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...  
ssh.cloud.google.com/v2/sh/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...  
  
SSH-in-browser  
  
// Exiting paste mode, now interpreting.  
  
lr_harshdeep: org.apache.spark.ml.regression.LinearRegression = linReg_6d5e79fc06d1  
  
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_48a8d29d259c  
  
scala> :paste  
// Entering paste mode (ctrl-D to finish)  
  
val evaluator_harshdeep = new RegressionEvaluator()  
.setLabelCol("SalePrice")  
.setPredictionCol("prediction")  
.setMetricName("r2")  
  
// Exiting paste mode, now interpreting.  
  
evaluator_harshdeep: org.apache.spark.ml.evaluation.RegressionEvaluator = RegressionEvaluator: uid=regEval_760065d29fd8, metricName=r2, throughOrigin=false  
  
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

pipeline_harshdeep: org.apache.spark.ml.Pipeline = pipeline_48a8d29d259c
scala> :paste
// Entering paste mode (ctrl-D to finish)

val evaluator_harshdeep = new RegressionEvaluator()
.setLabelCol("SalePrice")
.setPredictionCol("prediction")
.setMetricName("r2")

// Exiting paste mode, now interpreting.

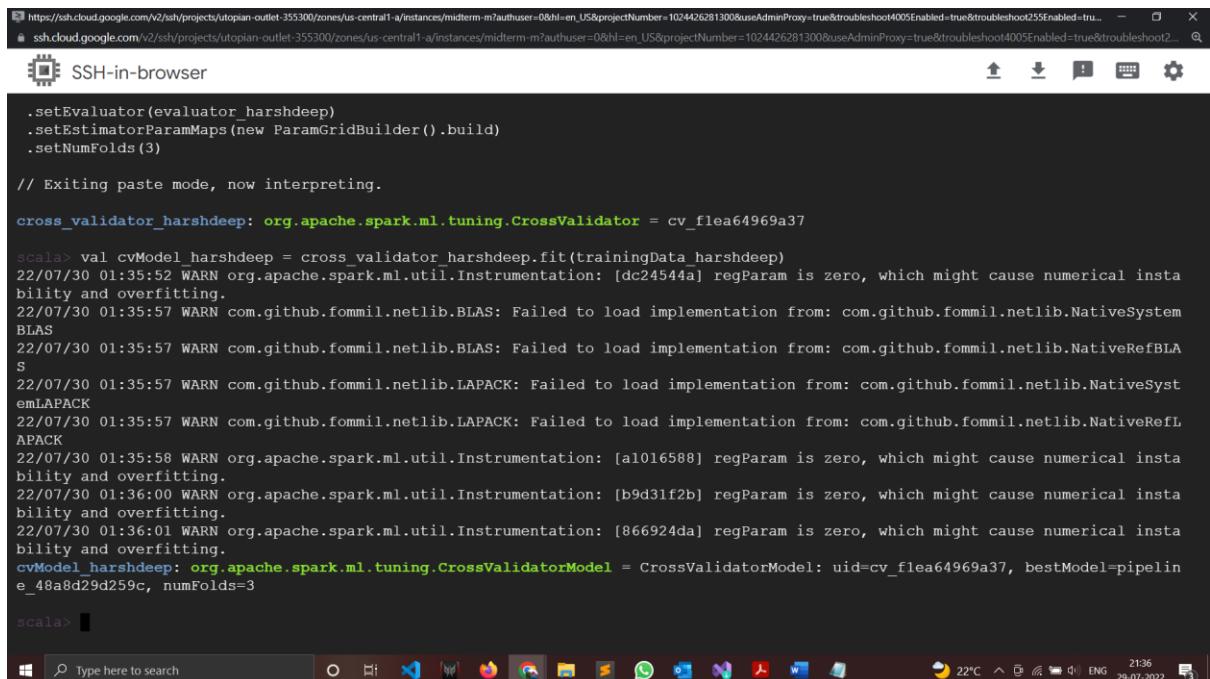
evaluator_harshdeep: org.apache.spark.ml.evaluation.RegressionEvaluator = RegressionEvaluator: uid=regEval_760065d29fd8, metricName=r2, throughOrigin=false
scala> :paste
// Entering paste mode (ctrl-D to finish)

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

// Exiting paste mode, now interpreting.

cross_validator_harshdeep: org.apache.spark.ml.tuning.CrossValidator = cv_flea64969a37
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

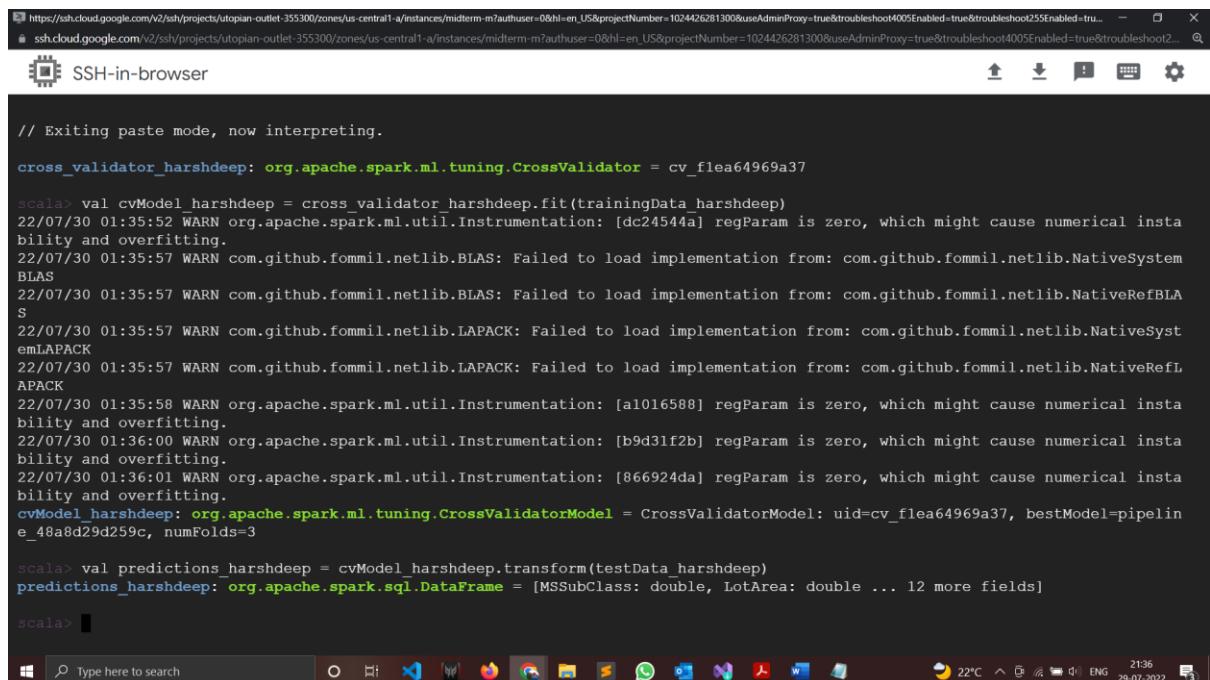
.setEvaluator(evaluator_harshdeep)
.setEstimatorParamMaps(new ParamGridBuilder().build())
.setNumFolds(3)

// Exiting paste mode, now interpreting.

cross_validator_harshdeep: org.apache.spark.ml.tuning.CrossValidator = cv_flea64969a37

scala> val cvModel_harshdeep = cross_validator_harshdeep.fit(trainingData_harshdeep)
22/07/30 01:35:52 WARN org.apache.spark.ml.util.Instrumentation: [dc24544a] regParam is zero, which might cause numerical instability and overfitting.
22/07/30 01:35:57 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
22/07/30 01:35:57 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
22/07/30 01:35:57 WARN com.github.fommil.netlib.LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeSystemLAPACK
22/07/30 01:35:57 WARN com.github.fommil.netlib.LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeRefLAPACK
22/07/30 01:35:58 WARN org.apache.spark.ml.util.Instrumentation: [a1016588] regParam is zero, which might cause numerical instability and overfitting.
22/07/30 01:36:00 WARN org.apache.spark.ml.util.Instrumentation: [b9d31f2b] regParam is zero, which might cause numerical instability and overfitting.
22/07/30 01:36:01 WARN org.apache.spark.ml.util.Instrumentation: [866924da] regParam is zero, which might cause numerical instability and overfitting.
cvModel_harshdeep: org.apache.spark.ml.tuning.CrossValidatorModel = CrossValidatorModel: uid=cv_flea64969a37, bestModel=pipeline_48a8d29d259c, numFolds=3
scala>
```

## Prediction using testdata



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

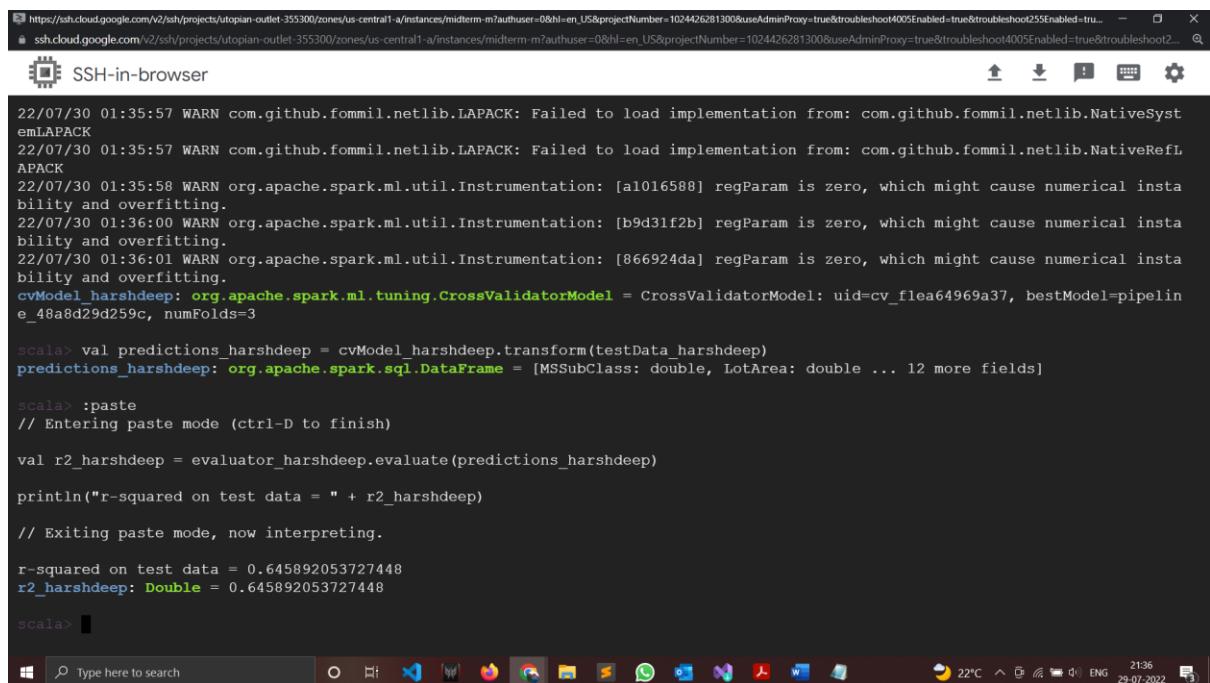
cross_validator_harshdeep: org.apache.spark.ml.tuning.CrossValidator = cv_flea64969a37

scala> val cvModel_harshdeep = cross_validator_harshdeep.fit(trainingData_harshdeep)
22/07/30 01:35:52 WARN org.apache.spark.ml.util.Instrumentation: [dc24544a] regParam is zero, which might cause numerical instability and overfitting.
22/07/30 01:35:57 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
22/07/30 01:35:57 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
22/07/30 01:35:57 WARN com.github.fommil.netlib.LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeSystemLAPACK
22/07/30 01:35:57 WARN com.github.fommil.netlib.LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeRefLAPACK
22/07/30 01:35:58 WARN org.apache.spark.ml.util.Instrumentation: [a1016588] regParam is zero, which might cause numerical instability and overfitting.
22/07/30 01:36:00 WARN org.apache.spark.ml.util.Instrumentation: [b9d31f2b] regParam is zero, which might cause numerical instability and overfitting.
22/07/30 01:36:01 WARN org.apache.spark.ml.util.Instrumentation: [866924da] regParam is zero, which might cause numerical instability and overfitting.
cvModel_harshdeep: org.apache.spark.ml.tuning.CrossValidatorModel = CrossValidatorModel: uid=cv_flea64969a37, bestModel=pipeline_48a8d29d259c, numFolds=3

scala> val predictions_harshdeep = cvModel_harshdeep.transform(testData_harshdeep)
predictions_harshdeep: org.apache.spark.sql.DataFrame = [MSSubClass: double, LotArea: double ... 12 more fields]

scala>
```

## Evaluating the performance of our model



```
22/07/30 01:35:57 WARN com.github.fommil.netlib.LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeSystemLAPACK
22/07/30 01:35:57 WARN com.github.fommil.netlib.LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeRefLAPACK
22/07/30 01:35:58 WARN org.apache.spark.ml.util.Instrumentation: [a1016588] regParam is zero, which might cause numerical instability and overfitting.
22/07/30 01:36:00 WARN org.apache.spark.ml.util.Instrumentation: [b9d31f2b] regParam is zero, which might cause numerical instability and overfitting.
22/07/30 01:36:01 WARN org.apache.spark.ml.util.Instrumentation: [866924da] regParam is zero, which might cause numerical instability and overfitting.
cvModel_harshdeep: org.apache.spark.ml.tuning.CrossValidatorModel = CrossValidatorModel: uid=cv_flea64969a37, bestModel=pipeline_48a8d29d259c, numFolds=3

scala> val predictions_harshdeep = cvModel_harshdeep.transform(testData_harshdeep)
predictions_harshdeep: org.apache.spark.sql.DataFrame = [MSSubClass: double, LotArea: double ... 12 more fields]

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

val r2_harshdeep = evaluator_harshdeep.evaluate(predictions_harshdeep)
println("r-squared on test data = " + r2_harshdeep)

// Exiting paste mode, now interpreting.

r-squared on test data = 0.645892053727448
r2_harshdeep: Double = 0.645892053727448

scala>
```

**ACCURACY: 64.58%**