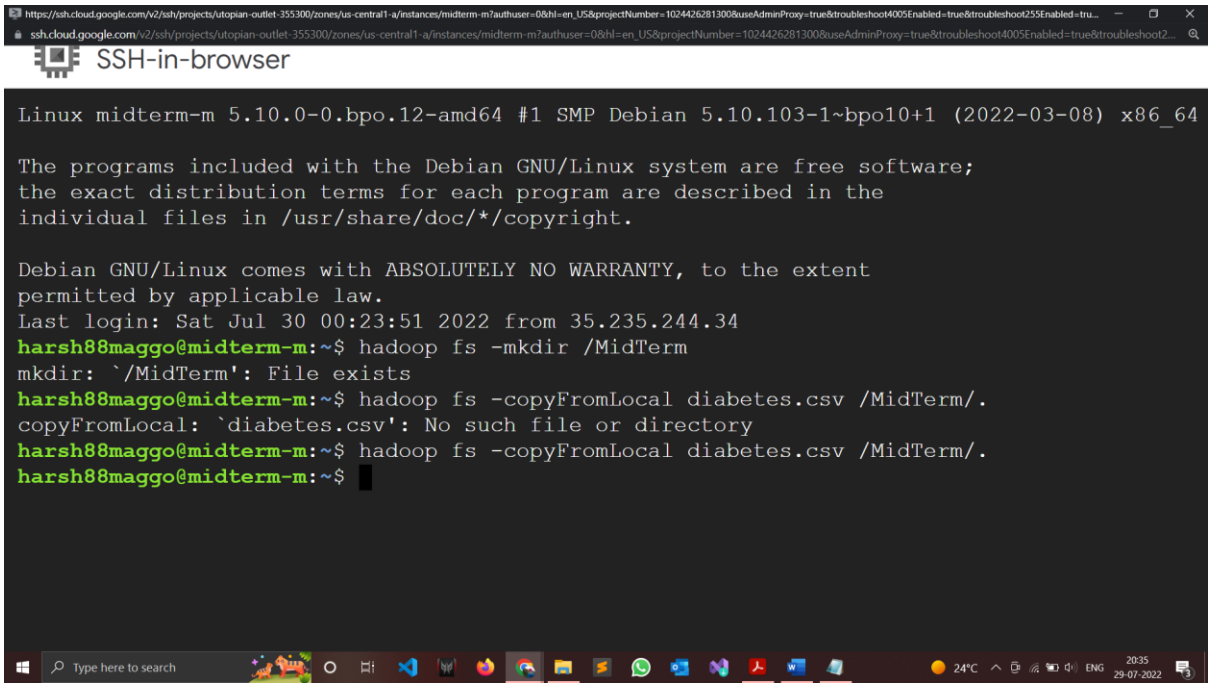
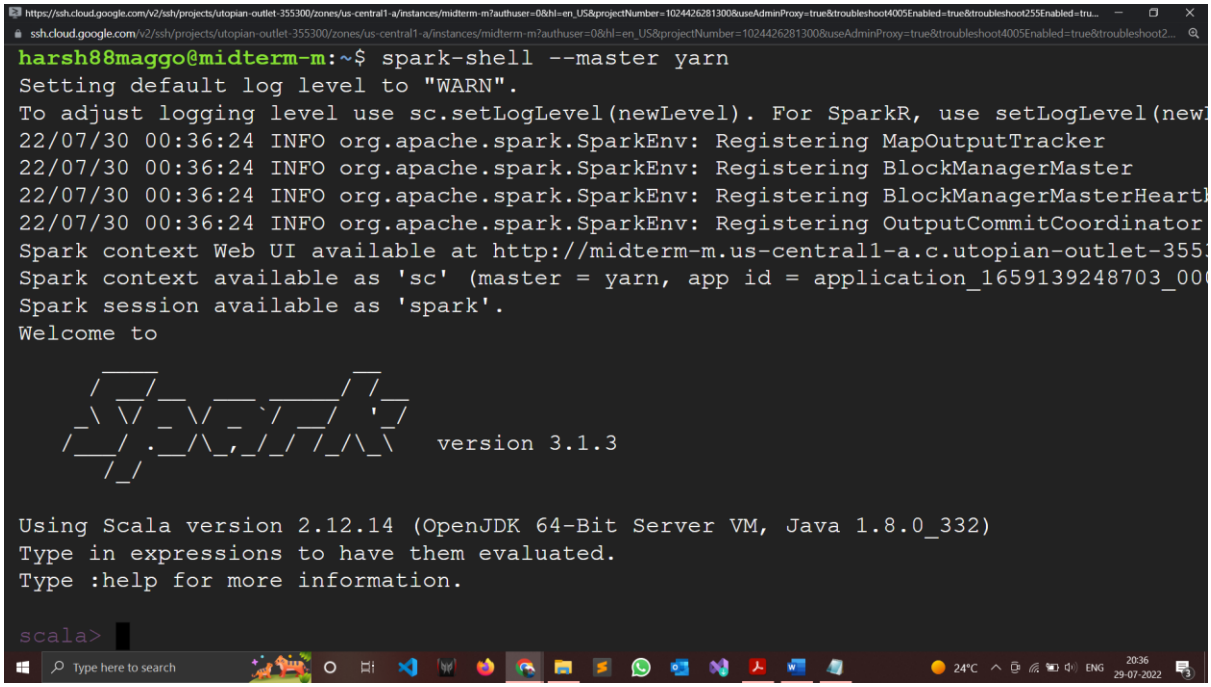
PROJECT EXECUTION REPORT

Problem 1: Diabetes Data (With Random Forest Classification)

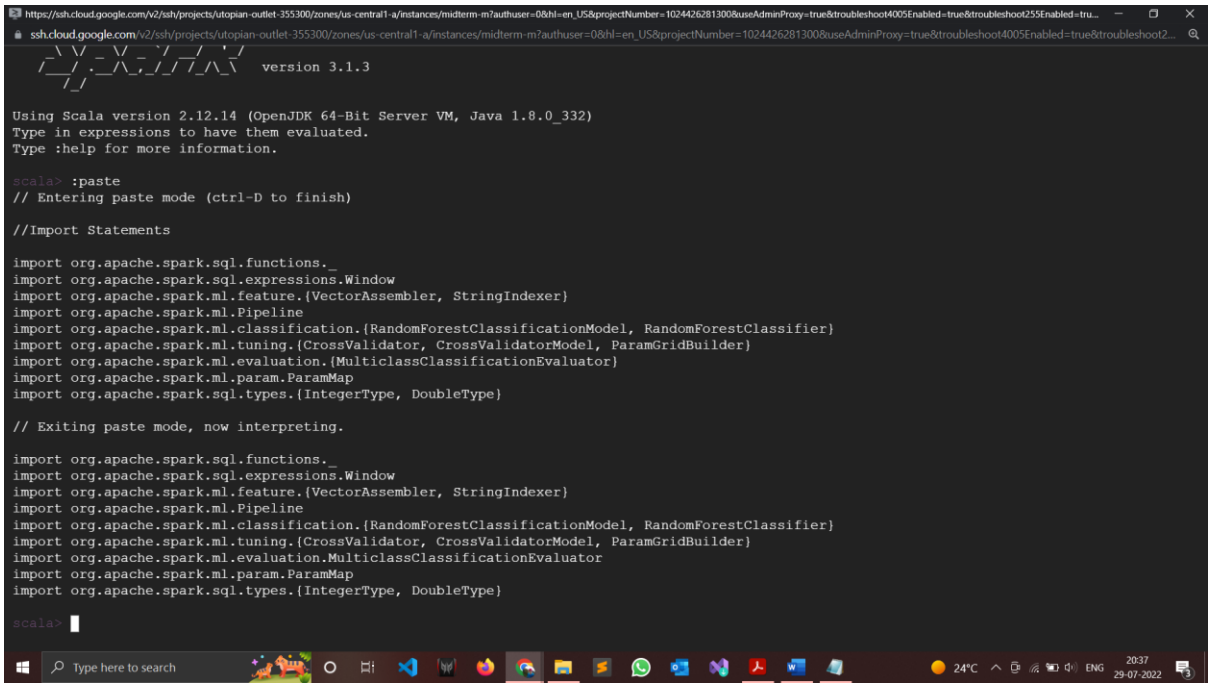
Creating a Hadoop directory and loading the data from the local files system to Hadoop



Running the Spark Shell

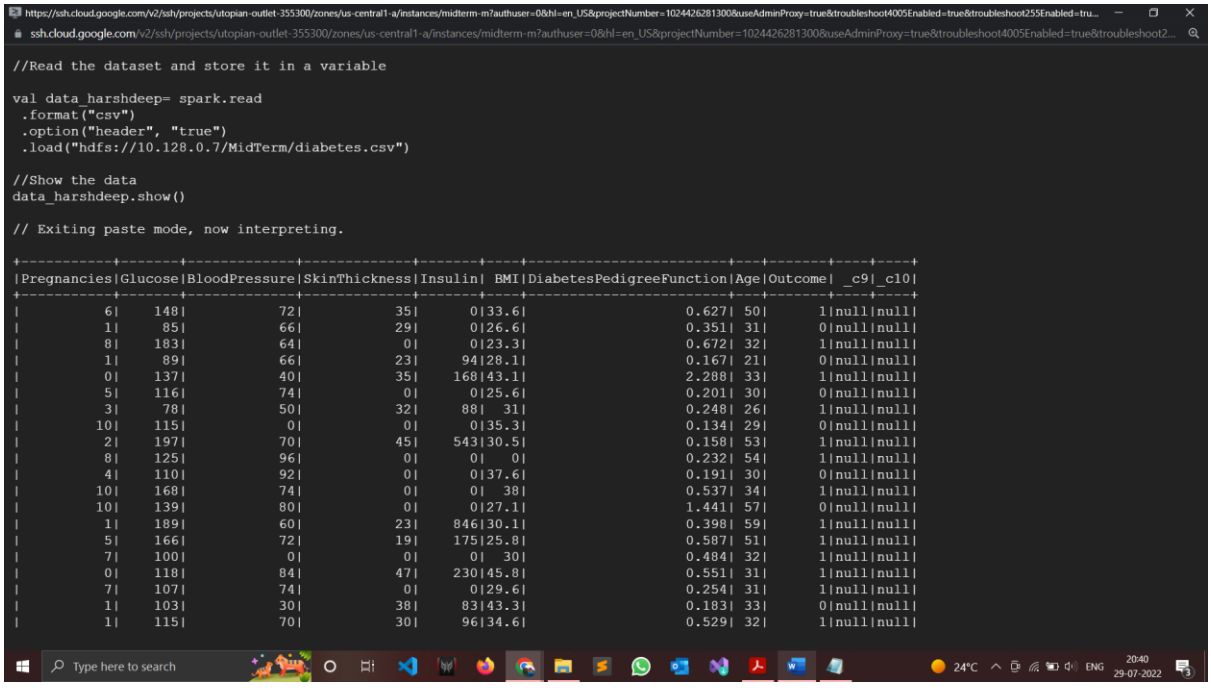


Import statements



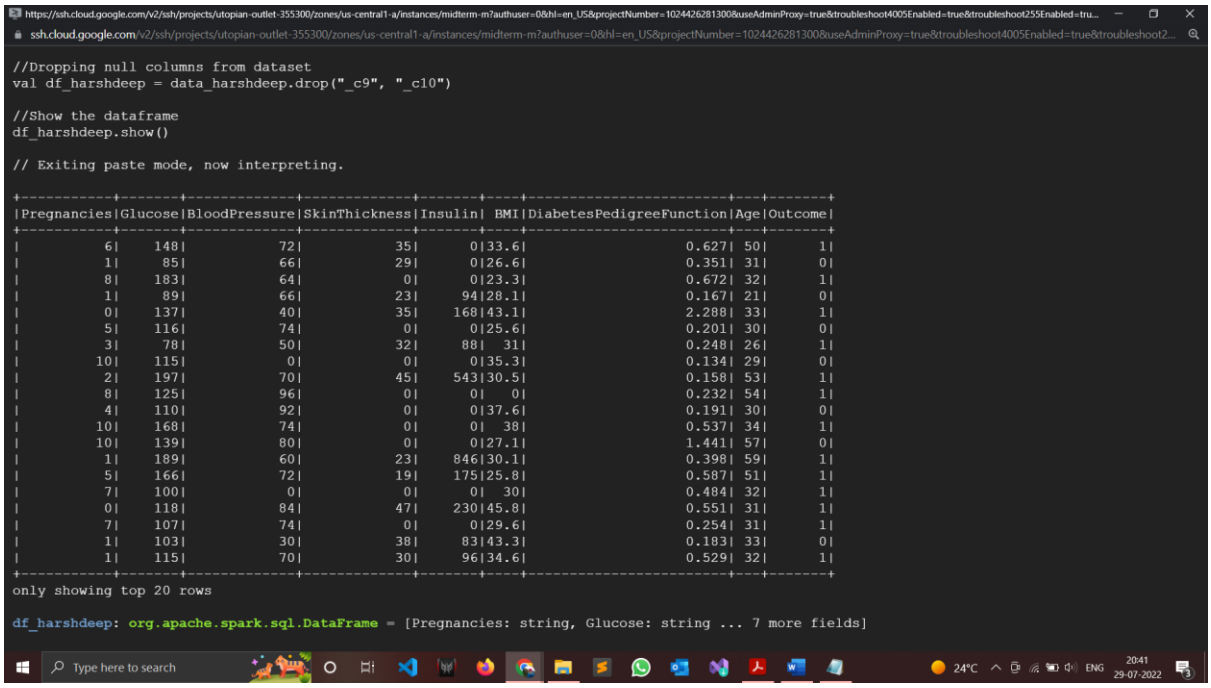
```
https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=tru...
ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot2...
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   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> :paste
// Entering paste mode (ctrl-D to finish)

//Import Statements

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.classification.{RandomForestClassificationModel, RandomForestClassifier}
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}

// 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.classification.{RandomForestClassificationModel, RandomForestClassifier}
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}

scala>
```

Reading the CSV file and checking the data



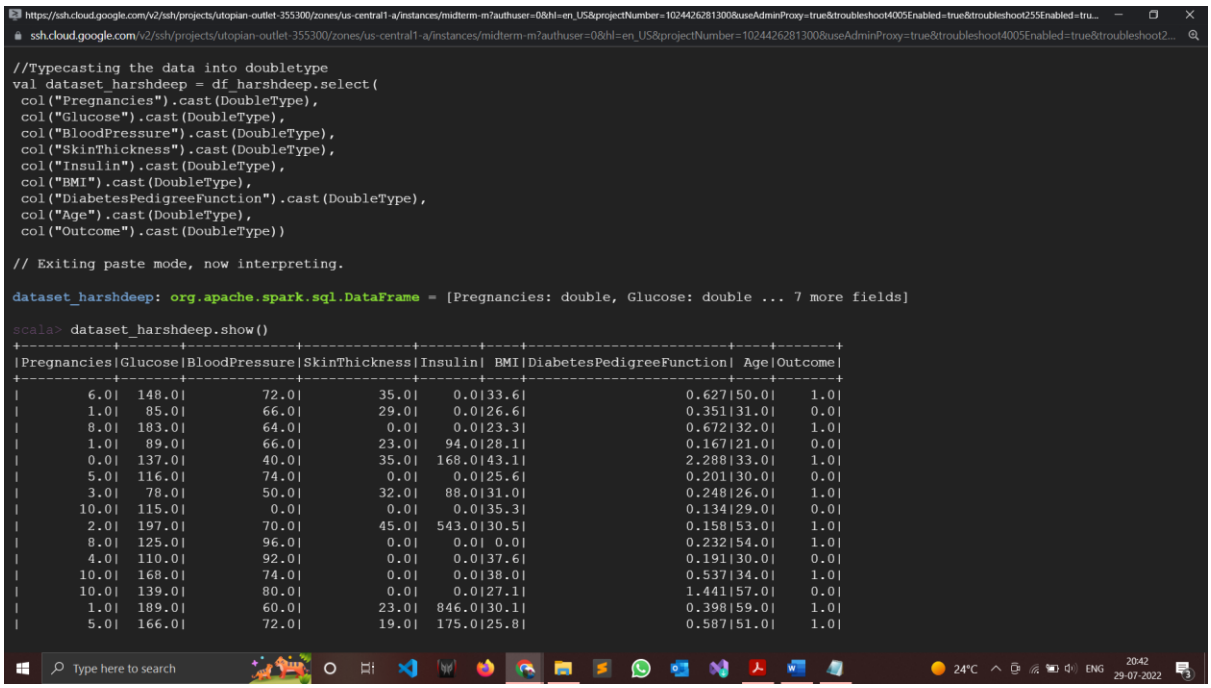
```
https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=tru...
ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m?authuser=0&hl=en_US&projectN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

//Read the dataset and store it in a variable

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

//Show the data
data_harshdeep.show()

// Exiting paste mode, now interpreting.

+-----------+-------+-------------+-------------+-------+----+------------------------+---+-------+----+----+
|Pregnancies|Glucose|BloodPressure|SkinThickness|Insulin| BMI|DiabetesPedigreeFunction|Age|Outcome| _c9|_c10|
+-----------+-------+-------------+-------------+-------+----+------------------------+---+-------+----+----+
|          6|    148|           72|           35|      0|33.6|                   0.627| 50|      1|null|null|
|          1|     85|           66|           29|      0|26.6|                   0.351| 31|      0|null|null|
|          8|    183|           64|            0|      0|23.3|                   0.672| 32|      1|null|null|
|          1|     89|           66|           23|     94|28.1|                   0.167| 21|      0|null|null|
|          0|    137|           40|           35|    168|43.1|                   2.288| 33|      1|null|null|
|          5|    116|           74|            0|      0|25.6|                   0.201| 30|      0|null|null|
|          3|     78|           50|           32|     88|  31|                   0.248| 26|      1|null|null|
|         10|    115|            0|            0|      0|35.3|                   0.134| 29|      0|null|null|
|          2|    197|           70|           45|    543|30.5|                   0.158| 53|      1|null|null|
|          8|    125|           96|            0|      0|   0|                   0.232| 54|      1|null|null|
|          4|    110|           92|            0|      0|37.6|                   0.191| 30|      0|null|null|
|         10|    168|           74|            0|      0|  38|                   0.537| 34|      1|null|null|
|         10|    139|           80|            0|      0|27.1|                   1.441| 57|      0|null|null|
|          1|    189|           60|           23|    846|30.1|                   0.398| 59|      1|null|null|
|          5|    166|           72|           19|    175|25.8|                   0.587| 51|      1|null|null|
|          7|    100|            0|            0|      0|  30|                   0.484| 32|      1|null|null|
|          0|    118|           84|           47|    230|45.8|                   0.551| 31|      1|null|null|
|          7|    107|           74|            0|      0|29.6|                   0.254| 31|      1|null|null|
|          1|    103|           30|           38|     83|43.3|                   0.183| 33|      0|null|null|
|          1|    115|           70|           30|     96|34.6|                   0.529| 32|      1|null|null|
```

Dropping null columns from the dataset



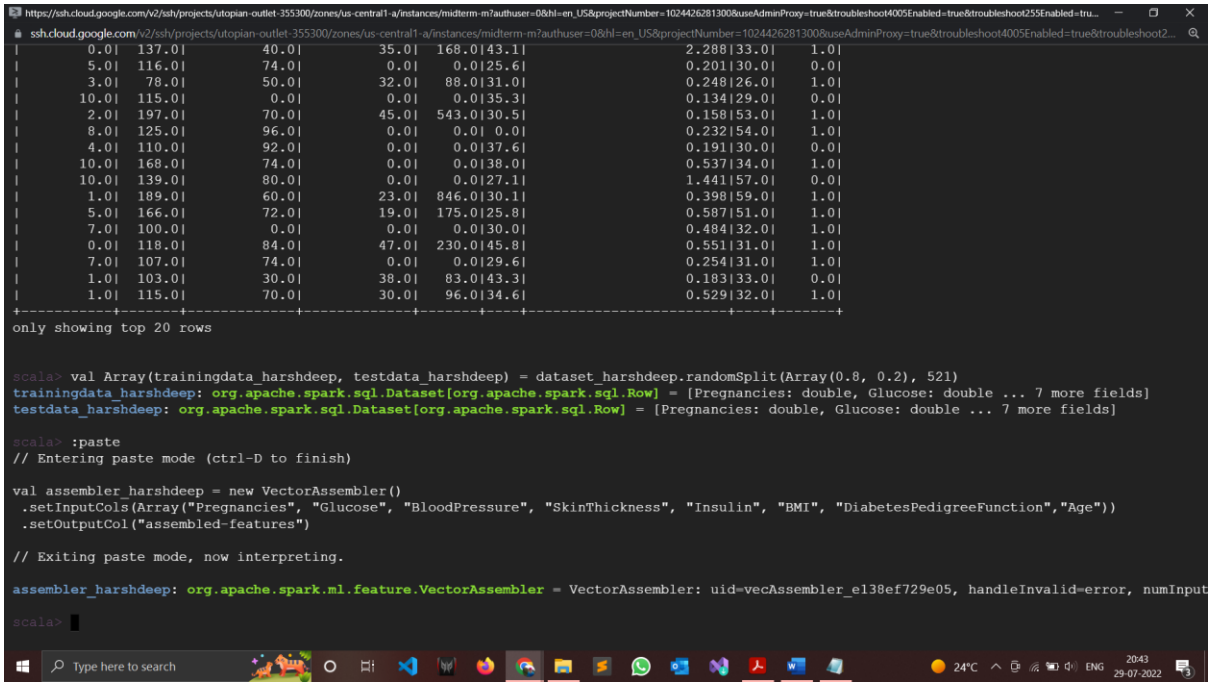
```
//Dropping null columns from dataset
val df_harshdeep = data_harshdeep.drop("_c9", "_c10")

//Show the dataframe
df_harshdeep.show()

// Exiting paste mode, now interpreting.

+-----------+-------+-------------+-------------+-------+----+------------------------+---+-------+
|Pregnancies|Glucose|BloodPressure|SkinThickness|Insulin| BMI|DiabetesPedigreeFunction|Age|Outcome|
+-----------+-------+-------------+-------------+-------+----+------------------------+---+-------+
|          6|    148|           72|           35|      0|33.6|                   0.627| 50|      1|
|          1|     85|           66|           29|      0|26.6|                   0.351| 31|      0|
|          8|    183|           64|            0|      0|23.3|                   0.672| 32|      1|
|          1|     89|           66|           23|     94|28.1|                   0.167| 21|      0|
|          0|    137|           40|           35|    168|43.1|                   2.288| 33|      1|
|          5|    116|           74|            0|      0|25.6|                   0.201| 30|      0|
|          3|     78|           50|           32|     88|31.0|                   0.248| 26|      1|
|         10|    115|            0|            0|      0|35.3|                   0.134| 29|      0|
|          2|    197|           70|           45|    543|30.5|                   0.158| 53|      1|
|          8|    125|           96|            0|      0| 0.0|                   0.232| 54|      1|
|          4|    110|           92|            0|      0|37.6|                   0.191| 30|      0|
|         10|    168|           74|            0|      0|38.0|                   0.537| 34|      1|
|         10|    139|           80|            0|      0|27.1|                   1.441| 57|      0|
|          1|    189|           60|           23|    846|30.1|                   0.398| 59|      1|
|          5|    166|           72|           19|    175|25.8|                   0.587| 51|      1|
|          7|    100|            0|            0|      0|30.0|                   0.484| 32|      1|
|          0|    118|           84|           47|    230|45.8|                   0.551| 31|      1|
|          7|    107|           74|            0|      0|29.6|                   0.254| 31|      1|
|          1|    103|           30|           38|     83|43.3|                   0.183| 33|      0|
|          1|    115|           70|           30|     96|34.6|                   0.529| 32|      1|
+-----------+-------+-------------+-------------+-------+----+------------------------+---+-------+
only showing top 20 rows

df_harshdeep: org.apache.spark.sql.DataFrame = [Pregnancies: string, Glucose: string ... 7 more fields]
```

Typecasting the data into type Double for our model training

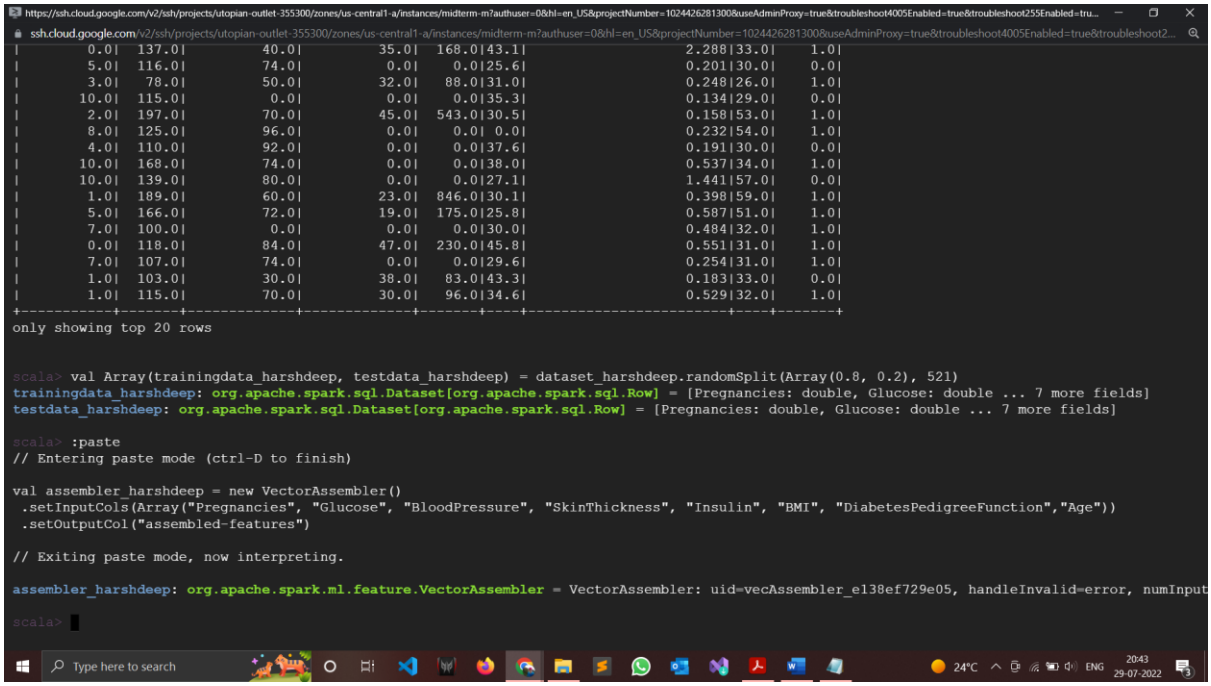


```
//Typecasting the data into doubletype
val dataset_harshdeep = df_harshdeep.select(
 col("Pregnancies").cast(DoubleType),
 col("Glucose").cast(DoubleType),
 col("BloodPressure").cast(DoubleType),
 col("SkinThickness").cast(DoubleType),
 col("Insulin").cast(DoubleType),
 col("BMI").cast(DoubleType),
 col("DiabetesPedigreeFunction").cast(DoubleType),
 col("Age").cast(DoubleType),
 col("Outcome").cast(DoubleType))

// Exiting paste mode, now interpreting.

dataset_harshdeep: org.apache.spark.sql.DataFrame = [Pregnancies: double, Glucose: double ... 7 more fields]

scala> dataset_harshdeep.show()
+-----------+-------+-------------+-------------+-------+----+------------------------+----+-------+
|Pregnancies|Glucose|BloodPressure|SkinThickness|Insulin| BMI|DiabetesPedigreeFunction| Age|Outcome|
+-----------+-------+-------------+-------------+-------+----+------------------------+----+-------+
|        6.0|  148.0|         72.0|         35.0|    0.0|33.6|                   0.627|50.0|    1.0|
|        1.0|   85.0|         66.0|         29.0|    0.0|26.6|                   0.351|31.0|    0.0|
|        8.0|  183.0|         64.0|          0.0|    0.0|23.3|                   0.672|32.0|    1.0|
|        1.0|   89.0|         66.0|         23.0|   94.0|28.1|                   0.167|21.0|    0.0|
|        0.0|  137.0|         40.0|         35.0|  168.0|43.1|                   2.288|33.0|    1.0|
|        5.0|  116.0|         74.0|          0.0|    0.0|25.6|                   0.201|30.0|    0.0|
|        3.0|   78.0|         50.0|         32.0|   88.0|31.0|                   0.248|26.0|    1.0|
|       10.0|  115.0|          0.0|          0.0|    0.0|35.3|                   0.134|29.0|    0.0|
|        2.0|  197.0|         70.0|         45.0|  543.0|30.5|                   0.158|53.0|    1.0|
|        8.0|  125.0|         96.0|          0.0|    0.0| 0.0|                   0.232|54.0|    1.0|
|        4.0|  110.0|         92.0|          0.0|    0.0|37.6|                   0.191|30.0|    0.0|
|       10.0|  168.0|         74.0|          0.0|    0.0|38.0|                   0.537|34.0|    1.0|
|       10.0|  139.0|         80.0|          0.0|    0.0|27.1|                   1.441|57.0|    0.0|
|        1.0|  189.0|         60.0|         23.0|  846.0|30.1|                   0.398|59.0|    1.0|
|        5.0|  166.0|         72.0|         19.0|  175.0|25.8|                   0.587|51.0|    1.0|
```

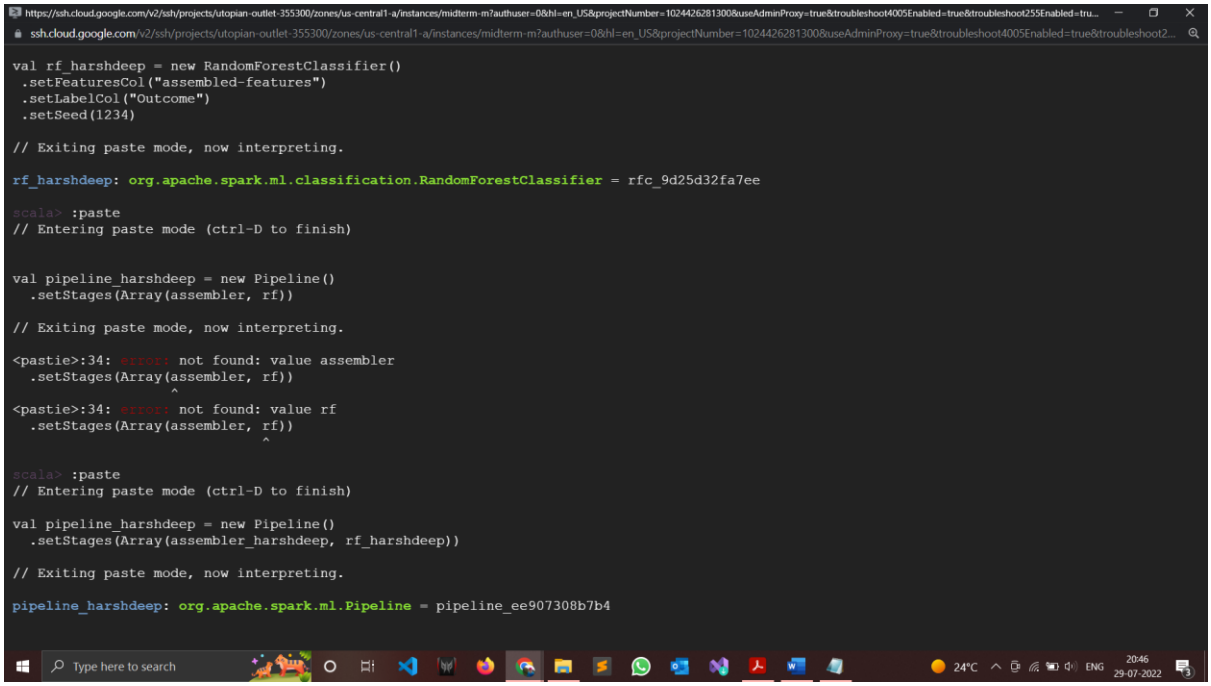
Split the dataset into train and test



Assembling the features using VectorAssembler

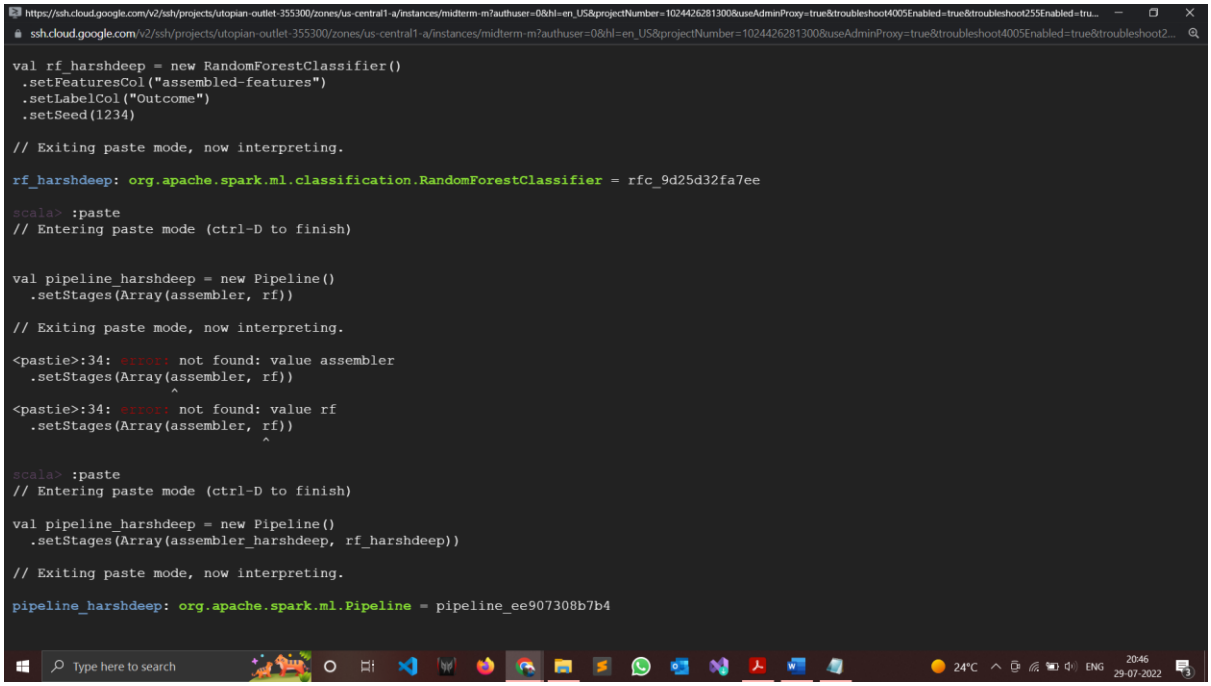


Creating the Random Forest Object and passing the features



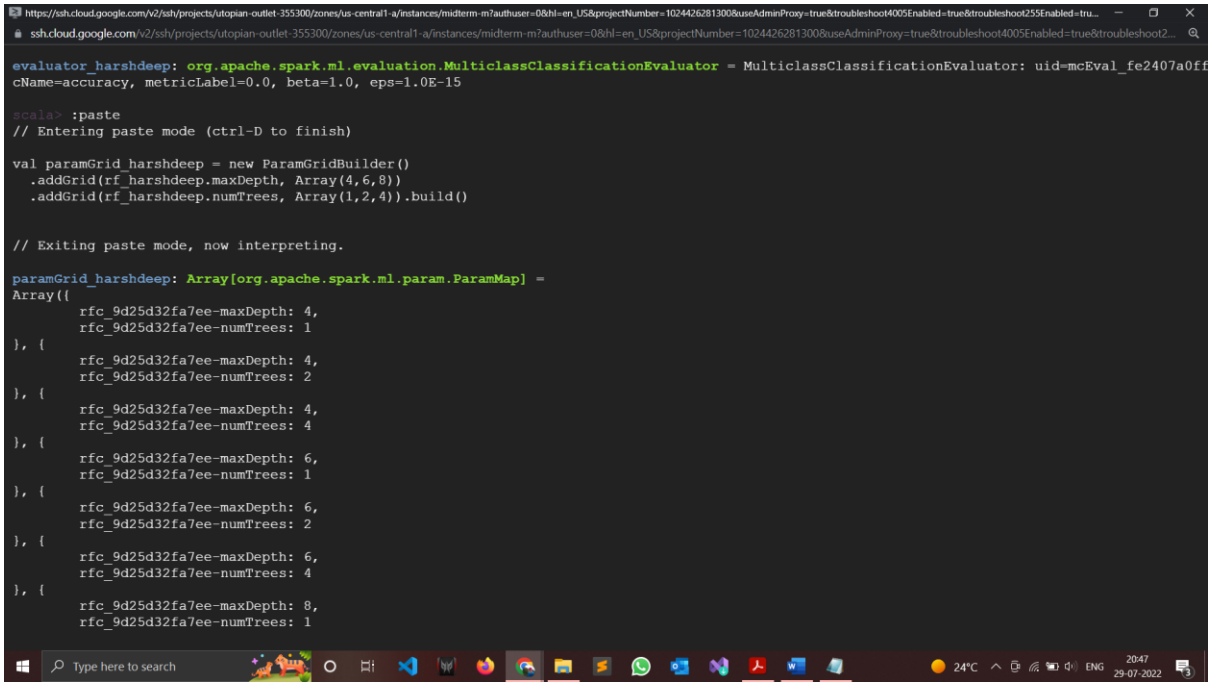
```
val rf_harshdeep = new RandomForestClassifier()
 .setFeaturesCol("assembled-features")
 .setLabelCol("Outcome")
 .setSeed(1234)

// 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

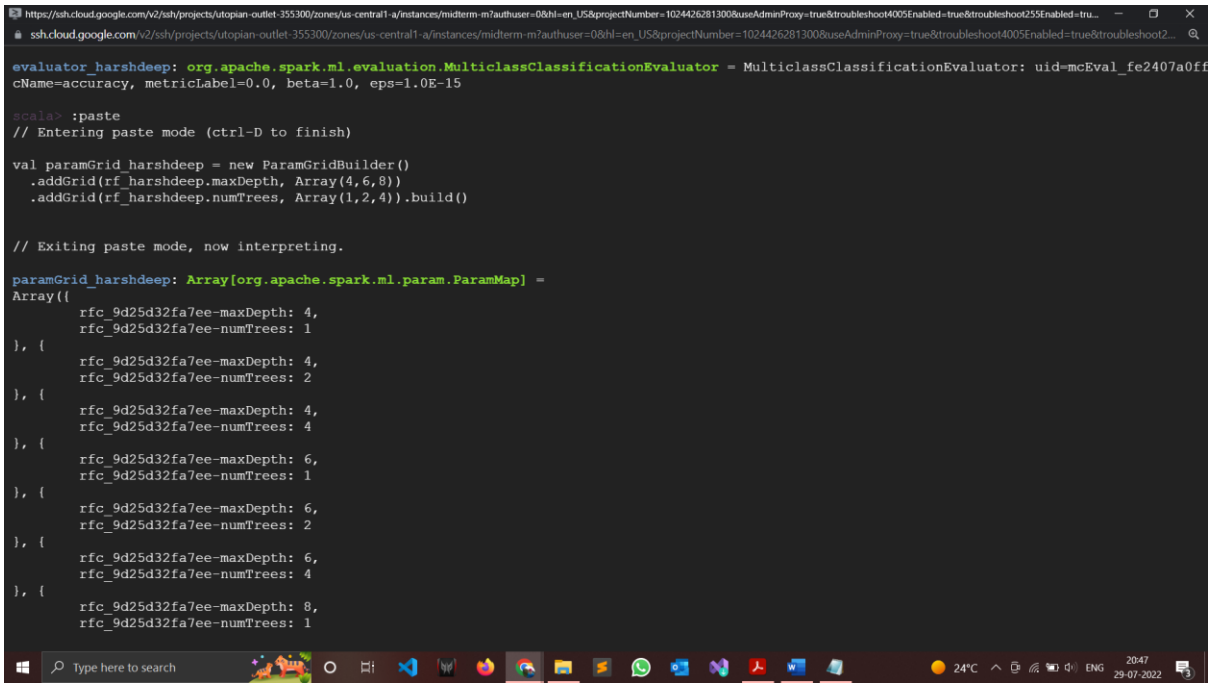


```
val rf_harshdeep = new RandomForestClassifier()
 .setFeaturesCol("assembled-features")
 .setLabelCol("Outcome")
 .setSeed(1234)

// 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
```

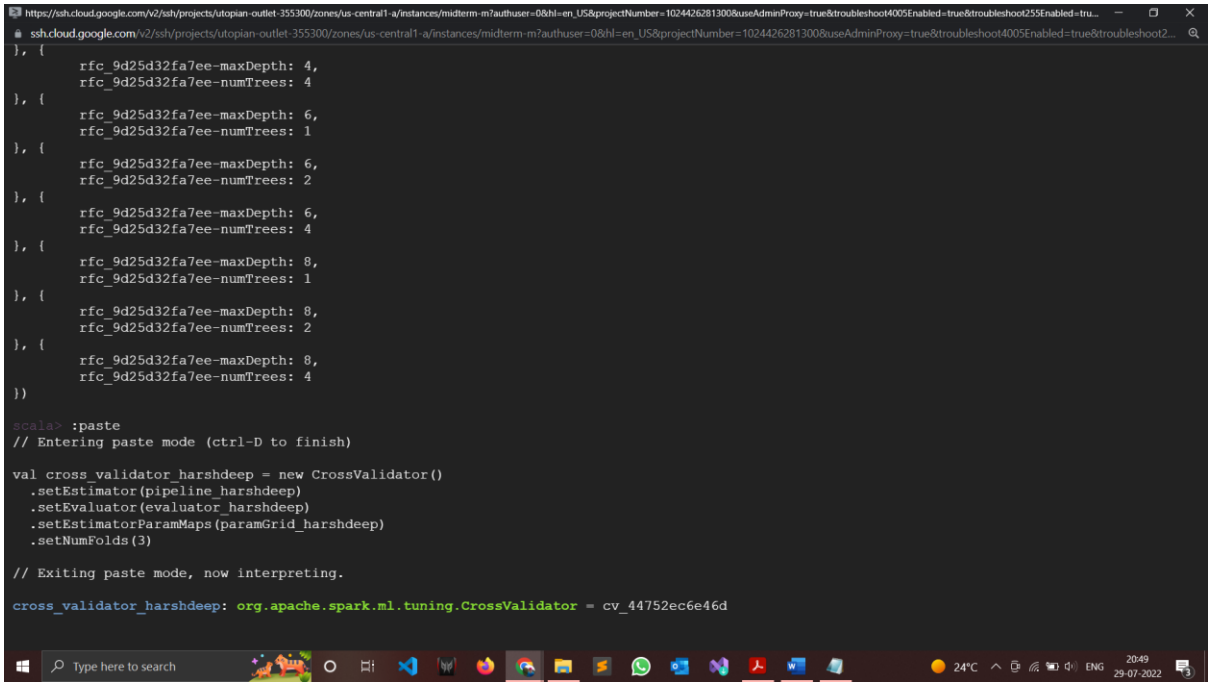
Evaluator for our model



Setting the hyperparameters

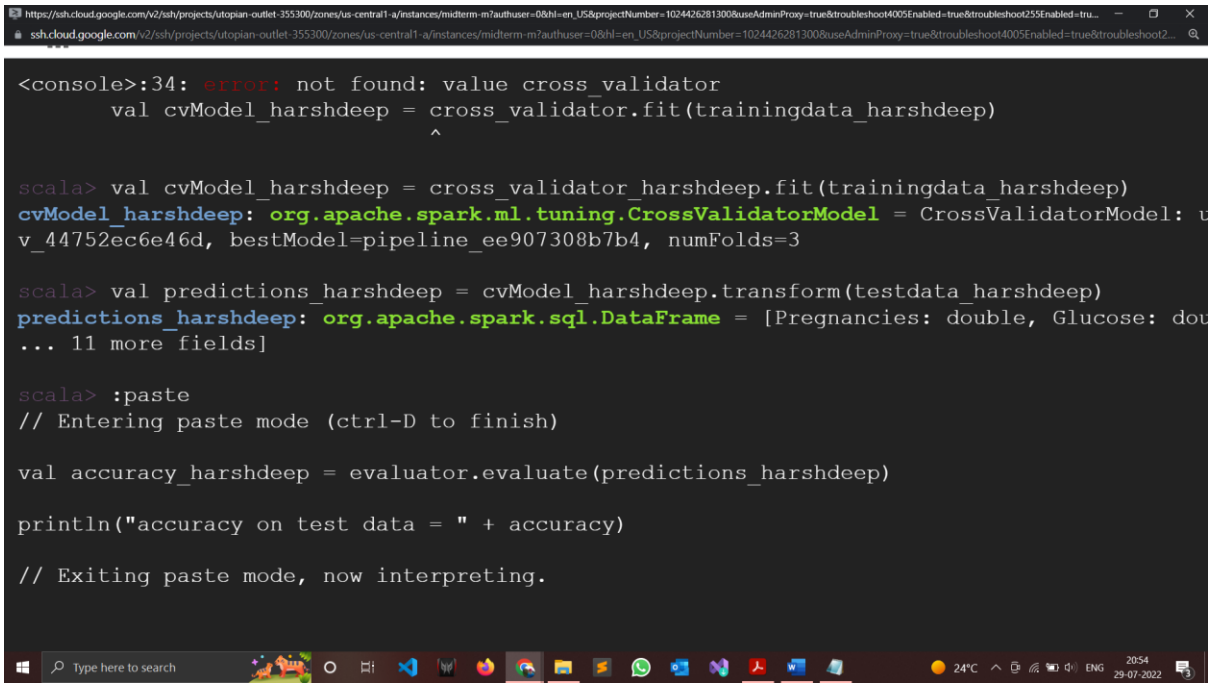


Creating the Cross Validator



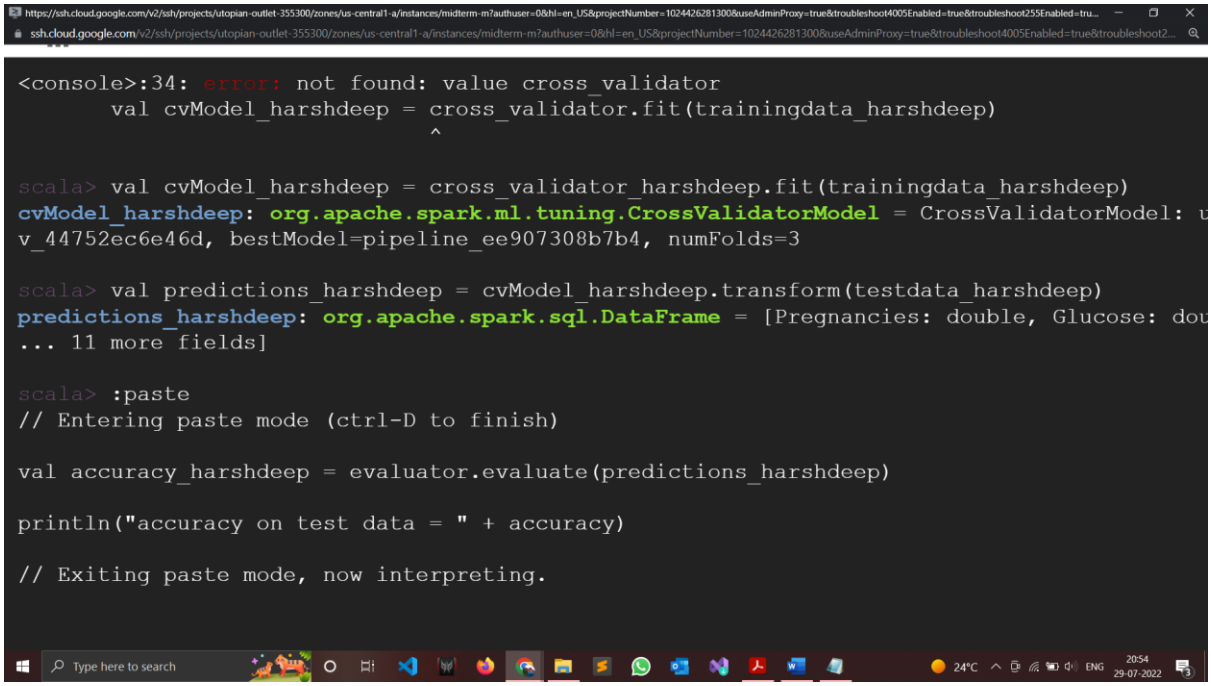
```
}, {
        rfc_9d25d32fa7ee-maxDepth: 4,
        rfc_9d25d32fa7ee-numTrees: 4
}, {
        rfc_9d25d32fa7ee-maxDepth: 6,
        rfc_9d25d32fa7ee-numTrees: 1
}, {
        rfc_9d25d32fa7ee-maxDepth: 6,
        rfc_9d25d32fa7ee-numTrees: 2
}, {
        rfc_9d25d32fa7ee-maxDepth: 6,
        rfc_9d25d32fa7ee-numTrees: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

Training our model

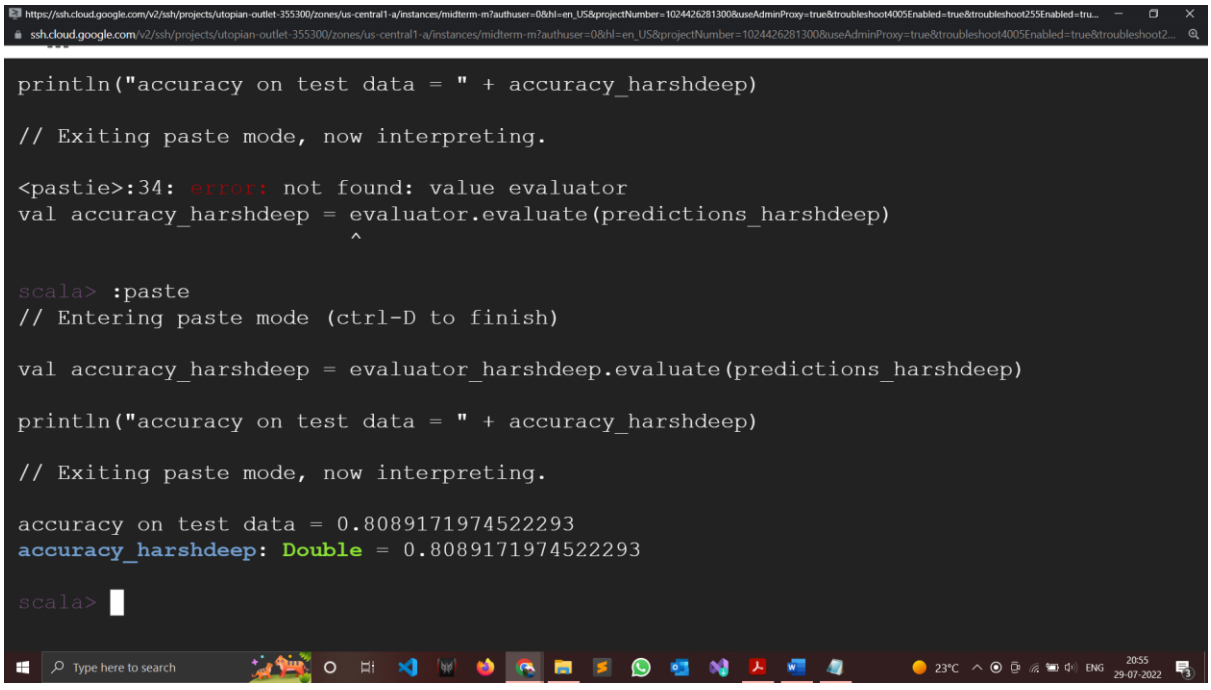


```
<console>:34: error: not found: value cross_validator
       val cvModel_harshdeep = cross_validator.fit(trainingdata_harshdeep)
                               ^

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.
```

Prediction using testdata



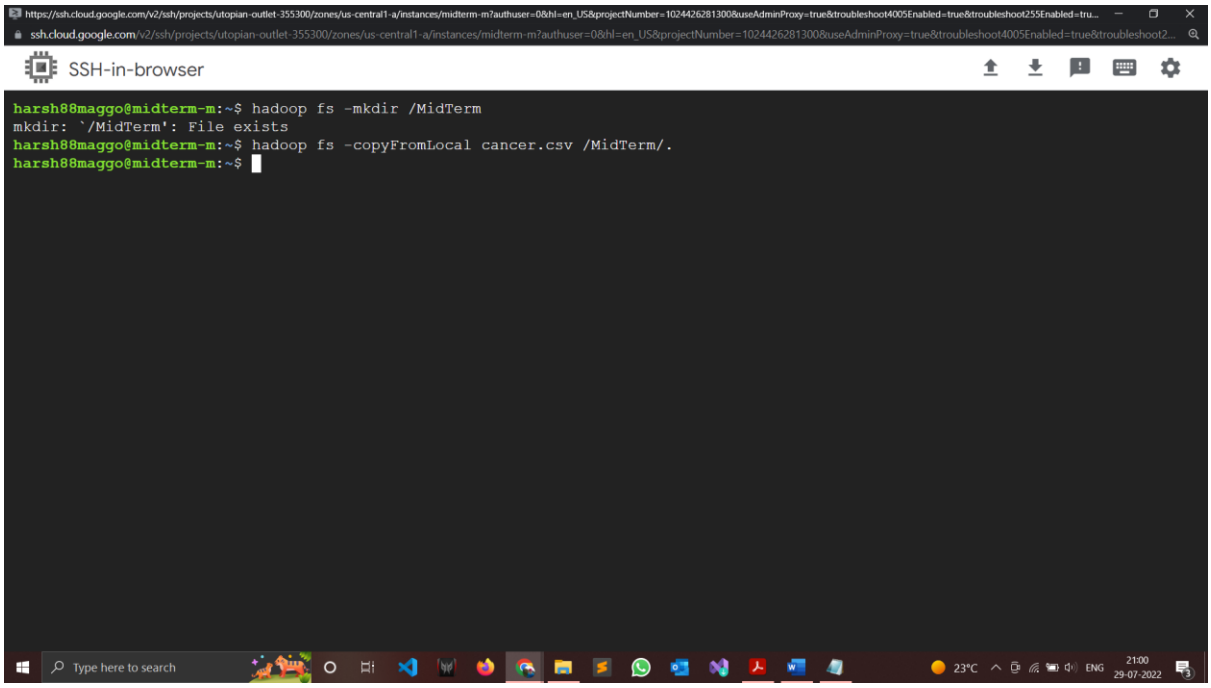
Evaluating the performance of our model



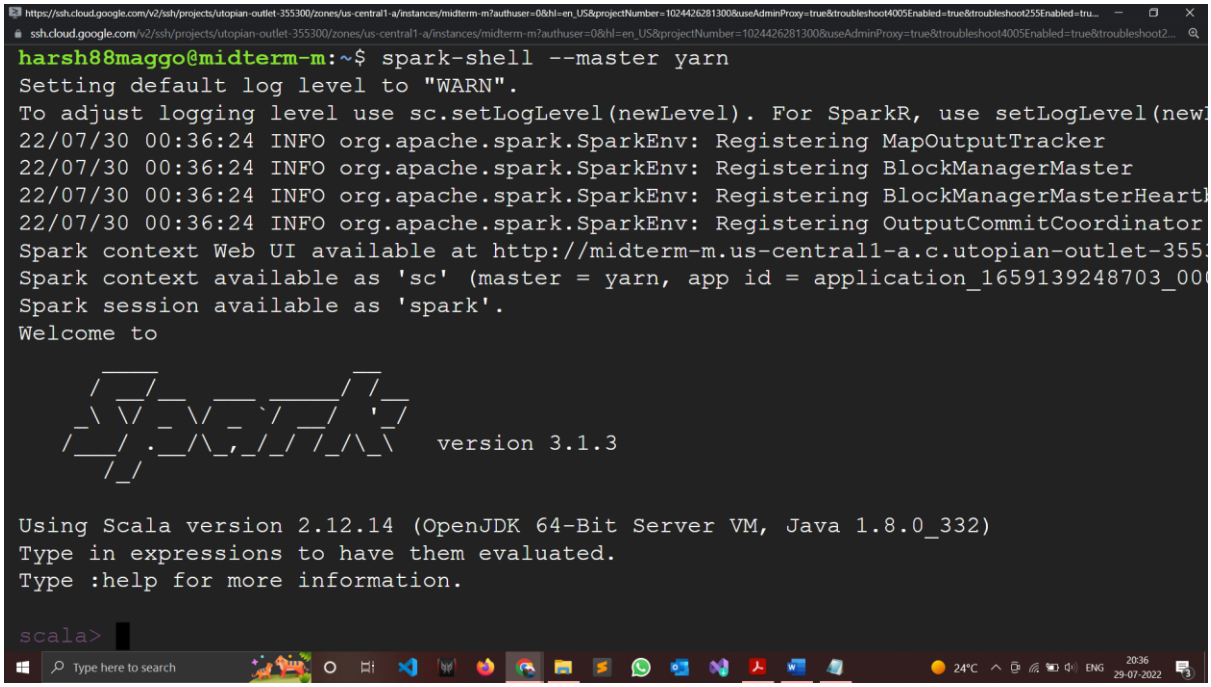
ACCURACY: 80.89%

Problem 2: Cancer Data (With Logistic Regression)

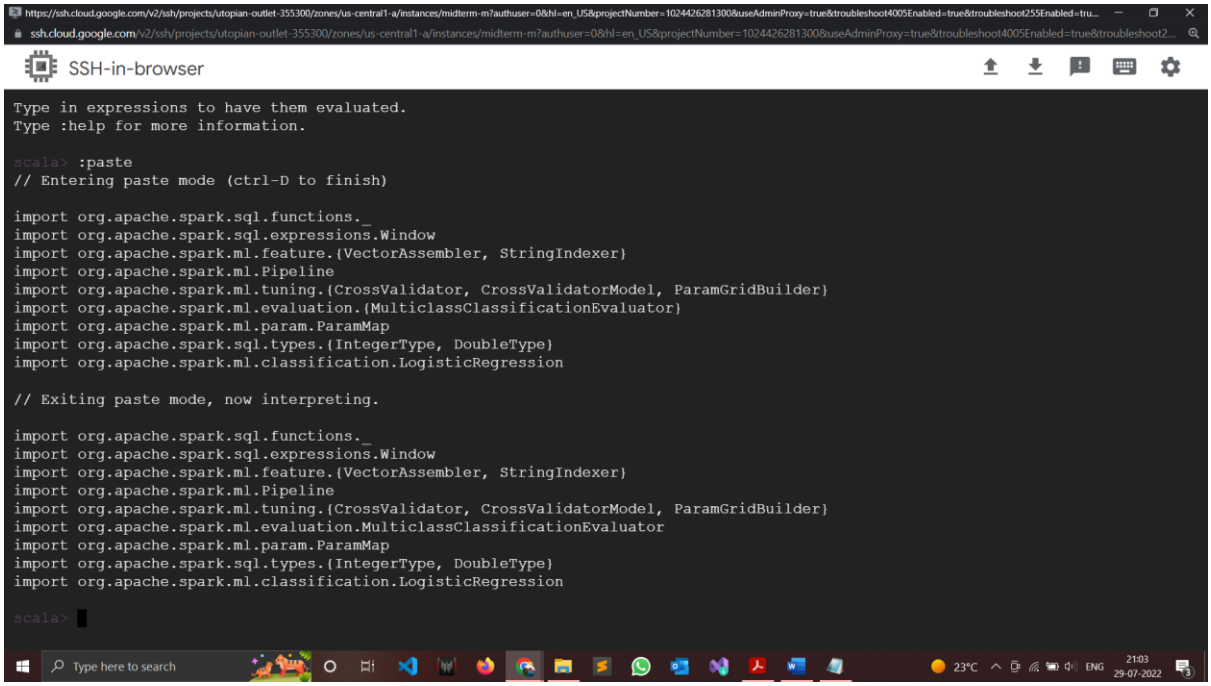
Creating a Hadoop directory and loading the data from the local files system to Hadoop



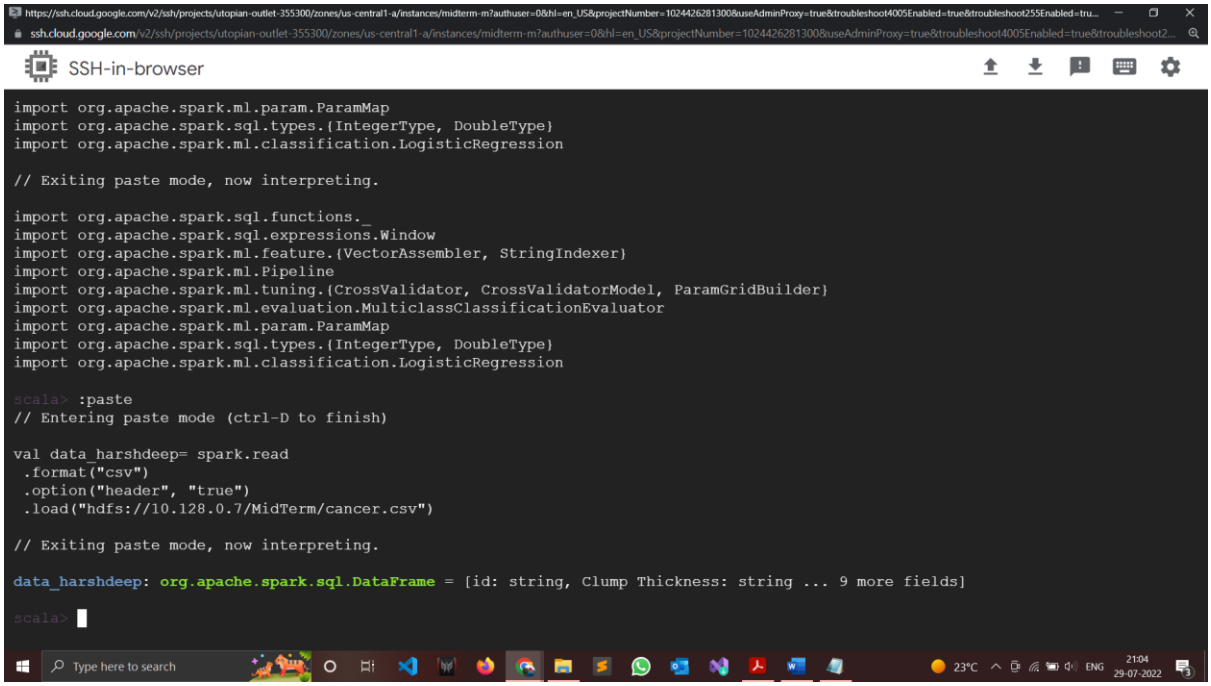
Running the Spark Shell



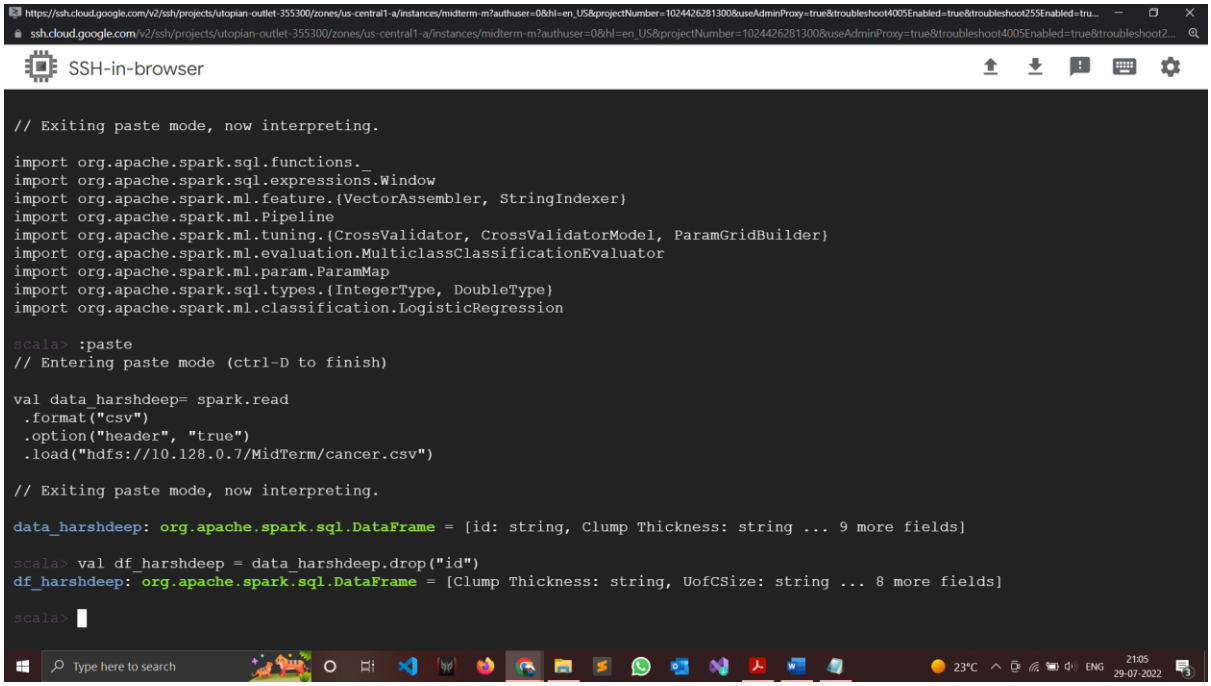
Import statements



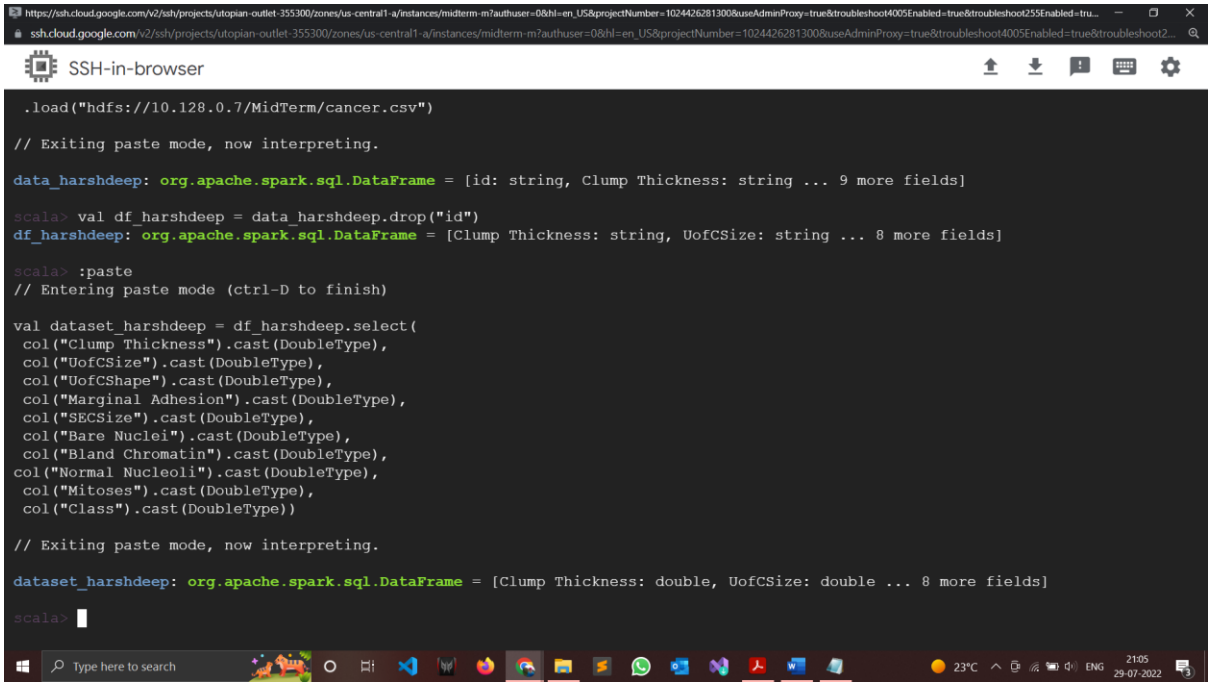
Reading the CSV file



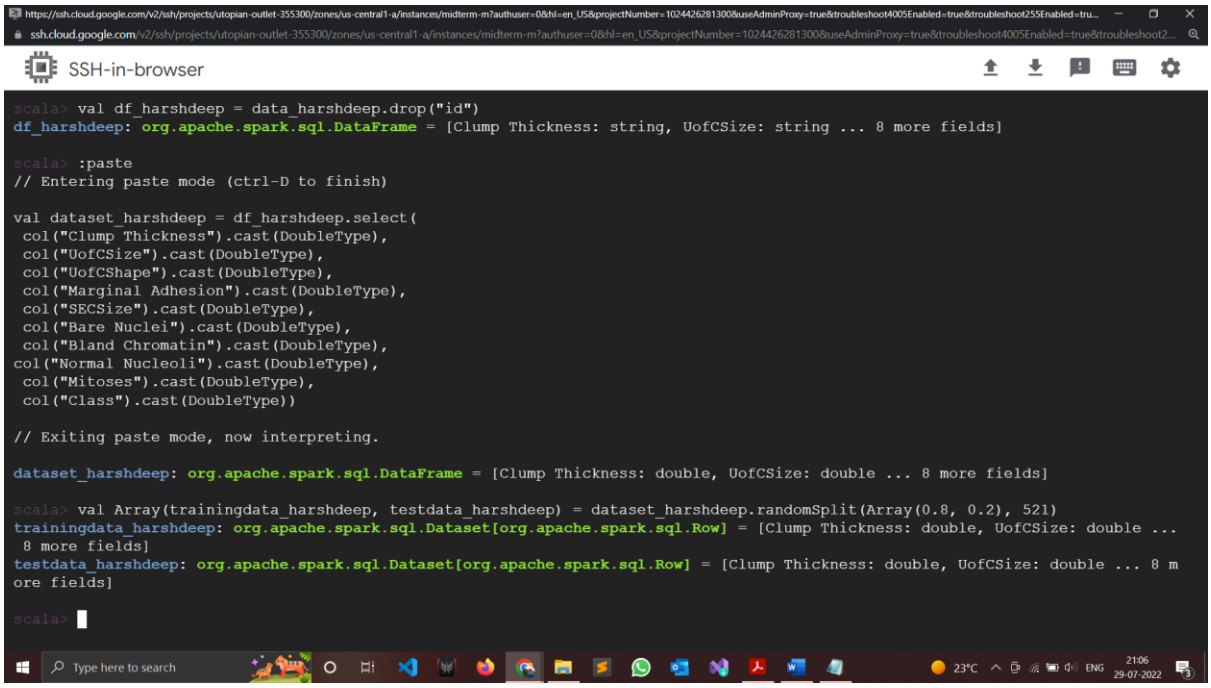
Dropping the ‘ID’ column since it won’t help us in prediction



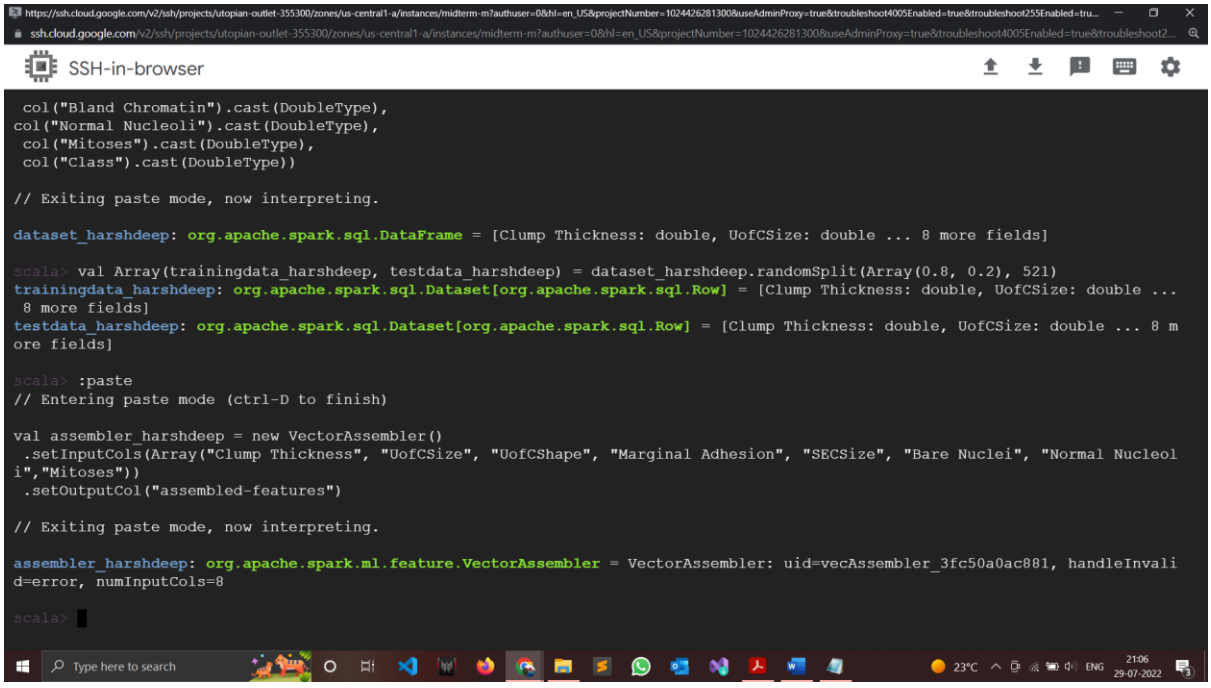
Typecasting the data into type Double for our model training



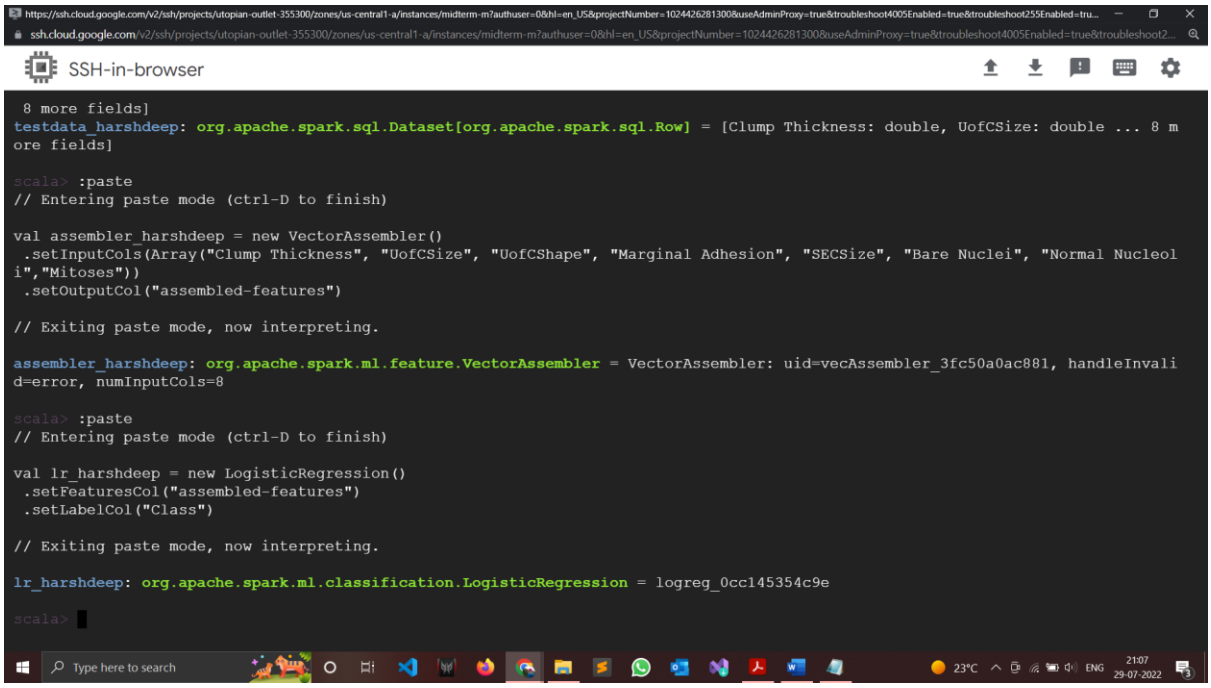
Split the dataset into train and test



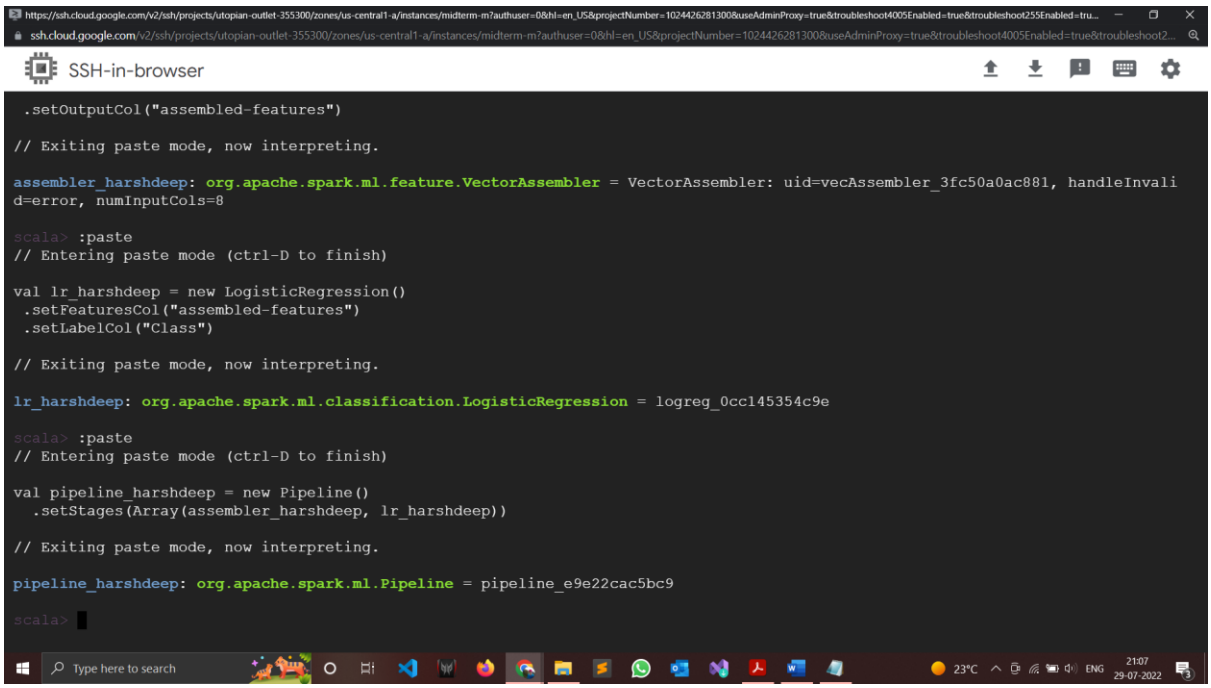
Assembling the features using VectorAssembler



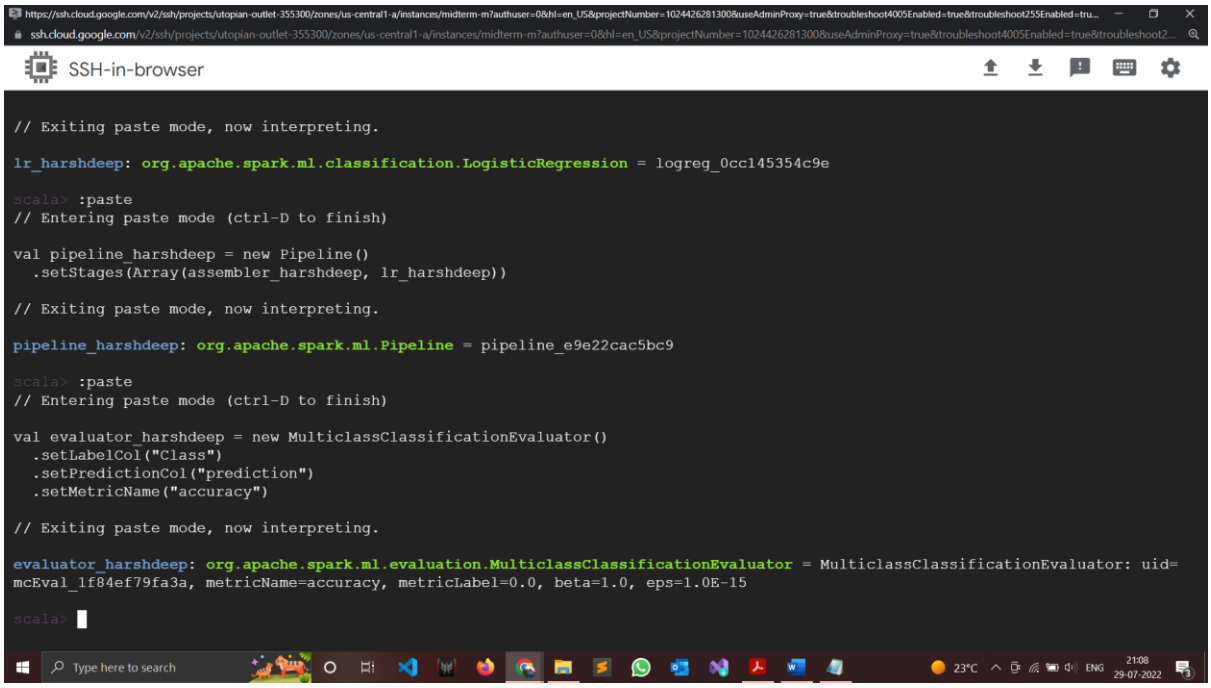
Creating the Logistic Regression Object and passing the features



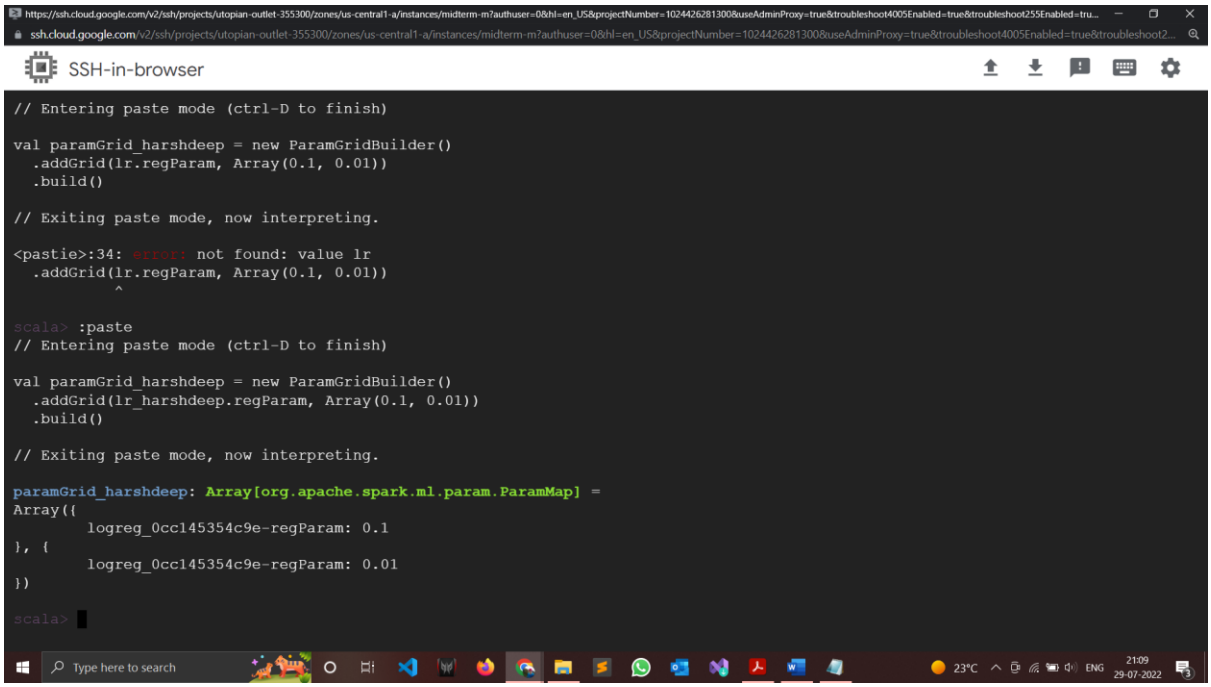
Creating the pipeling



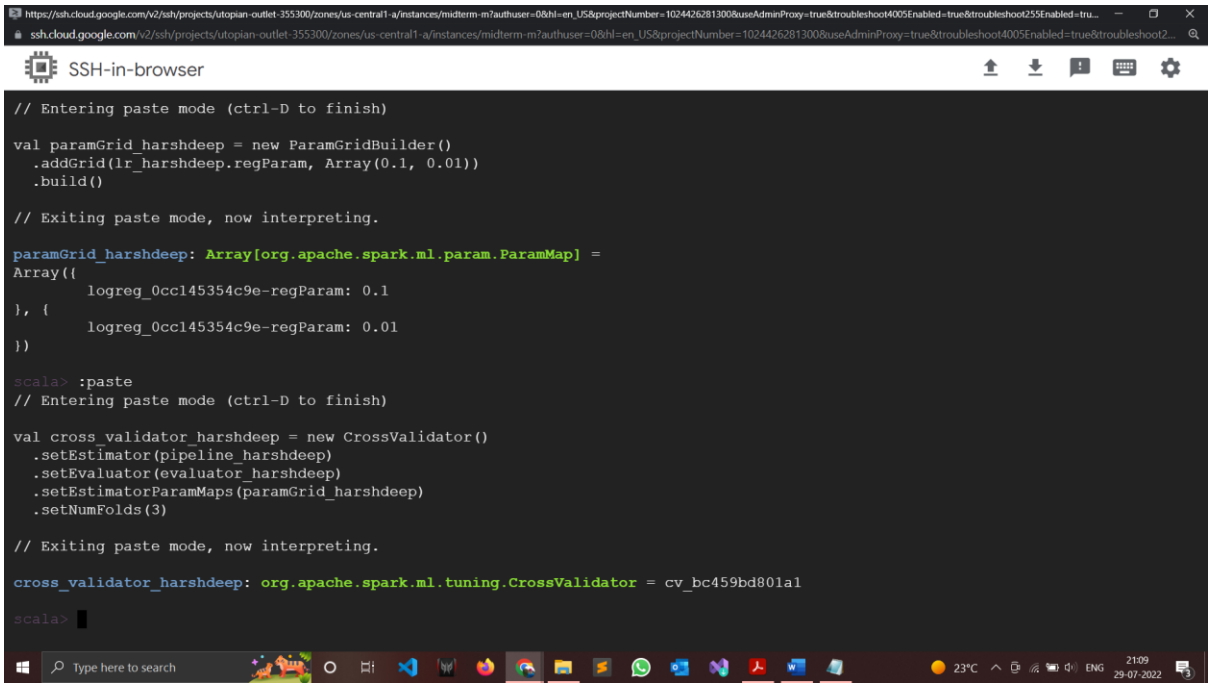
Evaluator for our model



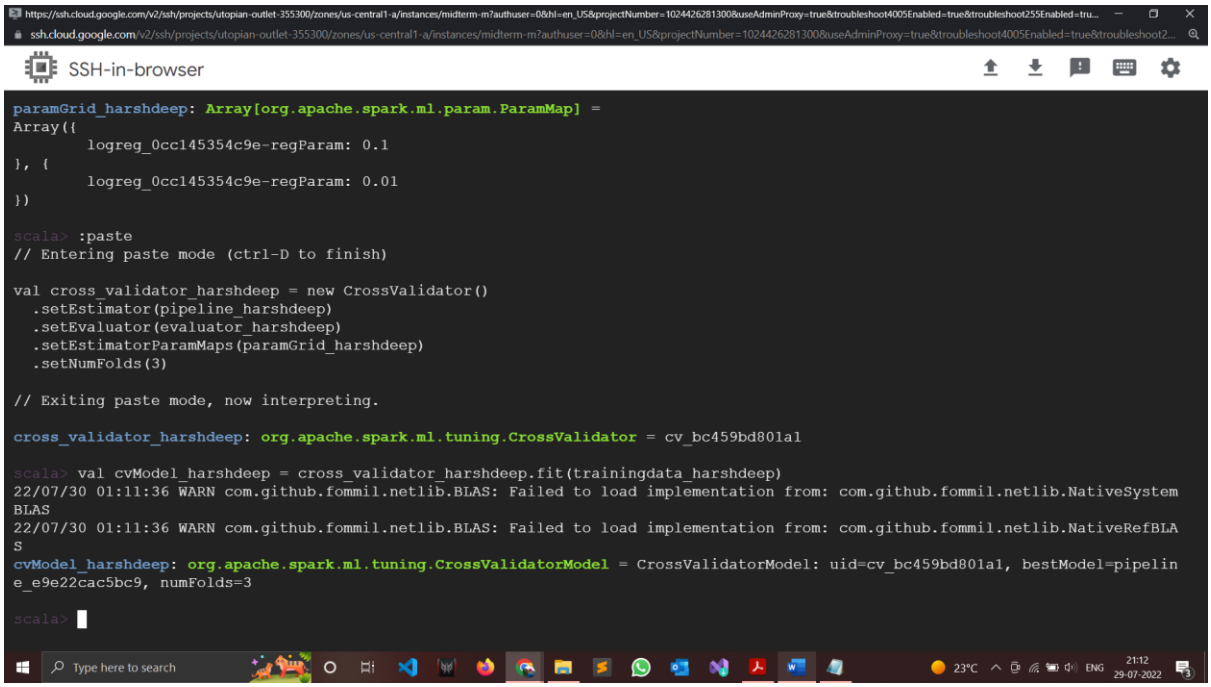
Setting the hyperparameters



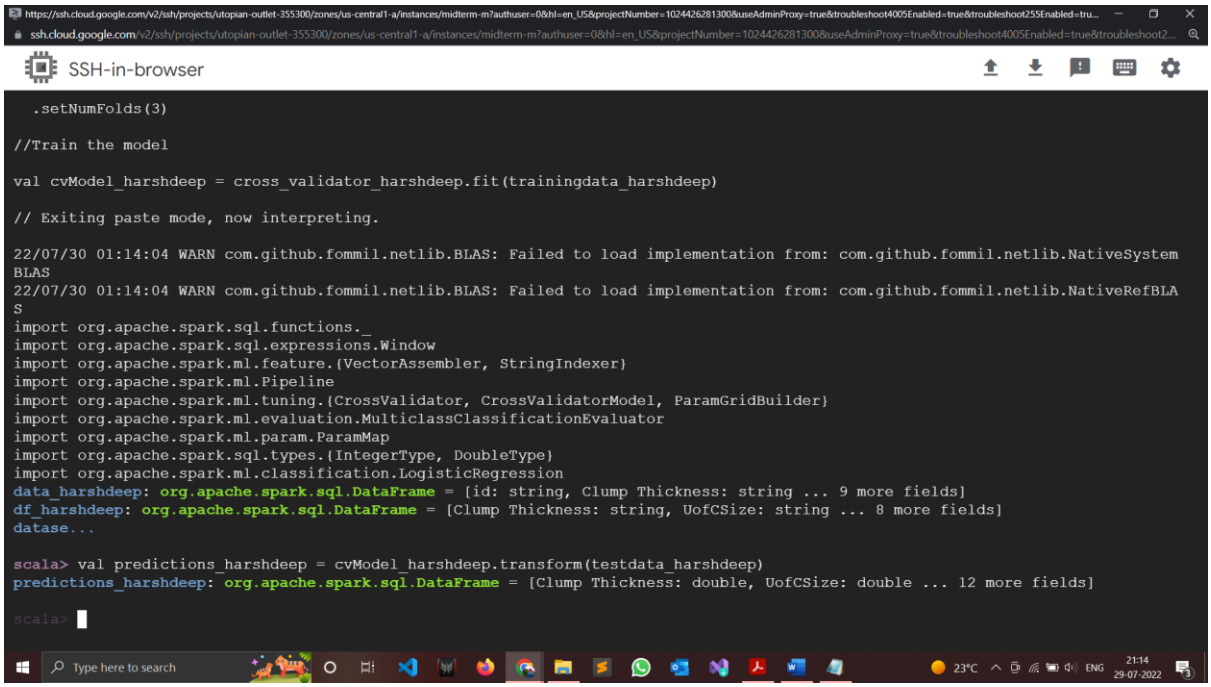
Creating the Cross Validator



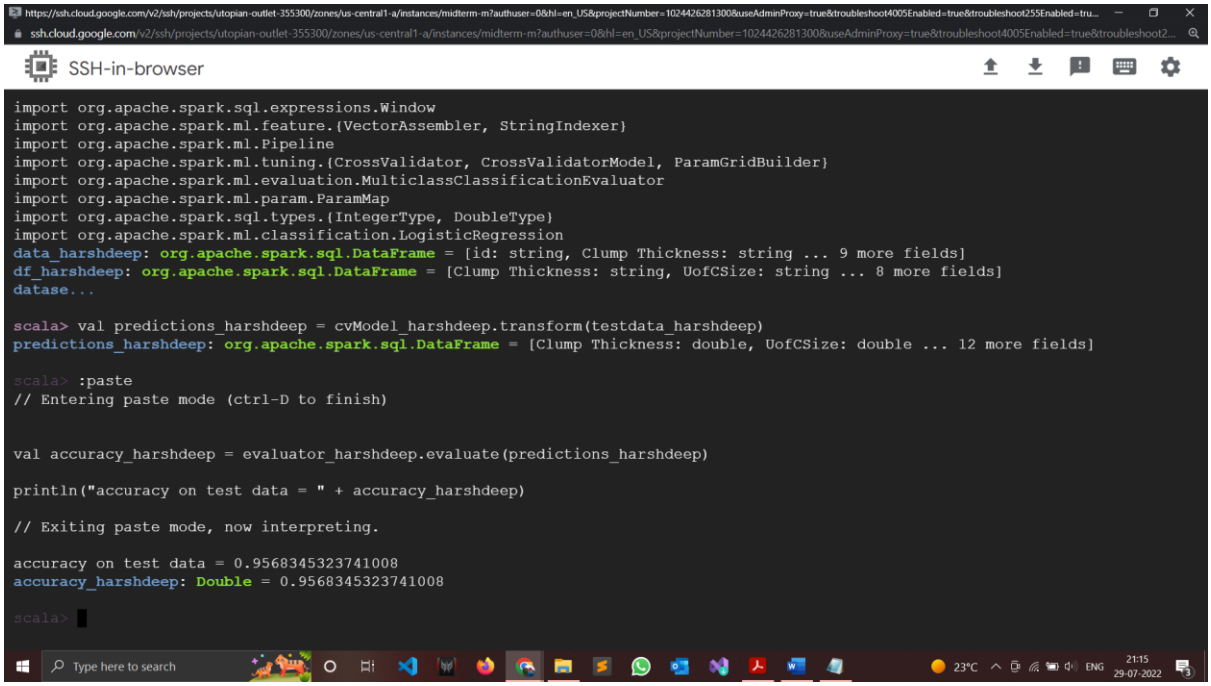
Training our model



Prediction using testdata



Evaluating the performance of our model

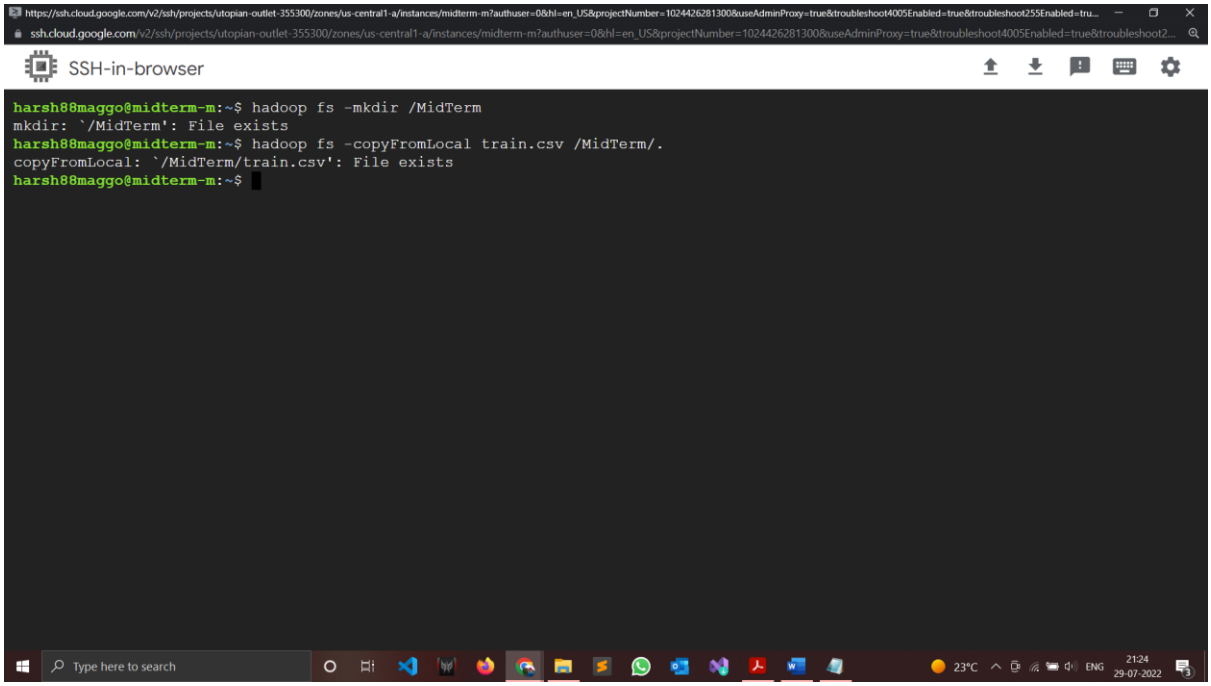


ACCURACY: 95.68%

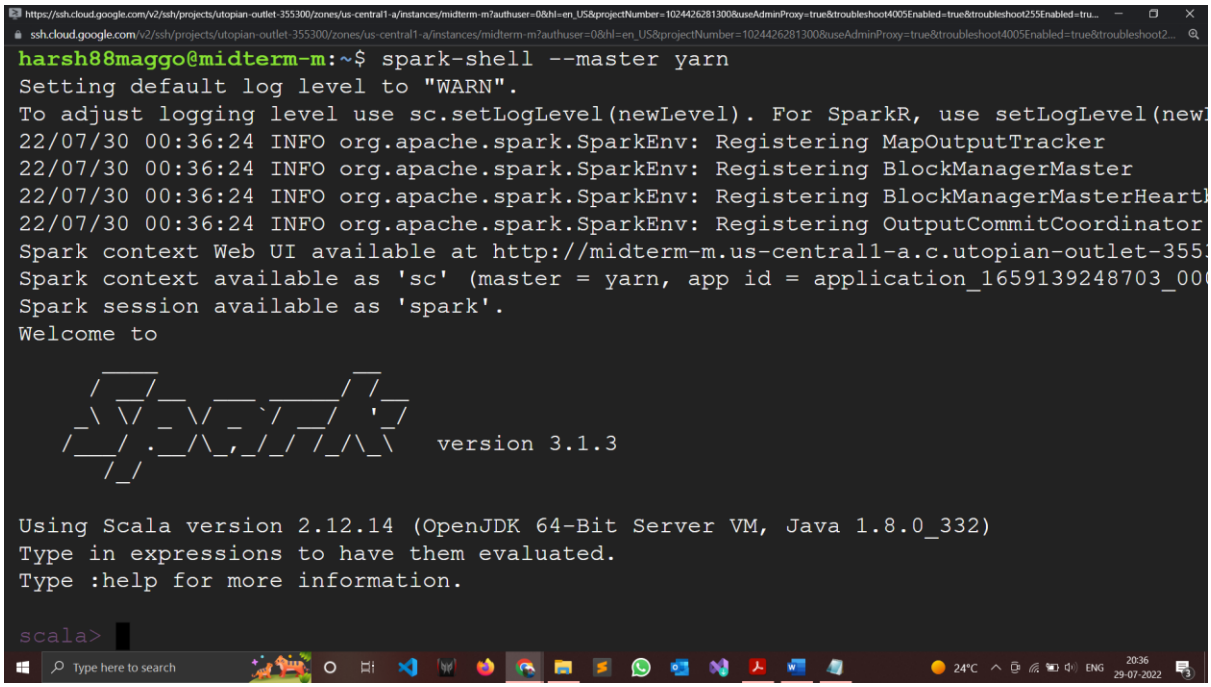
Problem 3: House Prices - Advanced Regression Techniques (With Linear Regression)

Dataset Link: https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview/evaluation

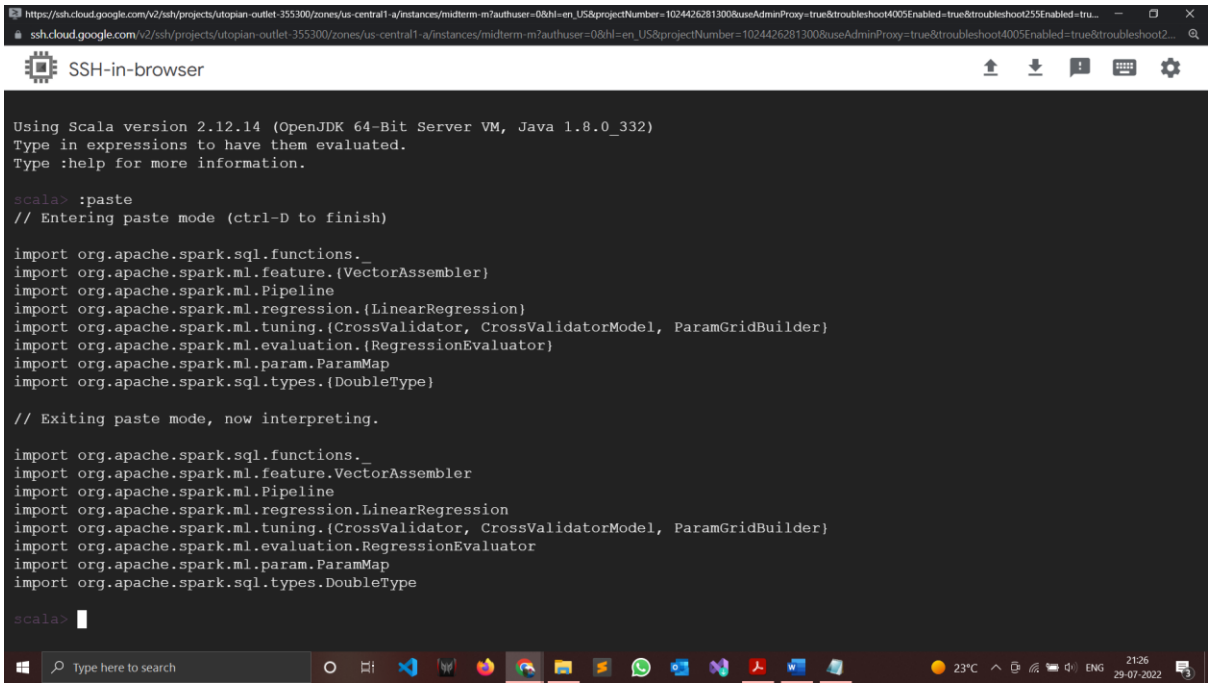
Creating a Hadoop directory and loading the data from the local files system to Hadoop



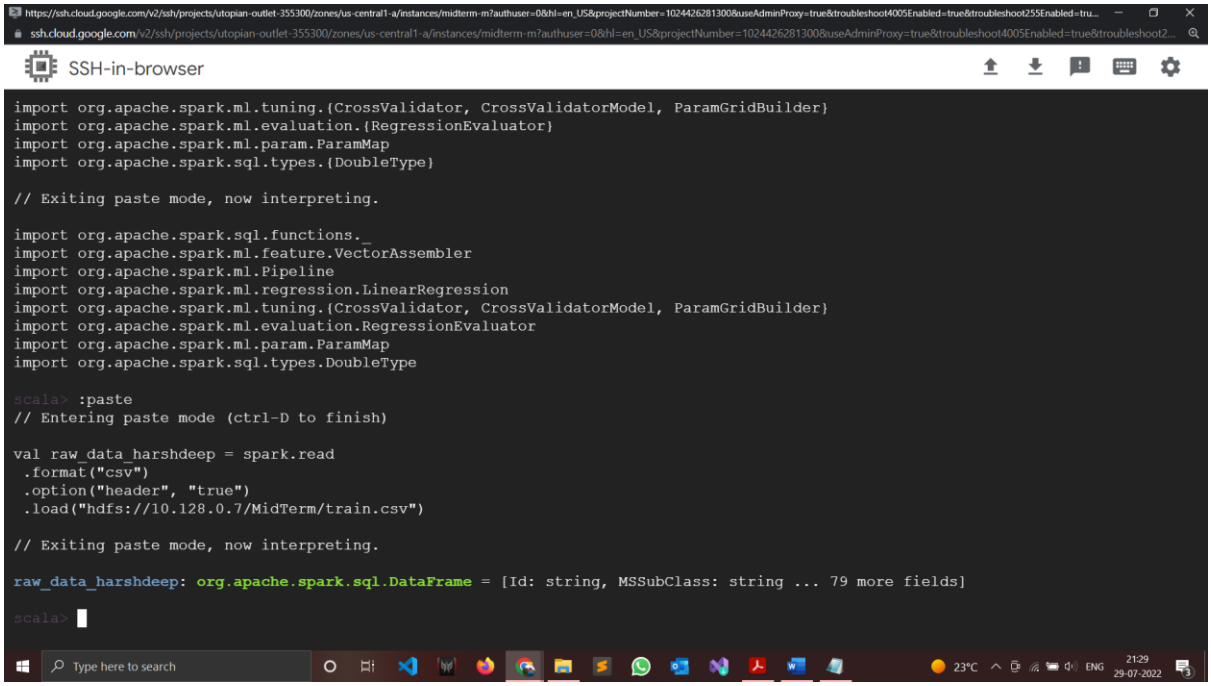
Running the Spark Shell



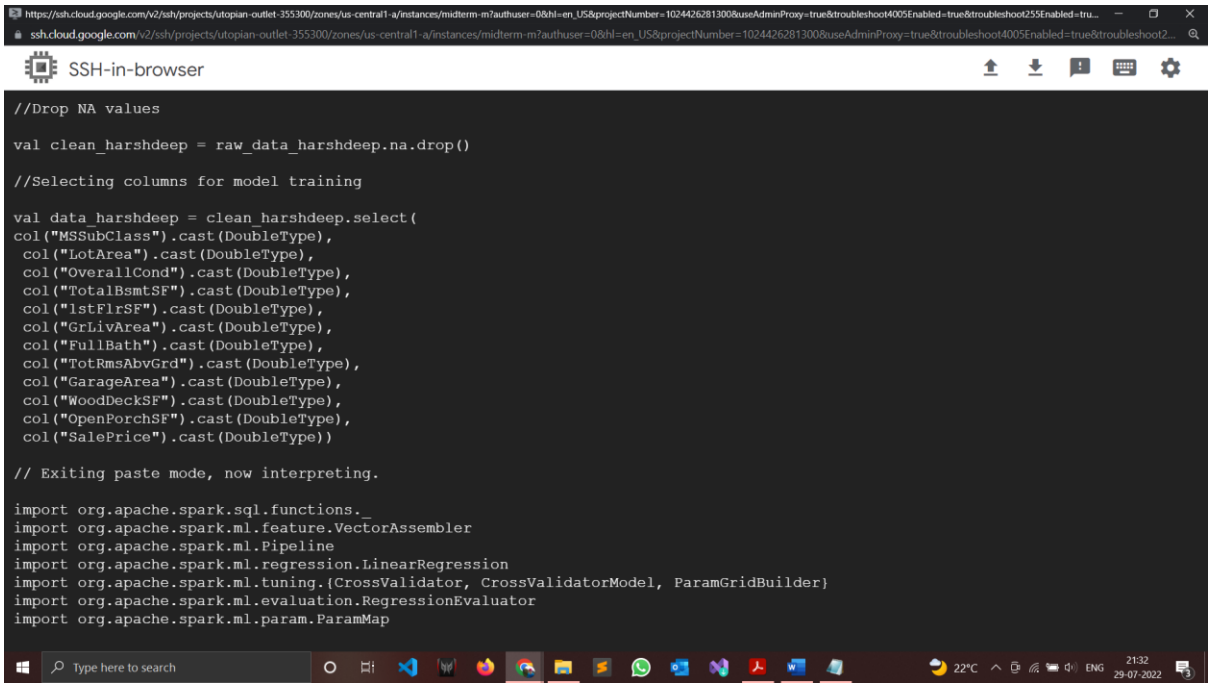
Import statements



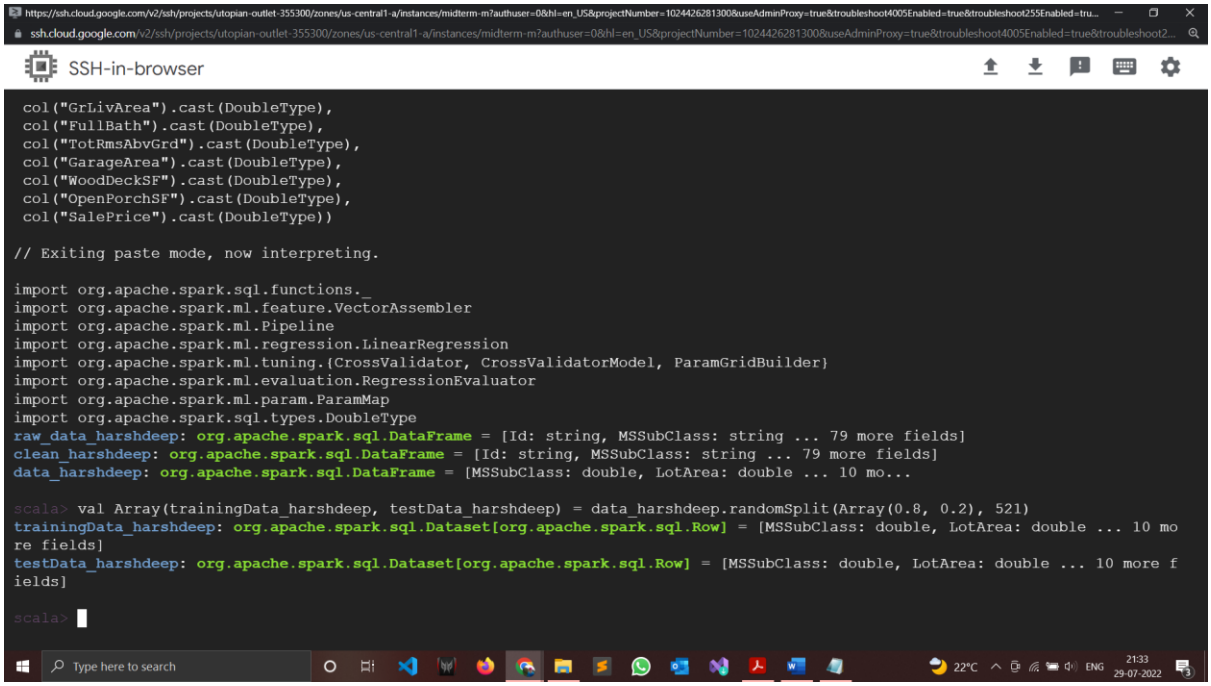
Reading the CSV file



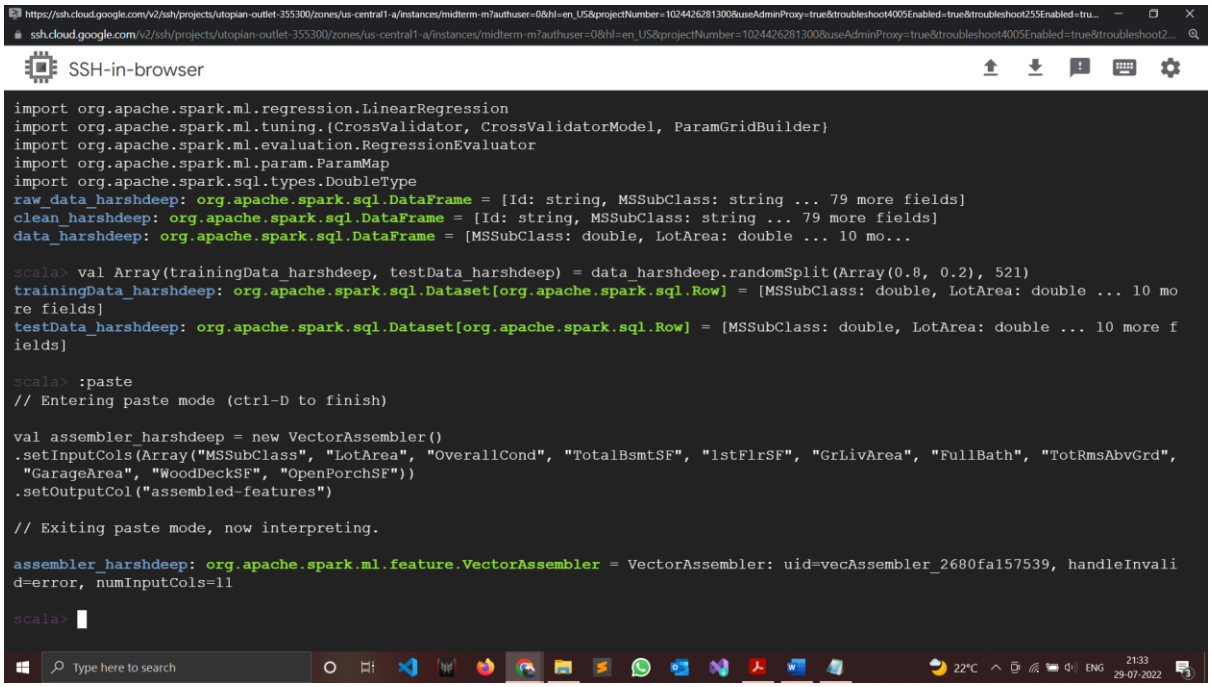
Dropping the NA values from the dataset, selecting columns for model training, and typecasting the data in type Double



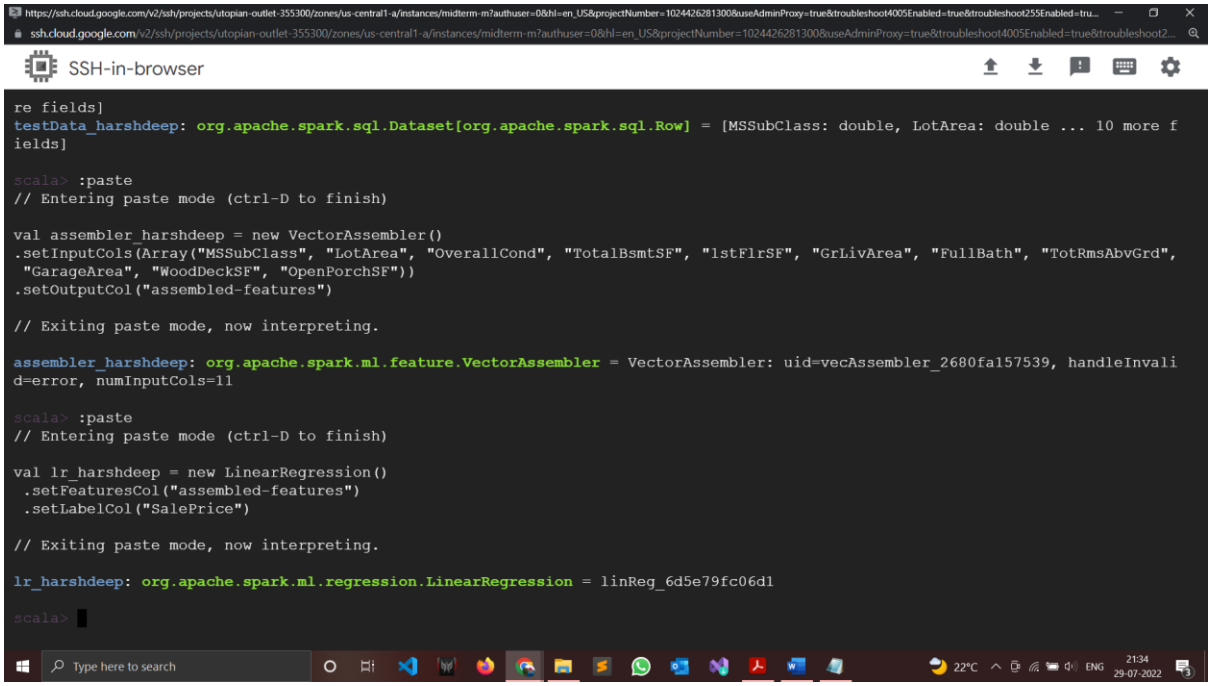
Split the dataset into train and test



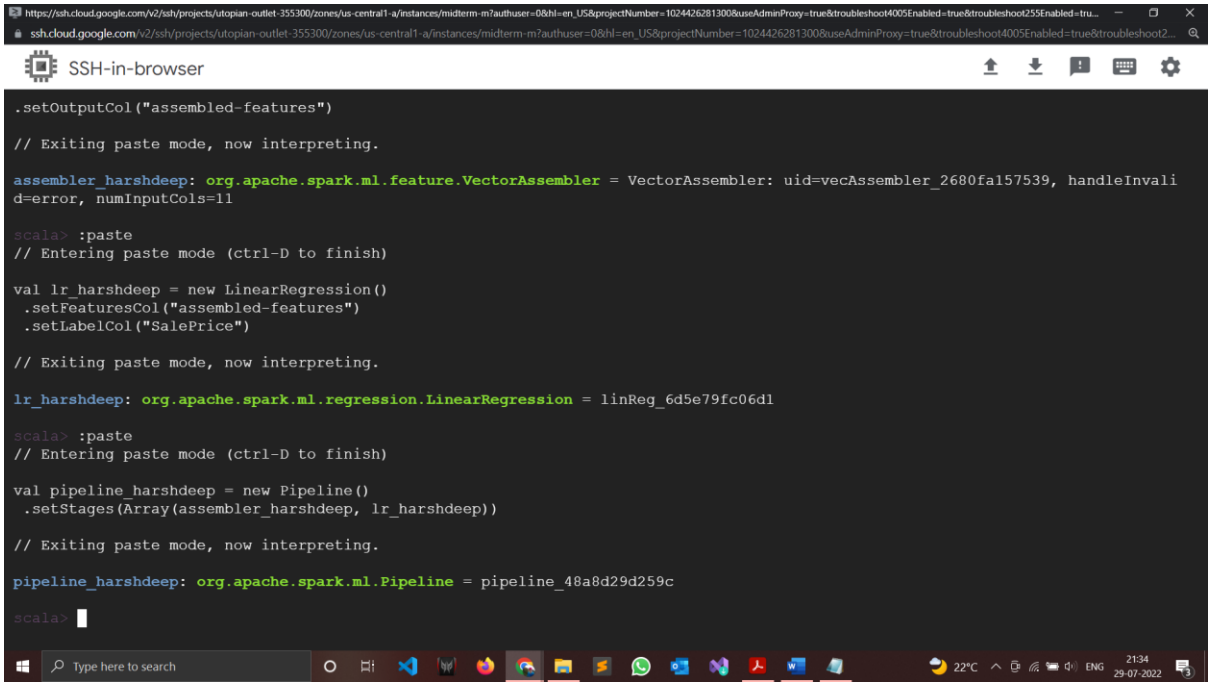
Assembling the features using VectorAssembler



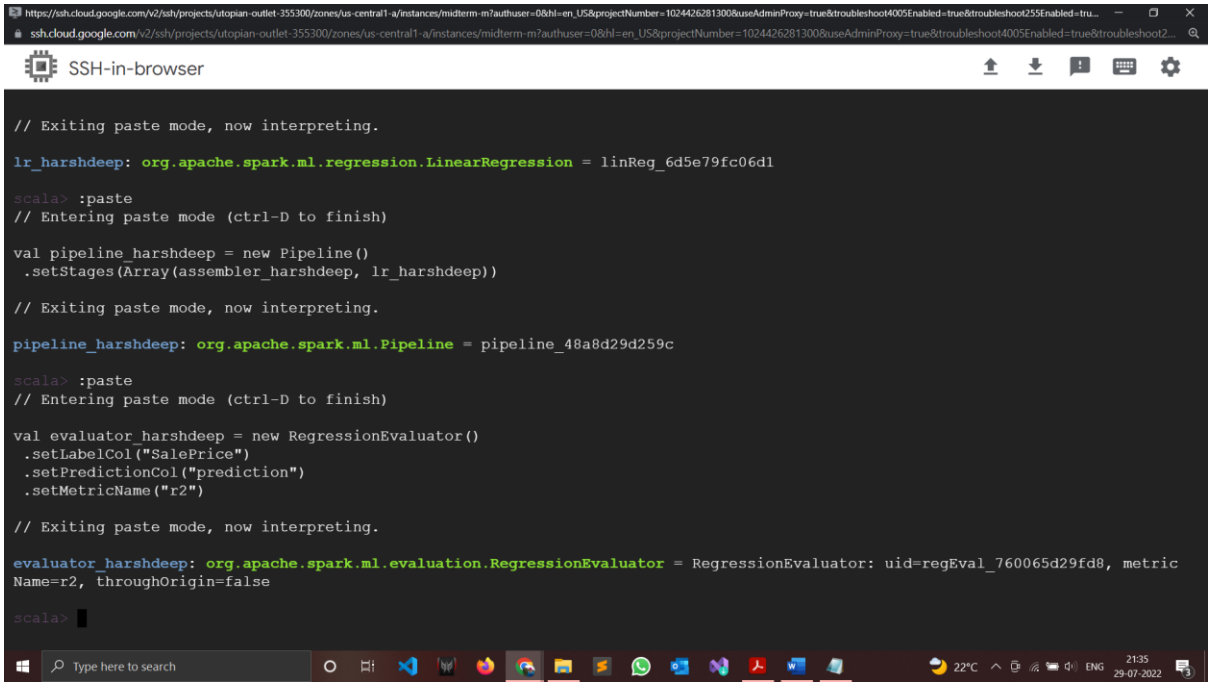
Creating the Linear Regression Object and passing the features



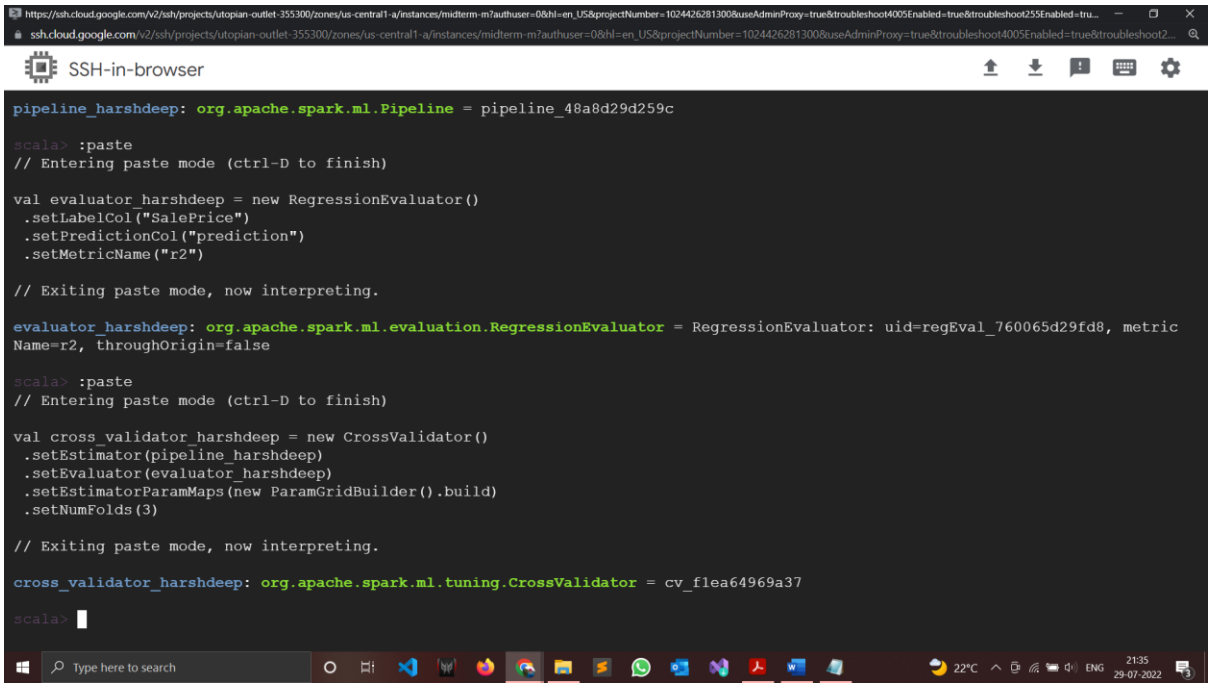
Creating the pipeling



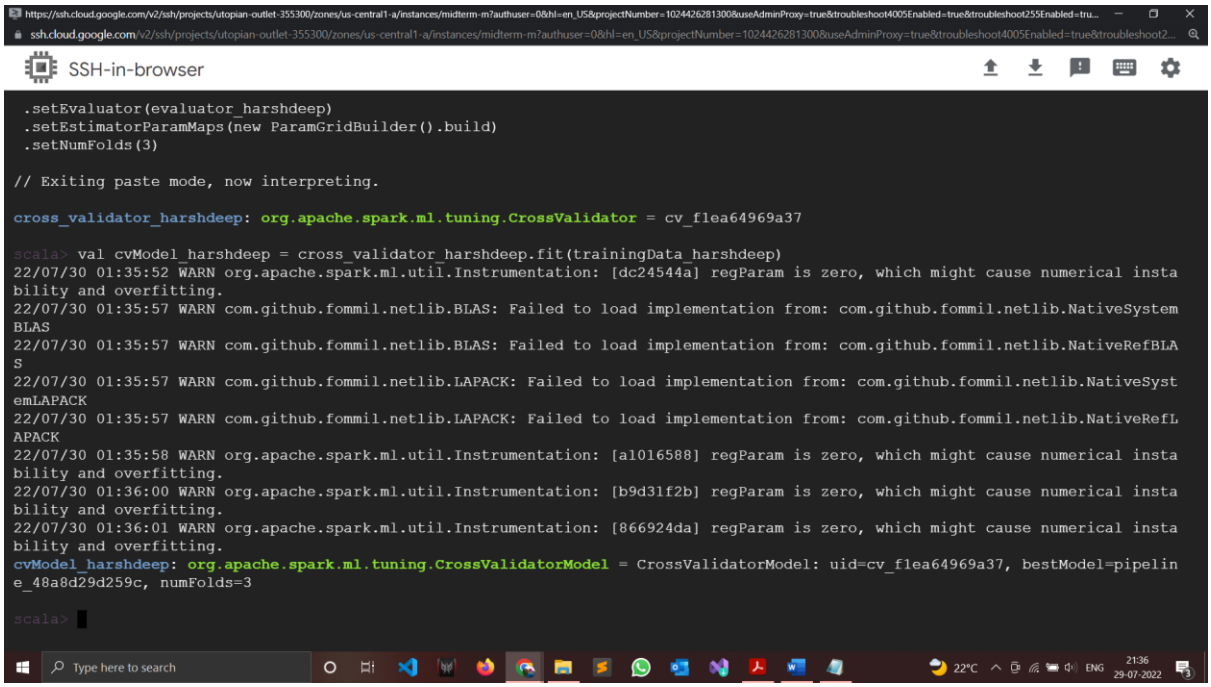
Evaluator for our model



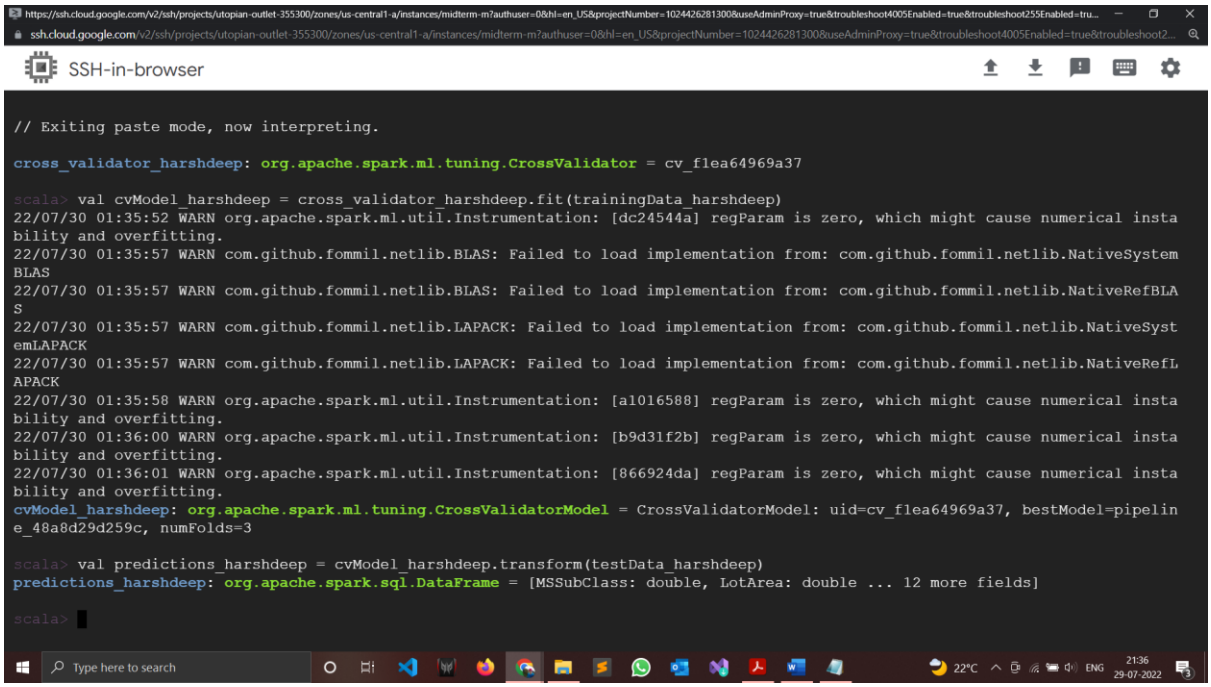
Creating the Cross Validator



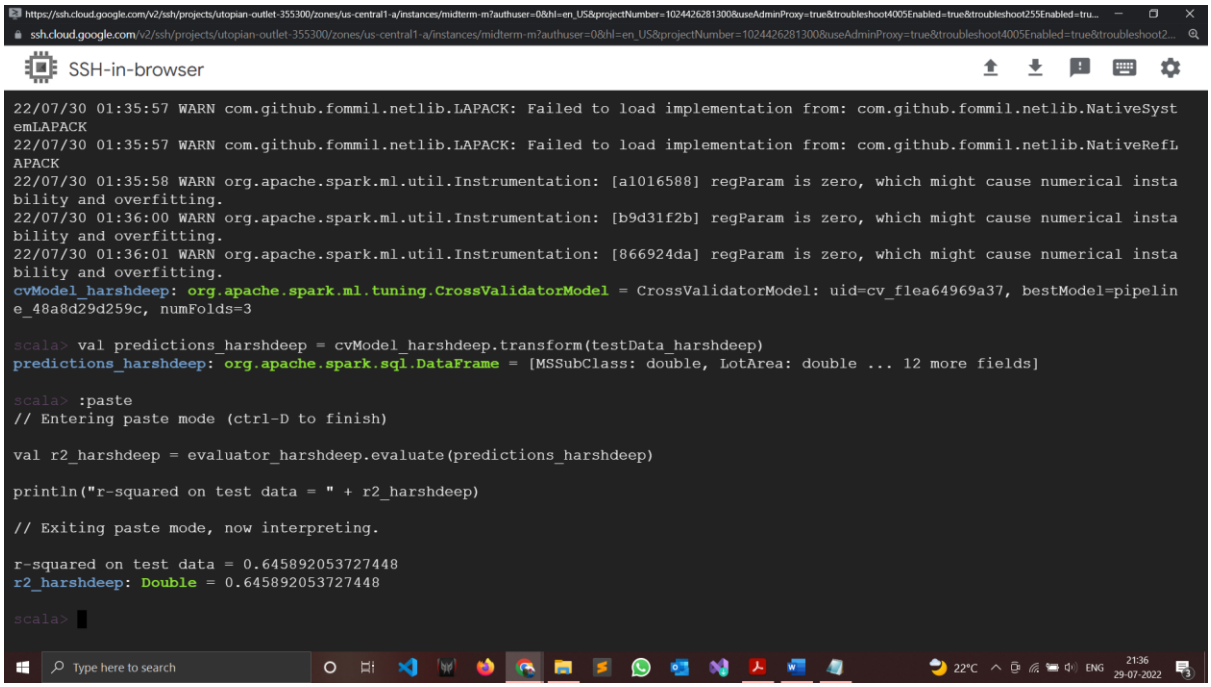
Training our model



Prediction using testdata



Evaluating the performance of our model



ACCURACY: 64.58%