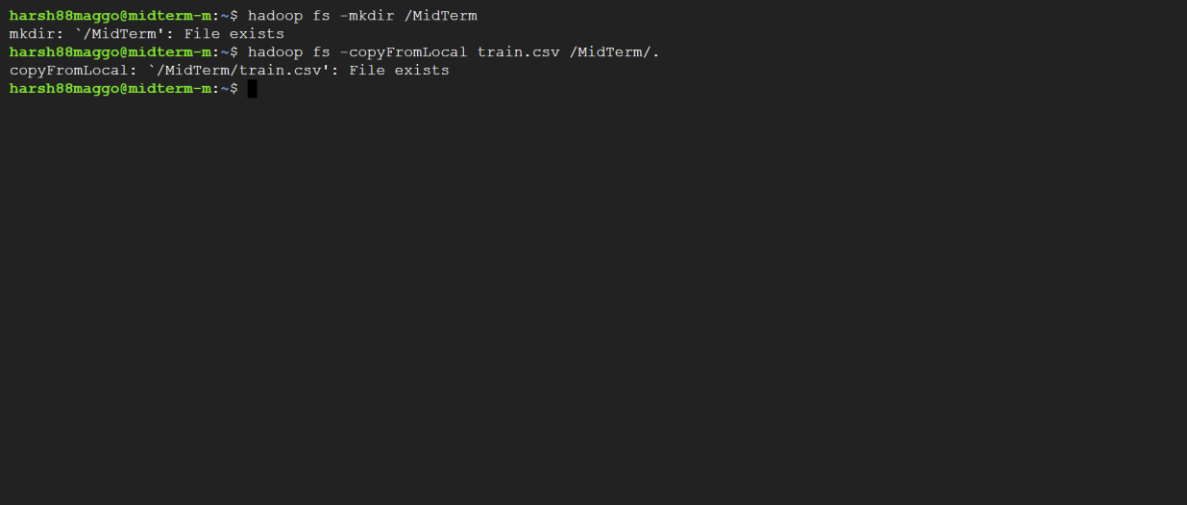


**Dataset Link:** <https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview/evaluation>



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...

SSH-in-browser

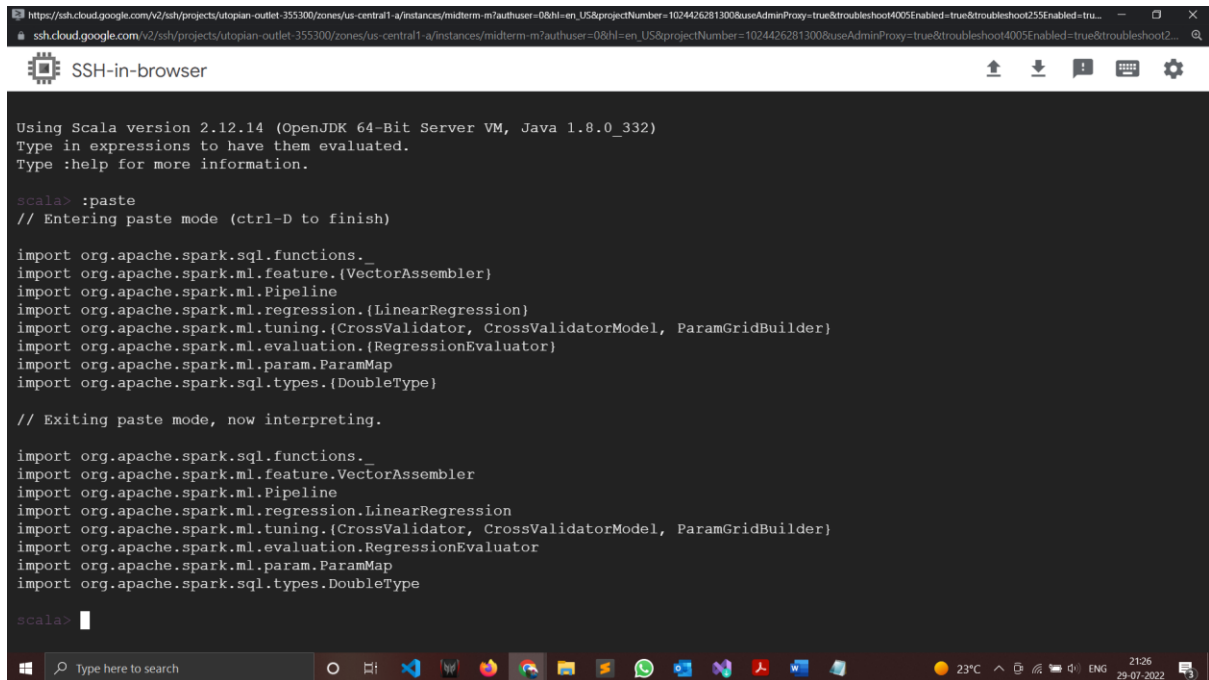
```
harsh88maggo@midterm-m:~$ hadoop fs -mkdir /MidTerm
mkdir: `/MidTerm': File exists
harsh88maggo@midterm-m:~$ hadoop fs -copyFromLocal train.csv /MidTerm/.
copyFromLocal: `/MidTerm/train.csv': File exists
harsh88maggo@midterm-m:~$
```

Type here to search

23°C 2124 29-07-2022

```
https://ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m7/authuser=08hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot4255Enabled=tru...  
■ ssh.cloud.google.com/v2/ssh/projects/utopian-outlet-355300/zones/us-central1-a/instances/midterm-m7/authuser=08hl=en_US&projectNumber=1024426281300&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot4255Enabled=tru...  
  
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(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  
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 the Scala REPL prompt and a series of import statements for Spark ML and SQL libraries. The user enters the command `:paste`, which enters paste mode. The pasted code includes imports for `org.apache.spark.sql.functions._`, `org.apache.spark.ml.feature.VectorAssembler`, `org.apache.spark.ml.Pipeline`, `org.apache.spark.ml.regression.LinearRegression`, `org.apache.spark.ml.tuning.{CrossValidator, CrossValidatorModel, ParamGridBuilder}`, `org.apache.spark.ml.evaluation.RegressionEvaluator`, `org.apache.spark.ml.param.ParamMap`, and `org.apache.spark.sql.types.DoubleType`. The terminal shows the code being interpreted line by line. The bottom of the window shows a Windows taskbar with various icons and a system tray displaying the date and time as 21:26 on 29-07-2022.

```
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)

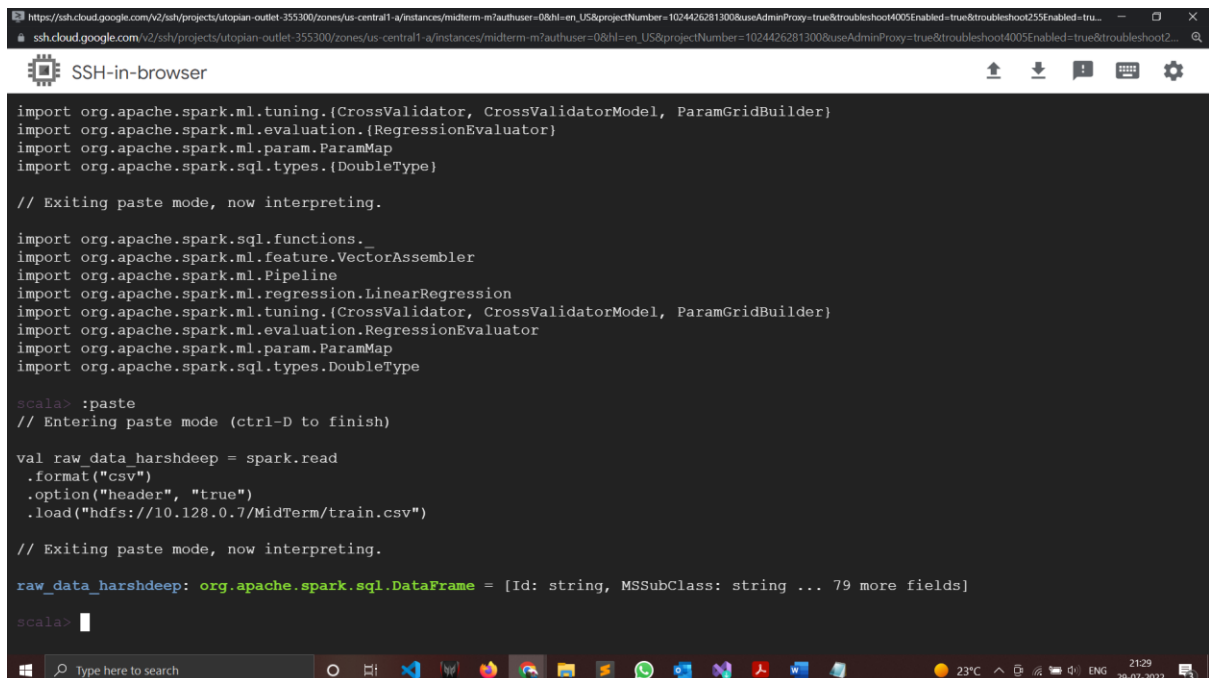
import org.apache.spark.sql.functions._
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.regression.LinearRegression
import org.apache.spark.ml.tuning.{CrossValidator, CrossValidatorModel, ParamGridBuilder}
import org.apache.spark.ml.evaluation.RegressionEvaluator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

## Reading the CSV file



The screenshot shows a terminal window titled "SSH-in-browser" with a dark background. The terminal displays the Scala REPL prompt and a series of import statements for Spark ML and SQL libraries. The user enters the command `:paste`, which enters paste mode. The pasted code includes imports for `org.apache.spark.ml.tuning.{CrossValidator, CrossValidatorModel, ParamGridBuilder}`, `org.apache.spark.ml.evaluation.RegressionEvaluator`, `org.apache.spark.ml.param.ParamMap`, and `org.apache.spark.sql.types.DoubleType`. The code then shows the user exiting paste mode and interpreting the code. The next part of the code shows the user entering `:paste` again, followed by the code to read a CSV file from HDFS into a DataFrame. The code includes `val raw_data_harshdeep = spark.read.format("csv").option("header", "true").load("hdfs://10.128.0.7/MidTerm/train.csv")`. The terminal shows the code being interpreted line by line. The bottom of the window shows a Windows taskbar with various icons and a system tray displaying the date and time as 21:29 on 29-07-2022.

```
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)

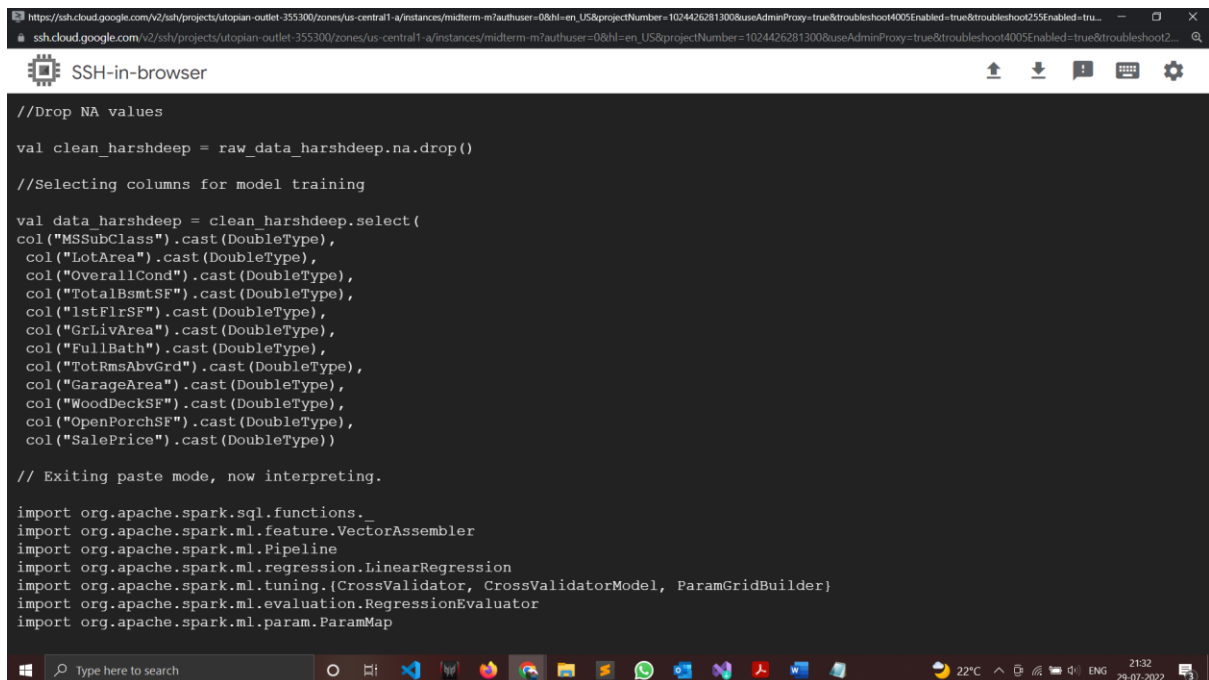
val raw_data_harshdeep = spark.read
  .format("csv")
  .option("header", "true")
  .load("hdfs://10.128.0.7/MidTerm/train.csv")

// Exiting paste mode, now interpreting.

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

scala>
```

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()

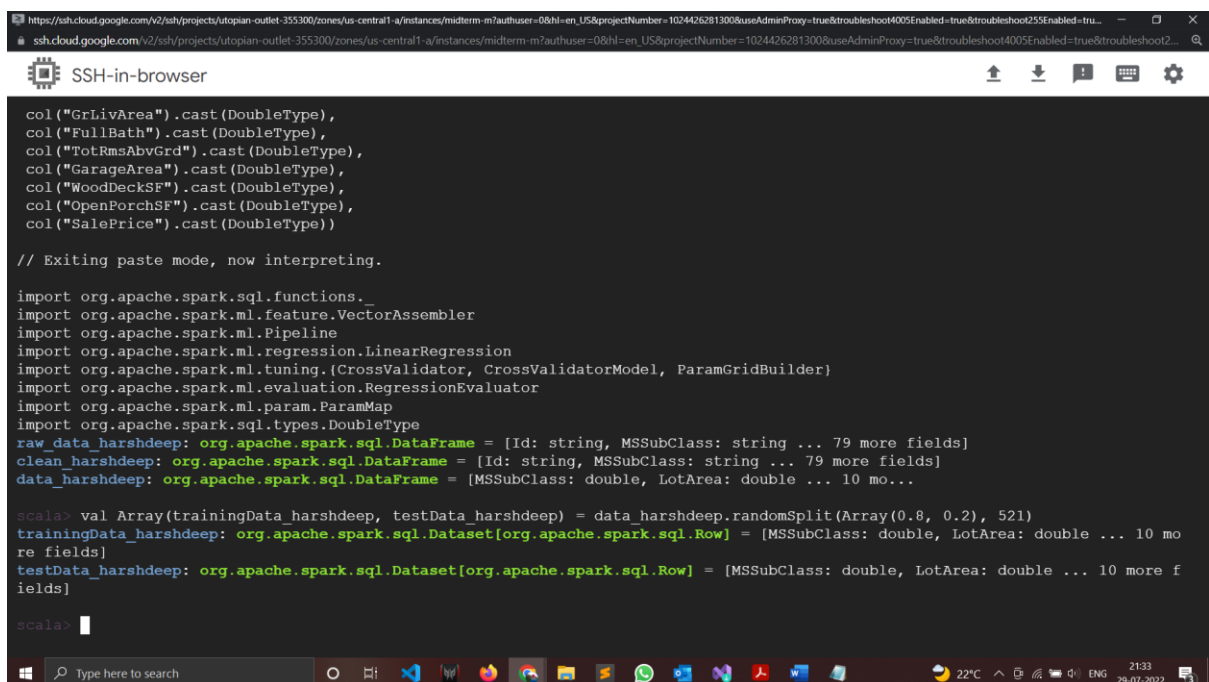
//Selecting columns for model training

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.RegresionEvaluator
import org.apache.spark.ml.param.ParamMap
```

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))

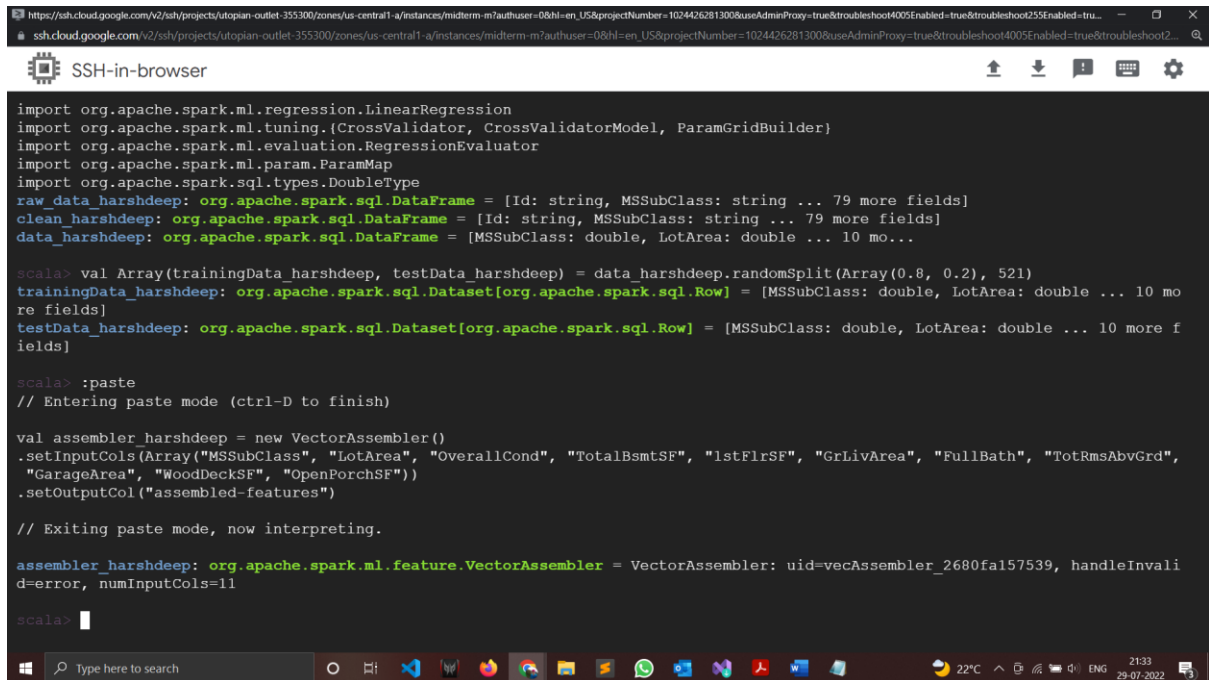
// Exiting paste mode, now interpreting.

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.RegresionEvaluator
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 more fields]
testData_harshdeep: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [MSSubClass: double, LotArea: double ... 10 more fields]

scala> 
```

## 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 more 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 more fields]
testData_harshdeep: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [MSSubClass: double, LotArea: double ... 10 more fields]

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

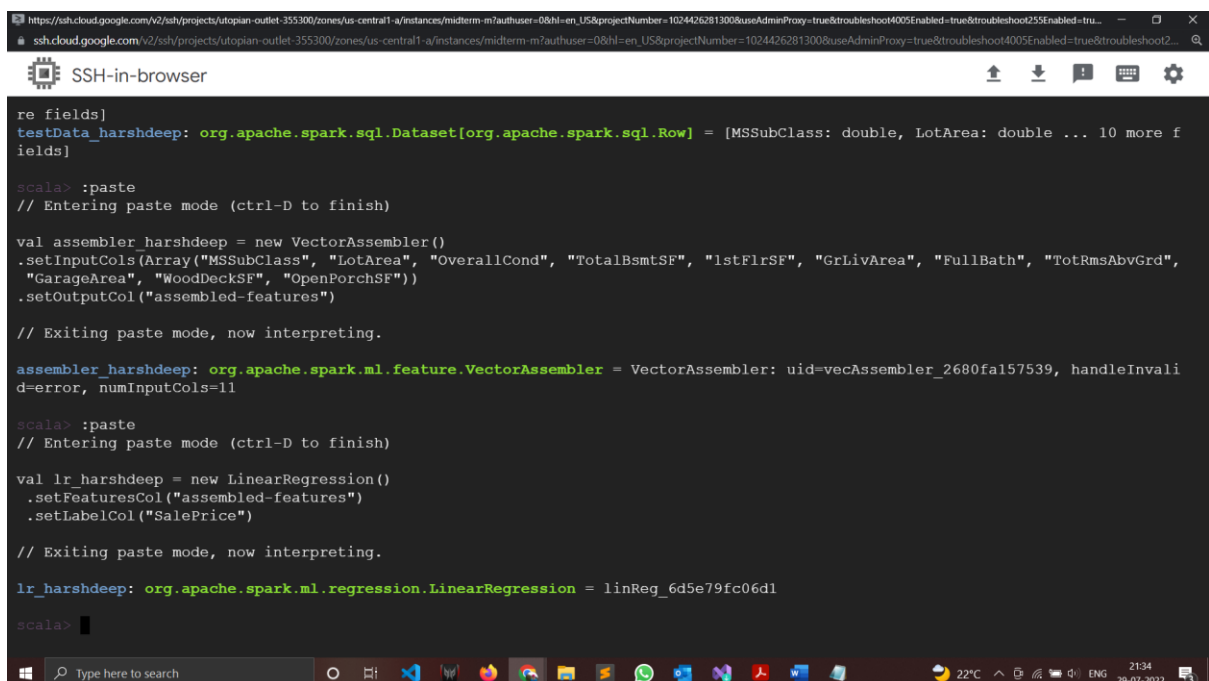
val assembler_harshdeep = new VectorAssembler()
.setInputCols(Array("MSSubClass", "LotArea", "OverallCond", "TotalBsmstSF", "1stFlrSF", "GrLivArea", "FullBath", "TotRmsAbvGrd",
"GarageArea", "WoodDeckSF", "OpenPorchSF"))
.setOutputCol("assembled-features")

// Exiting paste mode, now interpreting.

assembler_harshdeep: org.apache.spark.ml.feature.VectorAssembler = VectorAssembler: uid=vecAssembler_2680fa157539, handleInvalid=error, 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 fields]

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

val assembler_harshdeep = new VectorAssembler()
.setInputCols(Array("MSSubClass", "LotArea", "OverallCond", "TotalBsmstSF", "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, handleInvalid=error, numInputCols=11

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

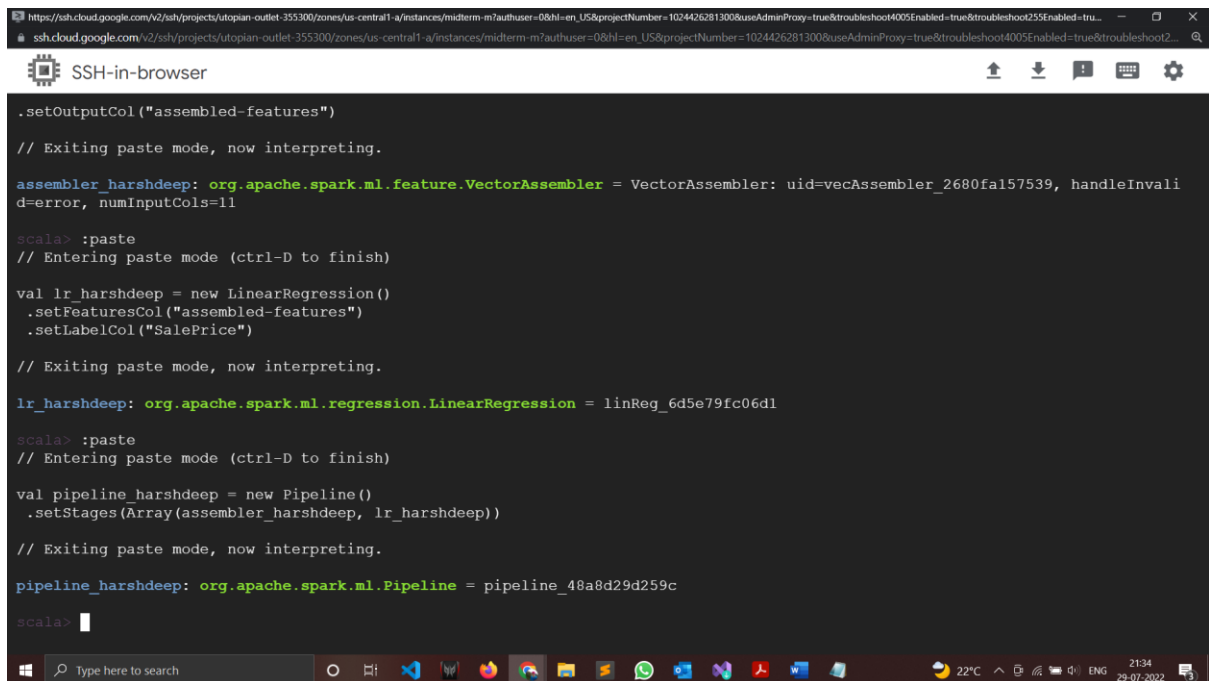
val lr_harshdeep = new LinearRegression()
.setFeaturesCol("assembled-features")
.setLabelCol("SalePrice")

// Exiting paste mode, now interpreting.

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

scala>
```

## Creating the pipeling



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

// Exiting paste mode, now interpreting.

assembler_harshdeep: org.apache.spark.ml.feature.VectorAssembler = VectorAssembler: uid=vecAssembler_2680fa157539, handleInvalid=error, 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)

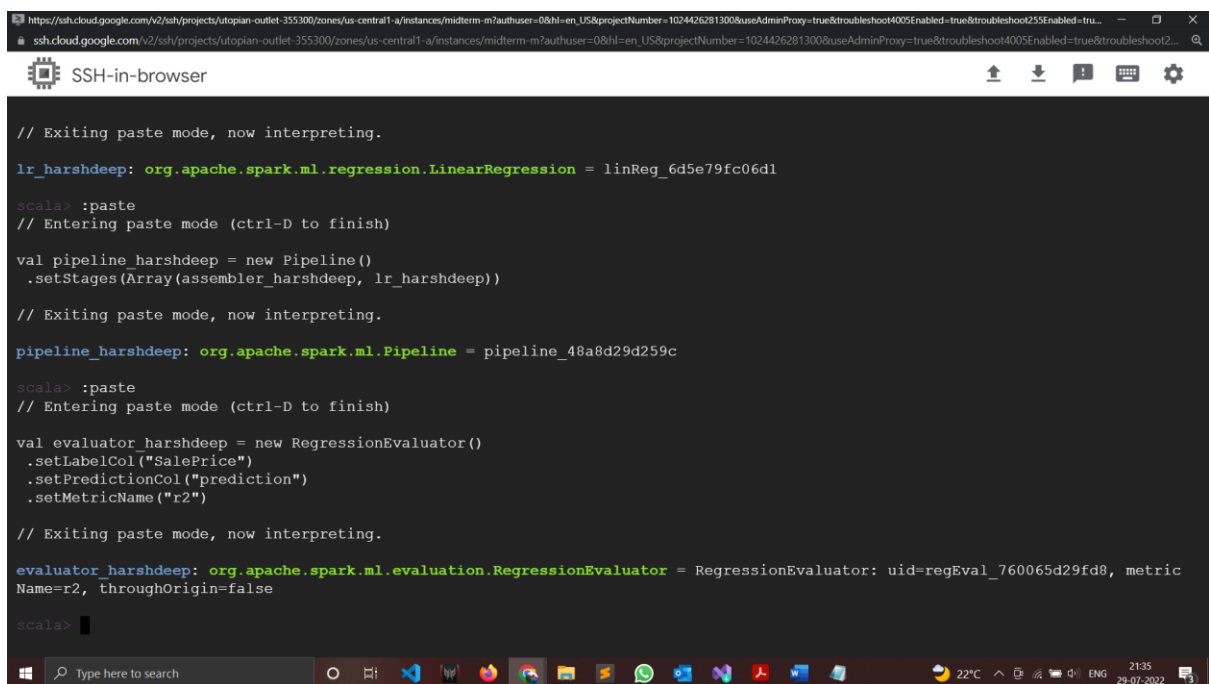
val pipeline_harshdeep = new Pipeline()
  .setStages(Array(assembler_harshdeep, lr_harshdeep))

// Exiting paste mode, now interpreting.

pipeline_harshdeep: org.apache.spark.ml.Pipeline = pipeline_48a8d29d259c

scala> 
```

## Evaluator for our model



```
// 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-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&troubleshoot2...

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-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&troubleshoot2...

SSH-in-browser

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

// Exiting paste mode, now interpreting.

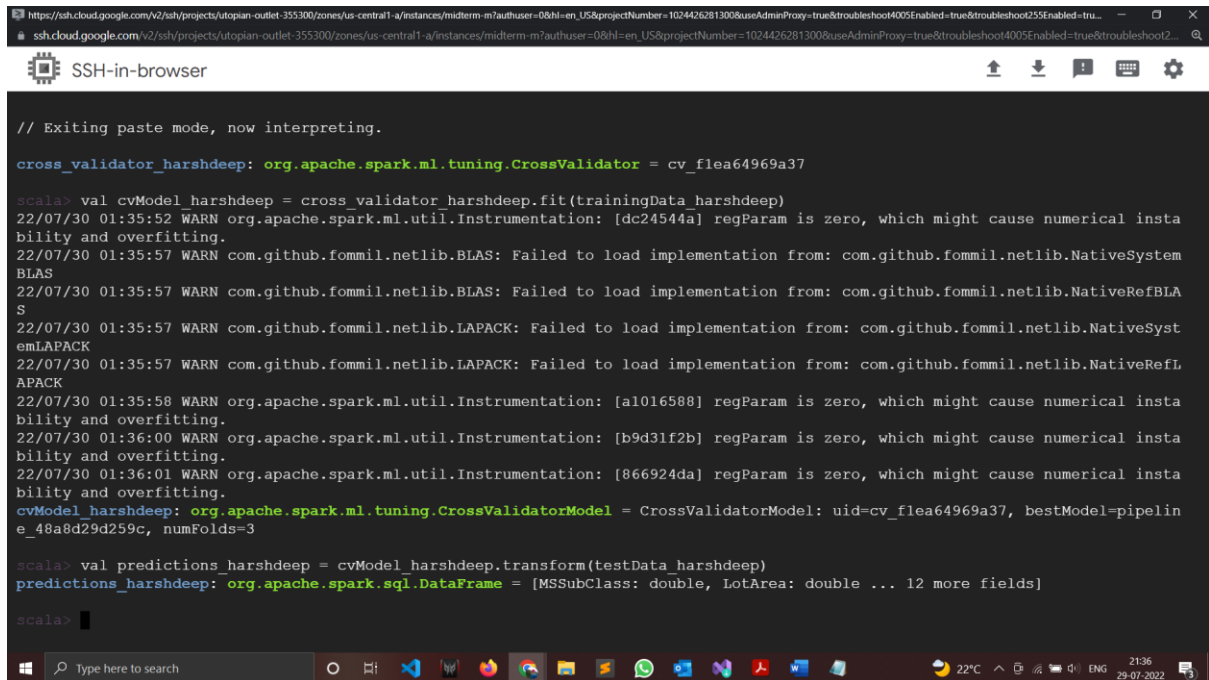
cross_validator_harshdeep: org.apache.spark.ml.tuning.CrossValidator = cv_flea64969a37

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.

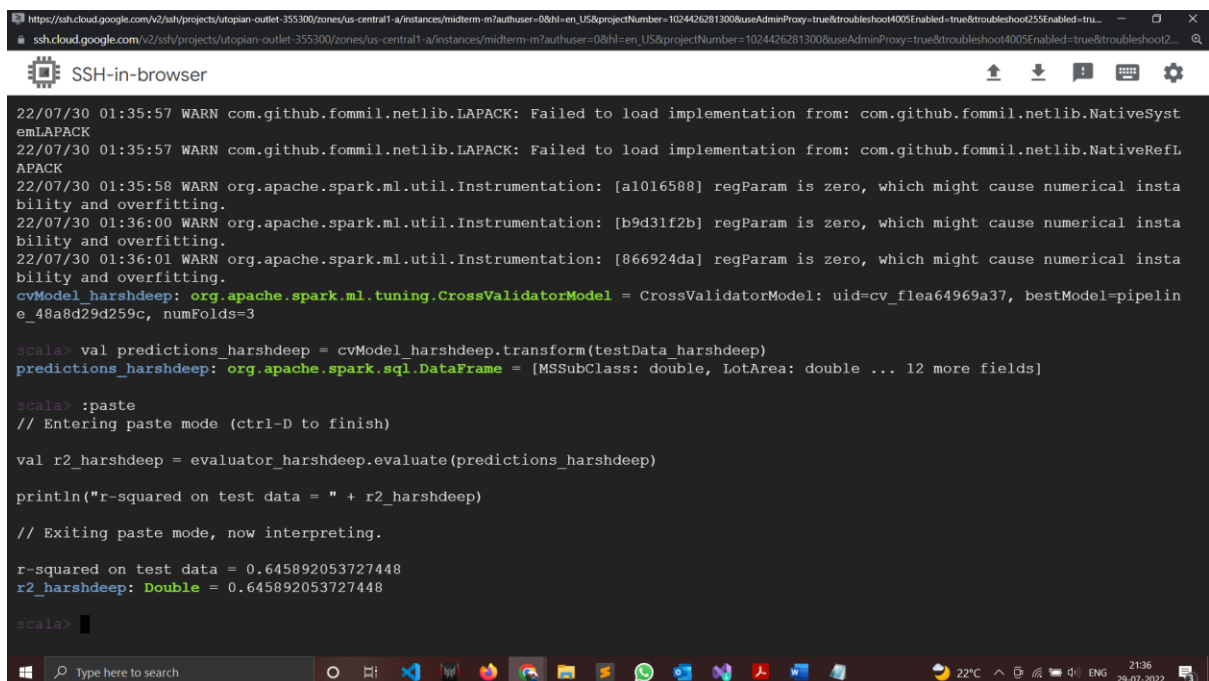
cross_validator_harshdeep: org.apache.spark.ml.tuning.CrossValidator = cv_flea64969a37

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%**