

Lab4 Report:

Task: Create your own dataset for Image Classification Problem. Use the workflow as discussed in the Tutorial 4 Session using Decision Tree Algorithm. Report the accuracy and confusion matrix obtained.

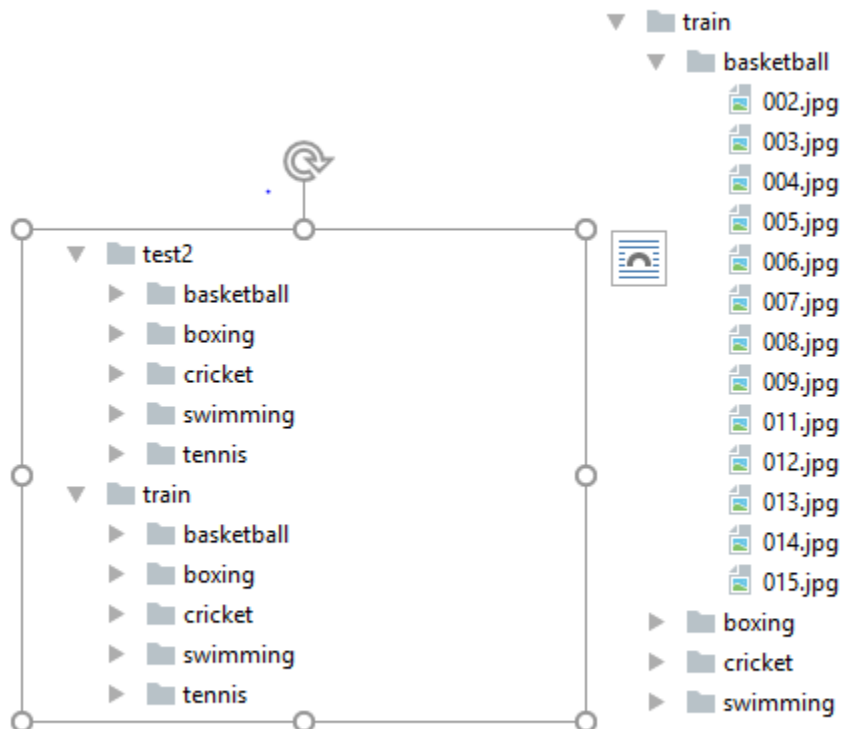
Solution:

In this Tutorial, I implemented the Image Classification for images in sports Category. I considered 5 different sports images as 5 classes and generated model using Decision tree as well as Random Forest. The 5 Classes are:

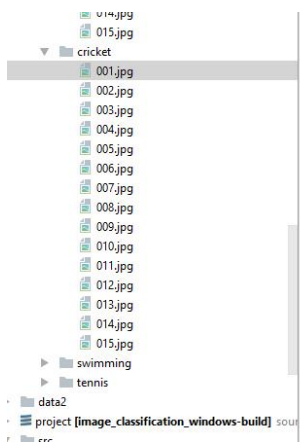
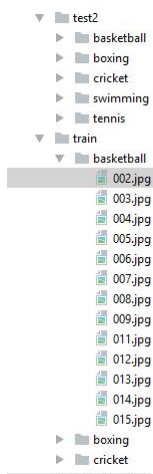
1. Basket Ball
2. Boxing
3. Cricket
4. Swimming
5. Tennis

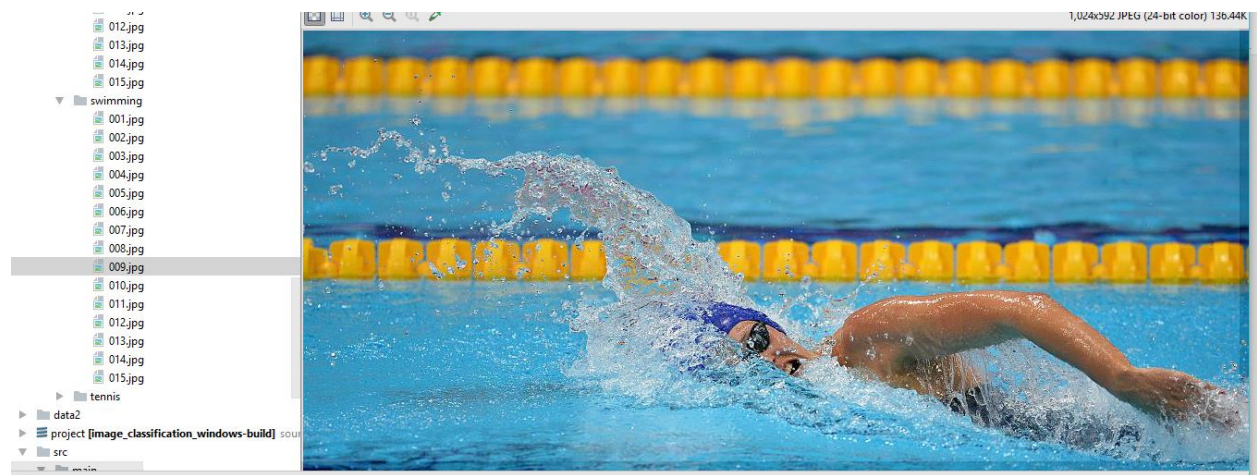
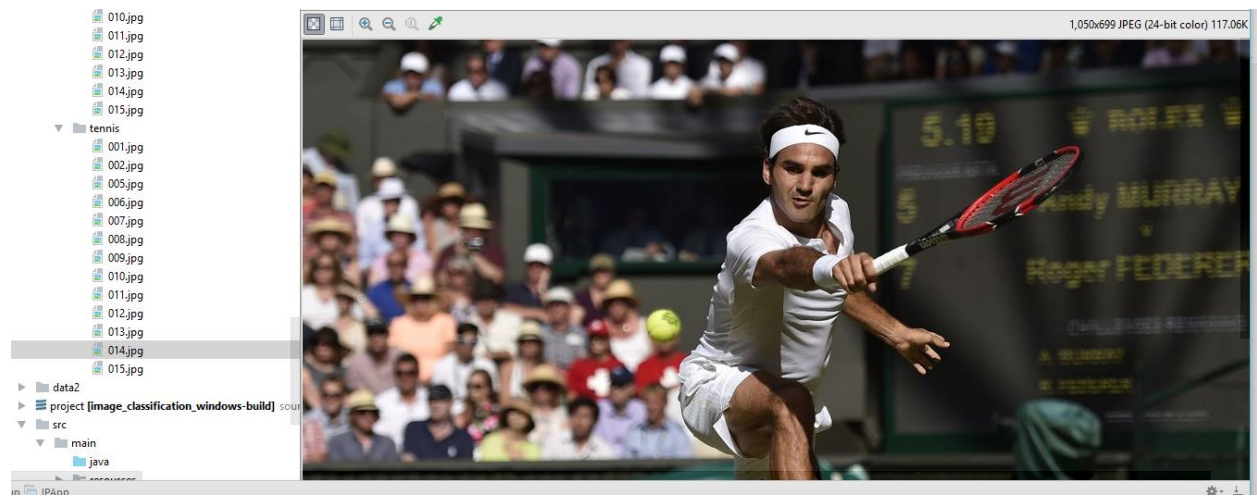
Training and Testing the Model:

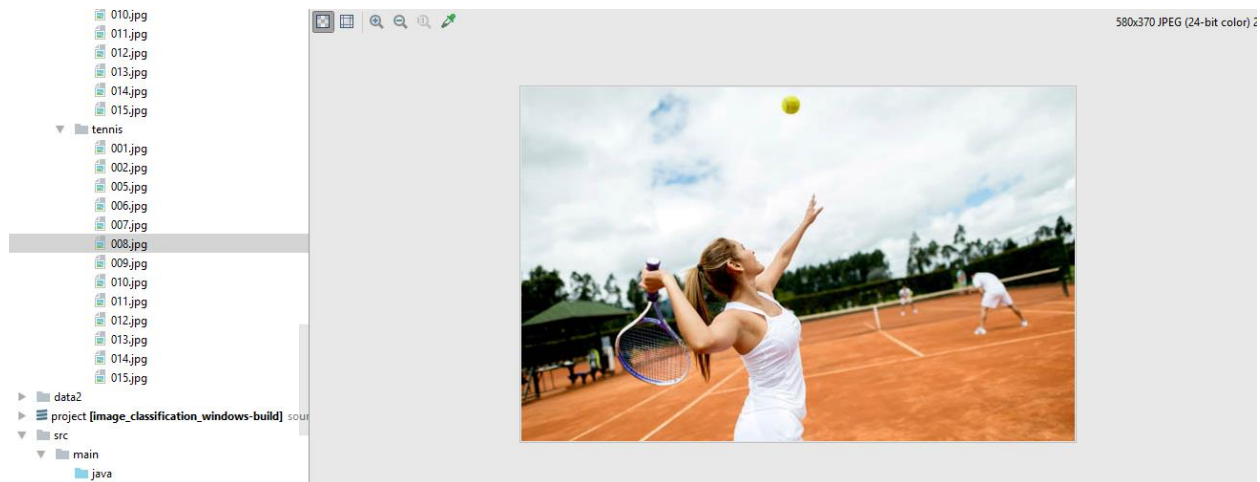
Took 15 images in each category for training the model and 5 images in each category for testing the model.



Sample Screen Shots of the training Data:







Screen shot of the generated model using decision Tree:

```

106 featuresSeq.saveAsTextFile(IPSettings.HISTOGRAM_PATH)
107 println("Total size : " + data.size)
108 }
109
110
111 def generateDecisionTreeModel(sc: SparkContext): Unit = {
112   if (Files.exists(Paths.get(IPSettings.DECISION_TREE_PATH))) {
113     println(s"${IPSettings.DECISION_TREE_PATH} exists, skipping Decision Tree model formation..")
114     return
115   }
116
117   val data = sc.textFile(IPSettings.HISTOGRAM_PATH)
118   val parsedData = data.map { line =>
119     val parts = line.split(',')
120     LabeledPoint(parts(0).toDouble, Vectors.dense(parts(1).split(' ').map(_.toDouble)))
121   }
122
123   // Split data into training (70%) and test (30%).
124   val splits = parsedData.randomSplit(Array(0.7, 0.3), seed = 11L)
125   val training = parsedData
126   val test = splits(1)
  
```

```

Run IPApp
17/02/17 06:44:16 INFO FileOutputCommitter: Saved output of task 'attempt_201702170644_4717_m_000001_0' to file:/D:/Tutorial Code/image_classification
17/02/17 06:44:16 INFO FileOutputCommitter: Saved output of task 'attempt_201702170644_4717_m_000002_0' to file:/D:/Tutorial Code/image_classification
17/02/17 06:44:16 INFO FileOutputCommitter: Saved output of task 'attempt_201702170644_4717_m_000003_0' to file:/D:/Tutorial Code/image_classification
17/02/17 06:44:16 INFO ParquetFileReader: Initiating action with parallelism: 5
Decision Tree Model generated
17/02/17 06:44:16 INFO FileInputFormat: Total input paths to process : 25
17/02/17 06:44:16 INFO FileInputFormat: Total input paths to process : 25
17/02/17 06:44:16 INFO CombineFileInputFormat: DEBUG: Terminated node allocation with : CompletedNodes: 1, size left: 2386433
file:/D:/Tutorial Code/image_classification/windows/data/test2/basketball
  
```

link for Source code:

<https://github.com/gt784/BigDataAnalyticProject/tree/master/Source/Lab4/ImageClassification%20Using%20Decision%20Tree>

Results:(Accuracy and Confusion Matrix)

Decision Tree: The model generated using decision tree has obtained 84% Accuracy. Below is the screen sot for accuracy and confusion Matrix

```
17/02/21 09:00:04 INFO tensorflow.keras.callbacks: Saving data at memory at 0 MB, 200 counts
Predicting test image : tennis as tennis
(4.0,4)
(4.0,4)
(2.0,4)
(2.0,4)
(4.0,4)
(3.0,3)
(2.0,3)
(3.0,3)
(3.0,3)
(3.0,3)
(2.0,2)
(2.0,2)
(2.0,2)
(2.0,2)
(1.0,1)
(1.0,1)
(1.0,1)
(1.0,1)
(1.0,1)
(0.0,0)
(0.0,0)
(2.0,0)
(0.0,0)
(0.0,0)

0.84

|===== Confusion matrix =====|
4.0 0.0 1.0 0.0 0.0
0.0 5.0 0.0 0.0 0.0
0.0 0.0 5.0 0.0 0.0
0.0 0.0 1.0 4.0 0.0
0.0 0.0 2.0 0.0 3.0
b.84
17/02/21 06:46:03 INFO RemoteBtorRefProvider$RemoteTerminator: Shutting down remote daemon
```

Random Forest

The model generated using random forest has obtained 72% accuracy. Below is the screen shot for confusion matrix and accuracy.

```
17/02/21 09:00:04 INFO tensorflow.keras.callbacks: Saving data at memory at 0 MB, 200 counts
Predicting test image : tennis as tennis
(4.0,4)
(2.0,4)
(4.0,4)
(4.0,4)
(0.0,4)
(2.0,3)
(3.0,3)
(3.0,3)
(3.0,3)
(3.0,3)
(2.0,2)
(2.0,2)
(2.0,2)
(2.0,2)
(2.0,2)
(1.0,1)
(1.0,1)
(1.0,1)
(1.0,1)
(3.0,1)
(2.0,0)
(0.0,0)
(0.0,0)
(2.0,0)
(1.0,0)

[Stage 177:=====> (1 + 1) / 2]0.72
|===== Confusion matrix =====|
2.0 1.0 2.0 0.0 0.0
0.0 4.0 0.0 1.0 0.0
0.0 0.0 5.0 0.0 0.0
0.0 0.0 1.0 4.0 0.0
1.0 0.0 1.0 0.0 3.0
0.72
```