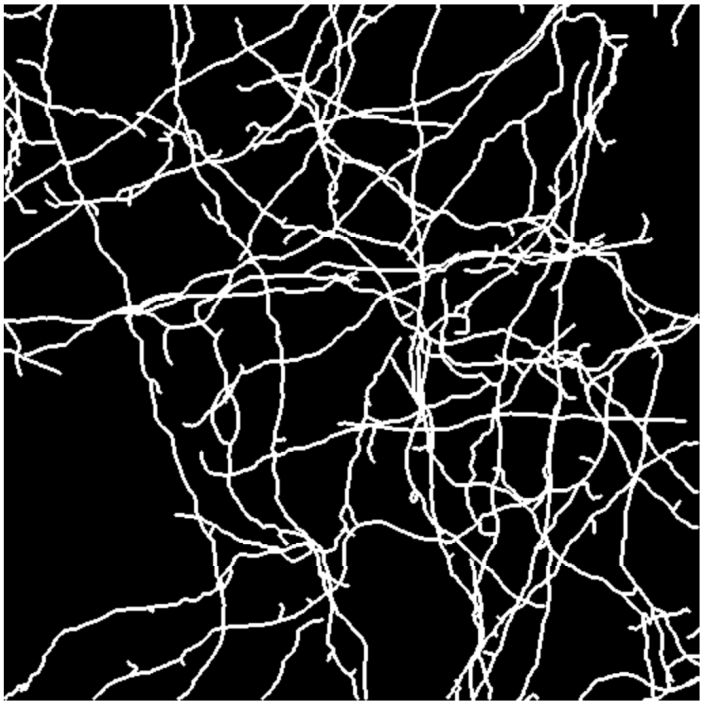
Project Report
Ajay Kauthule and Harsh Shukla
Parallel Data Processing with Map Reduce

Project to determine the background and Foreground pixels for a given image:



Type of model used and parameters explored.

Models Explored:
SVM, Logistic Regression and Random Forest.
The final evaluation that has been submitted has been done using random forest.

Parameters Considered for the development of this project:
Impurity :
Measure based on which optimal condition is measured, homogeneity is measured using this.
Maximum Depth :
The depth of the decision tree.Has a great impact on the performance overall ,if the depth is kept very high i.e. above 15 it gravely dismisses the performance.
Treecount:
Number of decision trees that must be formed.
Seed
Feature Subset Strategy:
Number of features that must be selected for the tree.

Preprocessing:

The data is very huge with almost 962240 records and 3088 columns hence it was difficult to analyze complete data at one go, hence we sliced the data into small parts to analyze that data. We broke records into small chunks and executed our code on it. The accuracy for this data set in this case was not considered.

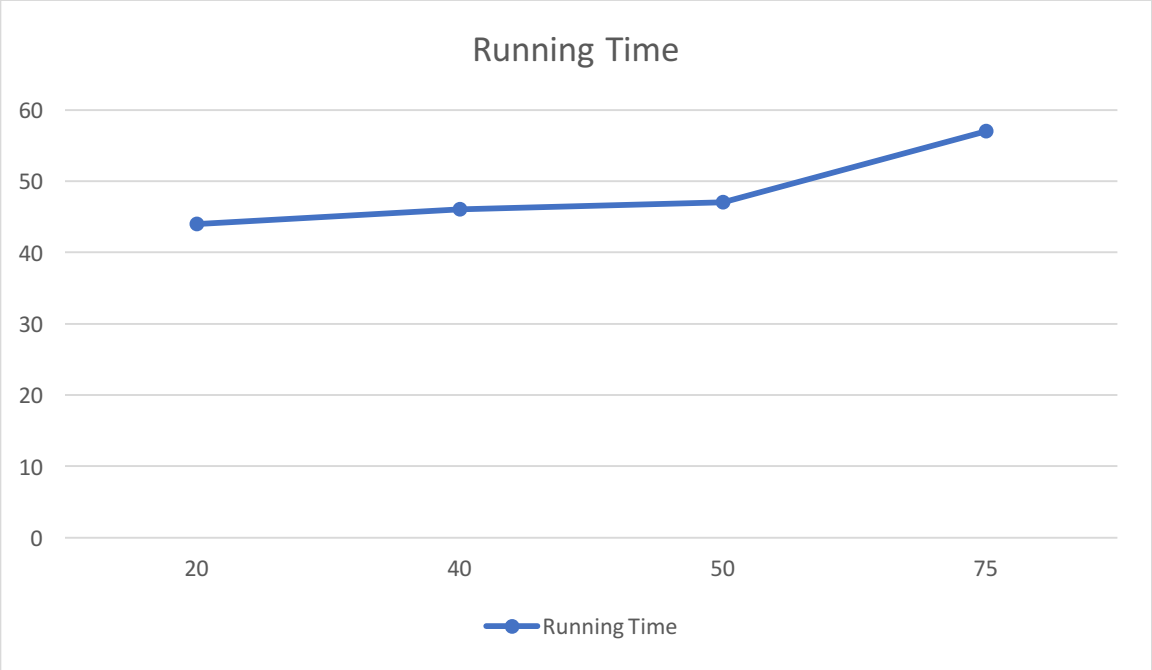
Accuracy:
The accuracy for all the parameter settings came out to be pretty consistent i.e. in the approx. 99%

Number of trees	Depth	Impurity	Accuracy
20	5	Gini	99.44%
50	10	Gini	99.44%
100	18	Gini	99.47%

Running Time and Speedup Results:
We will use the following graphs to represent our running times in comparison to different parameters.

- Running time v/s the depth of the Decision trees.
- Running time vs the number of trees in the model.

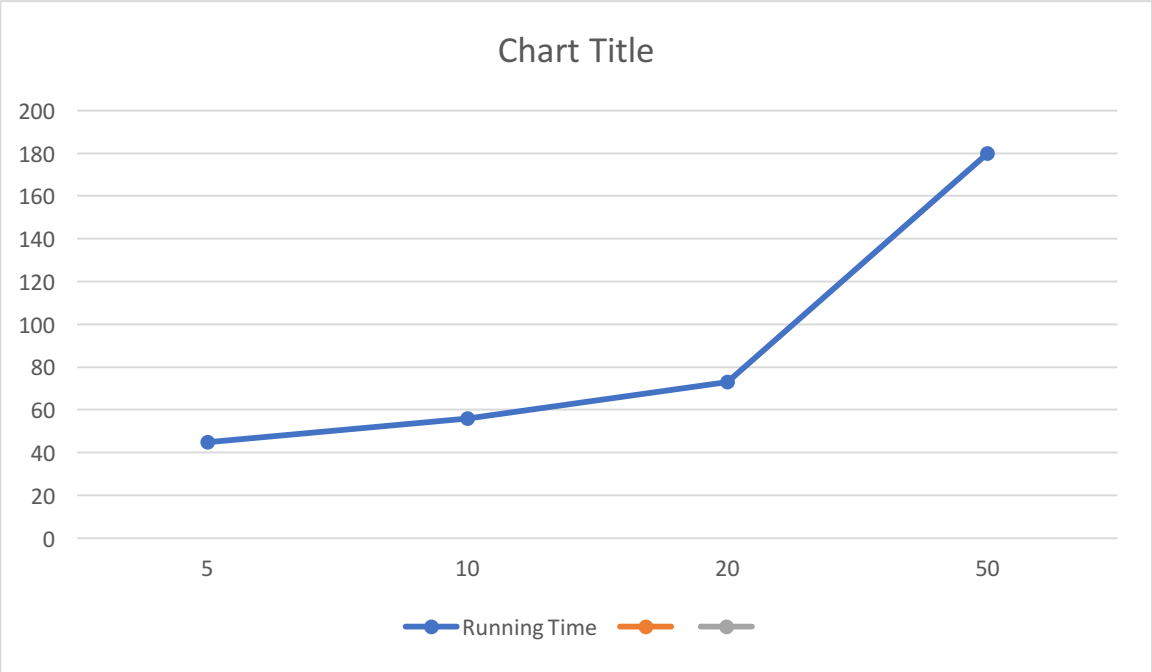
Running time v/s the number of trees in the model.



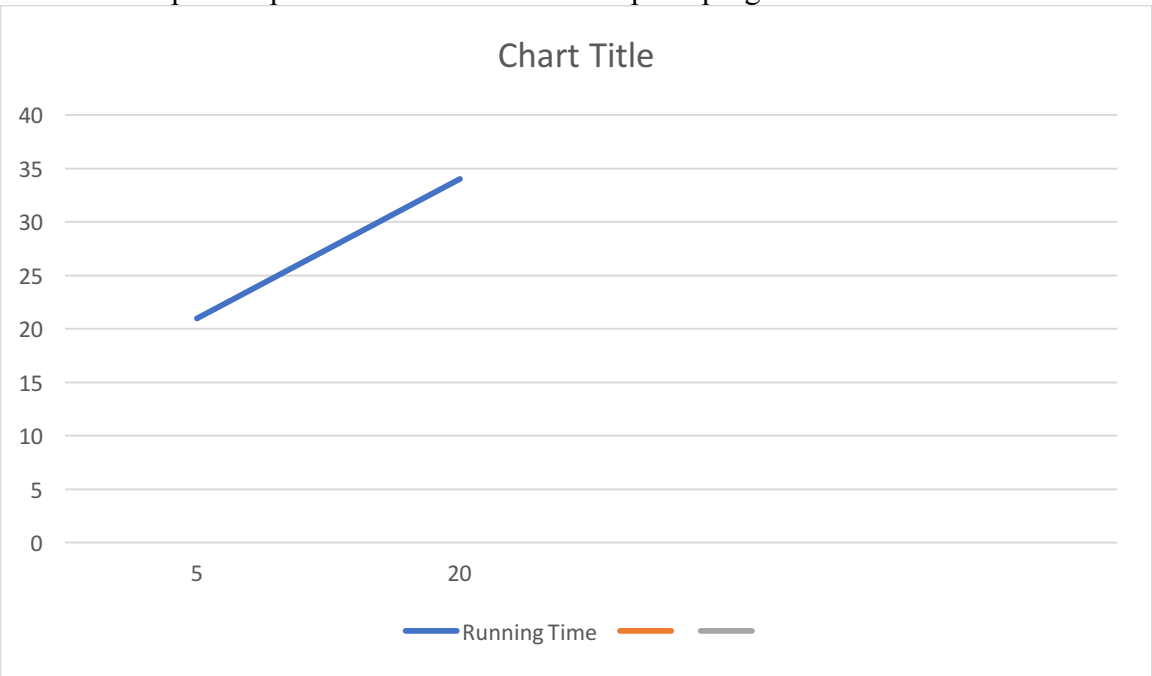
Number of machines: 5

Depth was kept constant at 5.

Running time v/s the depth of the Decision trees.



Number of Machines 5
The last data point represents our threshold to stop the program after 3 hrs.

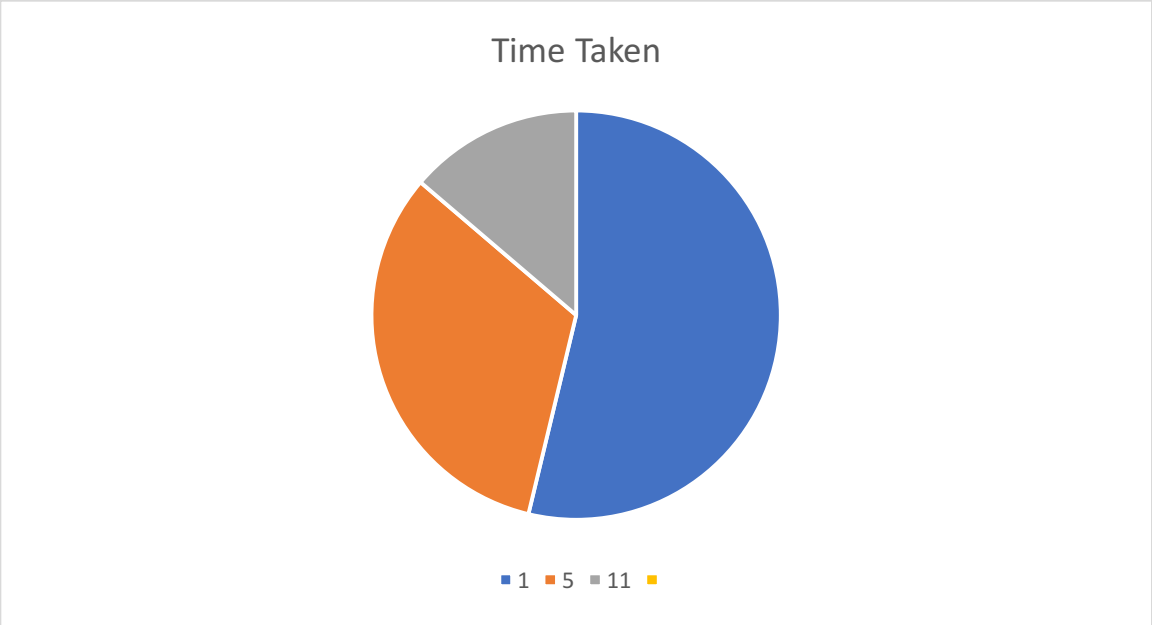


Number of Machines 1

Speedup:

Worker Nodes	1	5	11
Depth and Number of Trees	5 and 20	5 and 20	5 and 20
Time Taken	1hr 26 minutes	52 Minutes	22

Speed up for 5 machines: **1.65**
Speed up for 11 machines: **3.90**



The question and answers for the report:

(1) How many tasks are created during each stage of the model training process?

Ans: For 5:

Stage :0.0 Tasks:1
Stage :1.0 Tasks:386
Stage :2.0 Tasks:386
Stage :3.0 Tasks:386
Stage :4.0 Tasks:386
Stage :5.0 Tasks:40
Stage :6.0 Tasks:386
Stage :7.0 Tasks:40
Stage :8.0 Tasks:386
Stage :9.0 Tasks:40
Stage :10.0 Tasks:386
Stage :11.0 Tasks:40
Stage :12.0 Tasks:386
Stage :13.0 Tasks:40
Stage :14.0 Tasks:97
Stage :15.0 Tasks:97
Stage :16.0 Tasks:40
Stage :17.0 Tasks:97
Stage :18.0 Tasks:40

For 12:

Stage :0.0 Tasks:1
Stage :1.0 Tasks:386
Stage :2.0 Tasks:386
Stage :3.0 Tasks:386
Stage :4.0 Tasks:386
Stage :5.0 Tasks:96
Stage :6.0 Tasks:386
Stage :7.0 Tasks:96
Stage :8.0 Tasks:386
Stage :9.0 Tasks:96
Stage :10.0 Tasks:386
Stage :11.0 Tasks:96
Stage :12.0 Tasks:386
Stage :13.0 Tasks:96
Stage :14.0 Tasks:97
Stage :15.0 Tasks:97
Stage :16.0 Tasks:96
Stage :17.0 Tasks:97
Stage :18.0 Tasks:96

(2) Is data being shuffled?
Ans: For 5 and 12 both: Yes, data is shuffled 7 times.

(3) How many iterations are executed during model training (for methods that have multiple iterations)?
Ans: Total iterations will be 1 by our perpective and internally total iterations will be **maximumDepth * treeCount** = 20 * 5 = **100**

(4) You also need to find out, how to control performance. In particular, change the number of partitions or the actual partitioning itself and report how this affects running time.
Ans) In random forest model the partitions depends on the parameters we set , here basically it will be number of trees and the depth associated with a random forest model. The results vary when we change the depth or the number of trees as depicted in the table and effects can be observed in the table(a) and table(b).

Pseudo Code:

```
Object ModelFile {
    Method main(args[]) {
        // initialize SparkContext and SqlContext
        sc <-   SparkContext
        sqlContext <- SqlContext
        // read csv input into RDD
        rdd = sc.textFile(args(0)+"/*.csv")
        // transform data
        data = rdd.map(_.split(",")).map(_.map(_.toDouble))
        labeledPoints <- map(point => label) on rdd
        trainingData <- random split from labeledPoints
        // read csv input into RDD
        rdd2 = sc.textFile(args(2)+"/*.csv")
        labeledPoints2 <- map(point => label) on rdd2
        /******* Classification ***********/
        // initialize random forest classifier properties
        impurity = Gini
        maximumDepth = 5
        treeCount = 20
        featureSubsetStrategy = "auto"
        seed = 5043
        // train model
        model <- train classifier using random forest
        // predict the labels using trained model
        labeledPredictions = labeledPoints2.map { labeledPoint =>
            val predictions = model.predict(labeledPoint.features)

        // save the prediction
        labeledPredictions.map(x => x._2.toInt).saveAsTextFile(args(1))

        // print accuracy using evauluation metric
        val evaluationMetrics = new MulticlassMetrics(labeledPredictions.map(x =>
        (x._1, x._2)))
        println(evaluationMetrics.precision) } }
```