# CS5543 Real -Time Big Data Analytics
# Project Report
# Read Video

Akhilesh Gattu (5) Latha Muddu (15)

***Abstract*— The main moto of our project is to classify the swimming styles by analyzing each frame in a swimming video. For this we use certain tools like spark to build the model, Storm for real time streaming, Kafka for communication and Mongo database to save the results. We gave the data to the storm spout which in turn is given to the storm bolts. Here the analysis is done and the swimming style is decided. We save the results on to the mongo database and display these results on the UI.**

***Index Terms*—Storm, Spark, Kafka, Spout, Bolt, Backstroke, Sidestroke**

## I. INTRODUCTION

The track record of each player is maintained and the swimming style for each frame is determined. Say suppose a person doesn't know anything about the swimming styles and the video does not provide any information about the swimming styles. In that case our application will be very useful as our application analyzes the video and displays the swimming style on the UI. So we came up with an approach describing the swimming style of each player. We have done this by building decision tree model using Spark. This model is retrieved in storm and video features are analyzed in order to classify the swimming style.

## II. Related Work

We worked on different review papers as a background work to come up with a correct approach as how to classify the swimming styles. We dealt with the basics of storm, Kafka, Spark first and got to know the flow of data.

### Features

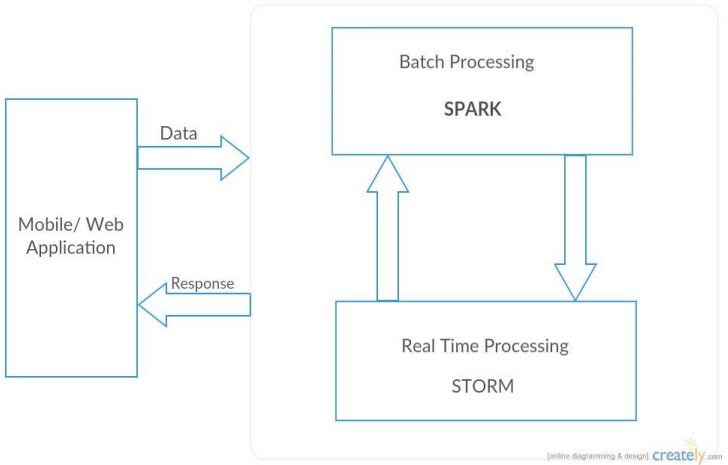
1) Identifying the key frames
2) Creating highlighted video
3) Maintaining track record of the player
4) Recognizing swimming style

**Data Sources:** YouTube, TRECVID
Analytic Tools: Spark and Storm

**Analytical Tasks:** Processing each and every frame individually, tracking each player in
Storm Bolt.
**Expected Input:** Swimming Video
**Expected Output:** Mainframes, swimming style, highlighted video.

## III. Proposed Solution

### Architecture




### Tools & Technologies:

1. Android (Mobile Application)
2. Open IMAJ
3. Spark (Batch Processing, Model)
4. Storm (Real Time Streaming)
5. Kafka (Communication)
6. Mongo Database

### Open IMAJ

Open IMAJ consists of set of tools and libraries to analyze the contents in a video. Though it contains many libraries, each library can be used independently.

### Spark

Spark is used to process large amounts of data. Spark works faster than many other tools comparatively. Spark usually doesn't have any dependency issues i.e.; we can run anywhere. Spark is easy to use. Spark runs 100 times better than Hadoop map reduce. The driver function in spark runs the main function. Various operations can be executed in parallel. Spark consists of the resilient distributed database which consists of number of elements across the nodes which executes in parallel in a cluster. The main advantage with the storm is that the same code which we used for batch processing can be used for the stream processing. Spark also supports SQL queries along with map and reduce operations. Using spark is very less expensive. The data processing is well optimized in spark. When there is a need to work on the same data multiple times spark stores the intermediate results in the memory so that it can be used in a better way. Spark first stores the data on to the memory later it will be transferred to the disk. The program in spark is written in Scala which will be executed on the java. Spark is also cluster computing.

### Storm:

Storm is a distributed real time system that can process large amounts of data with high velocity. The transmission rate of data in Storm is about million records per second. Storm uses Spouts and bolts to transmit data. The storm topology is like a directed acyclic graph with spouts and bolts as the vertices for the graph.

The characteristics of storm includes:

**Fast:** Processing of data is very fast and simple.

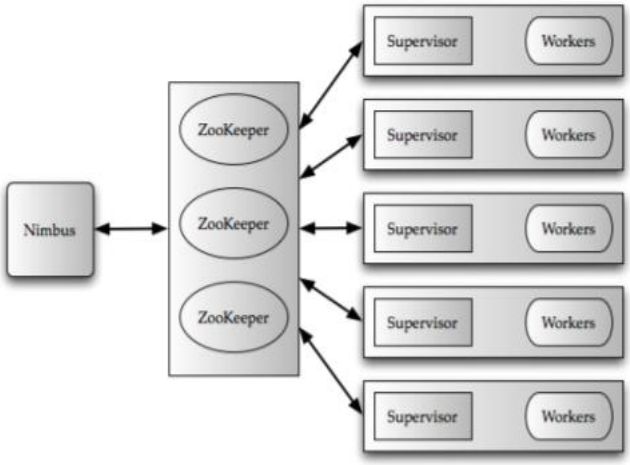
**Scalable:** In storm data is scalable as it supports parallel computations which can run across cluster of machines.

**Fault-Tolerance:** The system is fault tolerant because when the worker node dies, Storm will automatically restart them i.e., if the worker node dies, on the other node the worker will automatically be restarted.

**Reliable:** In storm the tuples will be processed at least once or exactly once. The messages will not be processed until unless the tuple fails in processing.

**Easy to operate:** Deployment of data is easy in storm.

Storm consists of Spouts and Bolts. Spout is a data generator. Bolts process the incoming tuples and pass them to next bolt or send the results.



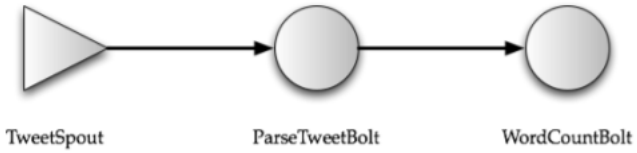

The above figure shows the high level storm topology. The topology is submitted to the nimbus by the client. Nimbus is responsible for the execution of the work flow. Storm supports certain strategies:

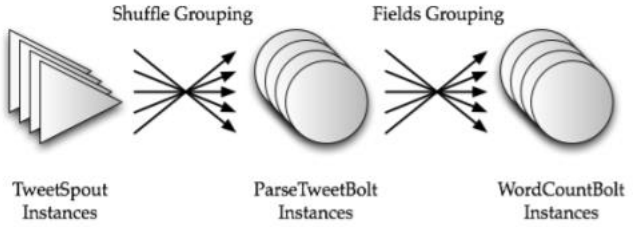
Shuffle: The tuples are randomly partitioned in shuffle grouping.

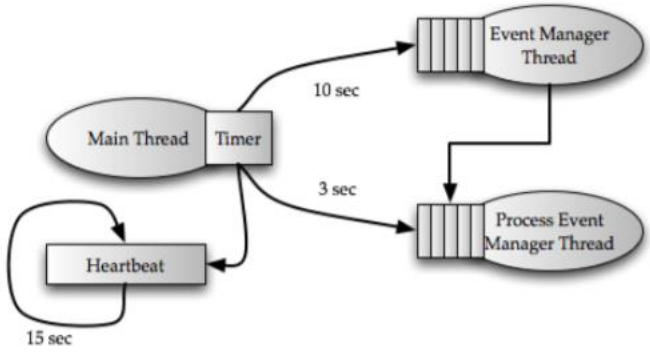
Fields: In field grouping the subsets are hashed.

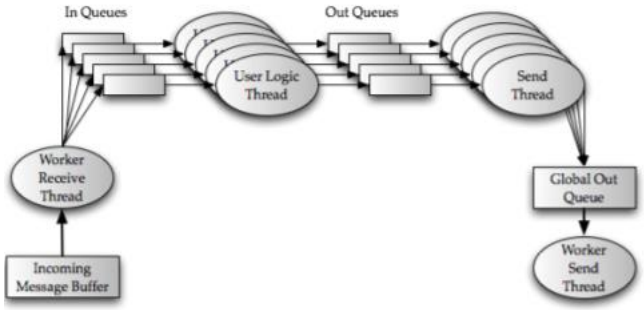
All: The complete stream is replicated over the consumer tasks.

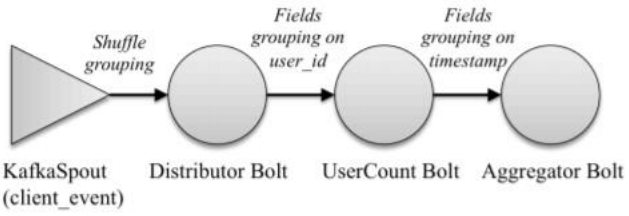
Global: The complete stream is sent to a single bolt in Global grouping.




**Kafka**

Kafka is an open source stream data processing platform developed by apache which is written in Scala and java. Kafka is a distributed message broker system that can handle large amounts of data per second.

The streaming of data has the following capabilities:

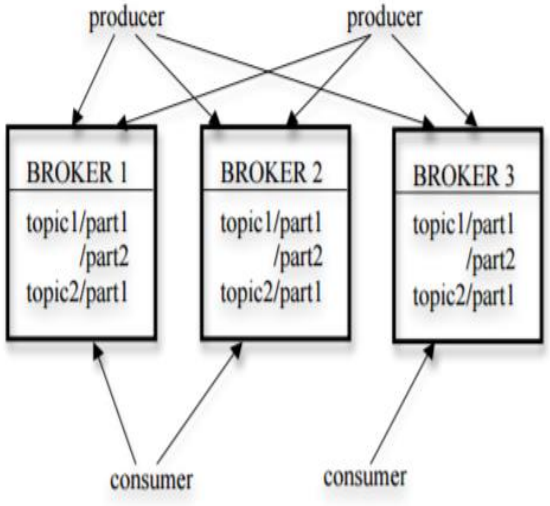
1. We can publish and subscribe to streams of data or records.
2. The streams of records can be stored in a fault tolerant way.
3. The streaming is done as soon as the record occurs.

Kafka is best for two kinds of applications:

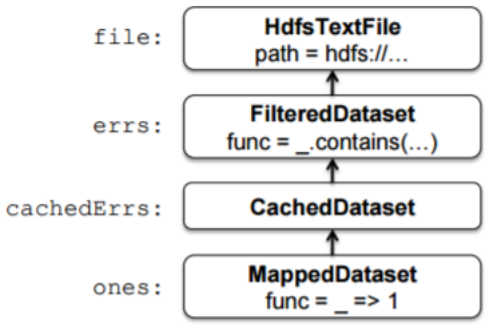
1. To build real time stream data pipelines which can fetch the data reliably between the systems.
2. To build real time streaming applications which can transform the streams of data easily.

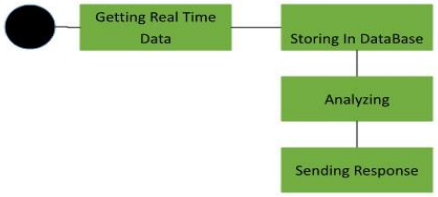
Kafka uses 4 API's mainly

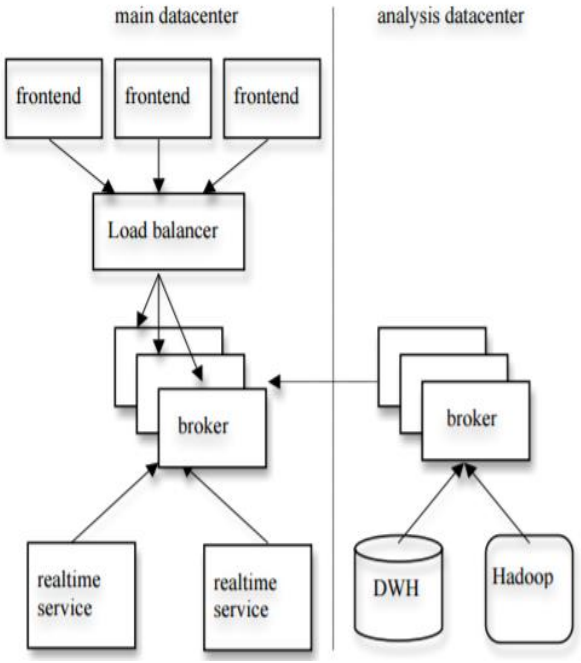
1. Producer API – used to publish streams of records to the Kafka topics
2. Consumer API – Allows an application to subscribe to one or more topic and process them.
3. Streams API – This API allows to consume an input stream for one or more topics and produce an output stream to one or more topics accordingly.
4. Connect API – Allows to build and reuse the producers or consumers that connects the Kafka topics

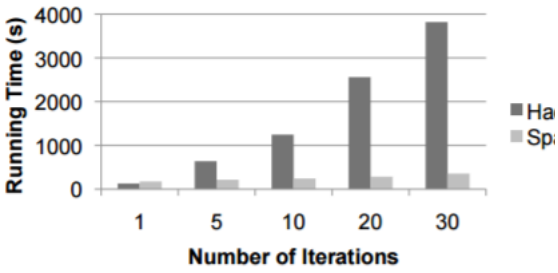



Kafka architecture consists of multiple brokers. Each topic is divided into multiple parts to balance the load. A message can be published by producers and consumers.



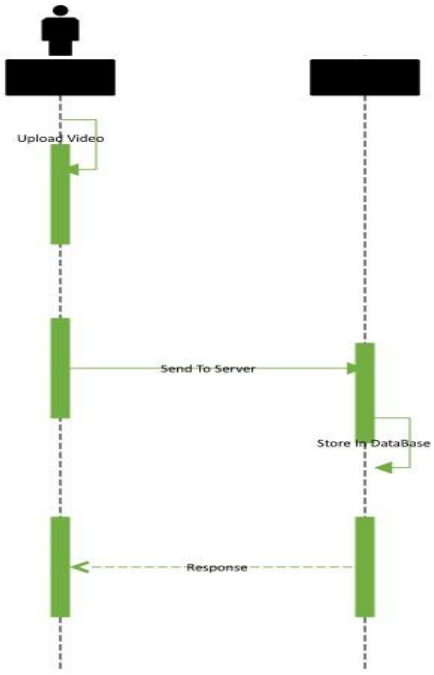




**Activity Diagram**

The activity diagram for our project is shown above. First we get the real time data then store it in the database, analyze the video and send back the response.

We take this response from the mongo database and display the results on the UI. The video which we take as input is divided into multiple frames and each frame is analyzed separately and the swimming style is decided. This operation is done in the storm bolt.

**Sequence Diagram**



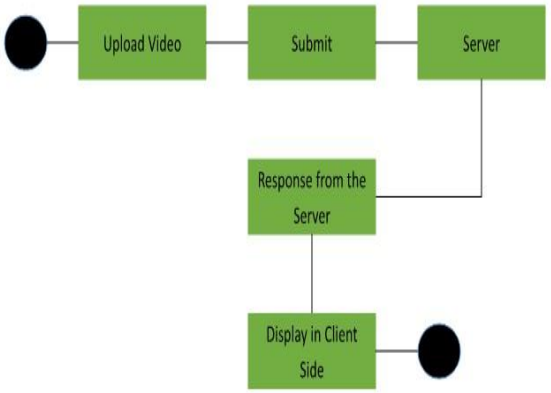

In the sequence diagram we take the video, send to the server and store that video in the database

initially, analyze that video and send back the response to the client.

**Operation Specification**
**Input:** Video
**Output:** Text Description, Video

We take a swimming video as input and divide the video into number of frames. Now each frame is analyzed and sent to the bolts. The bolt checks for a particular swimming style and returns true if it is that swimming style else returns false. So now even a layman can easily understand the what swimming style it is.

**Activity Diagram:**

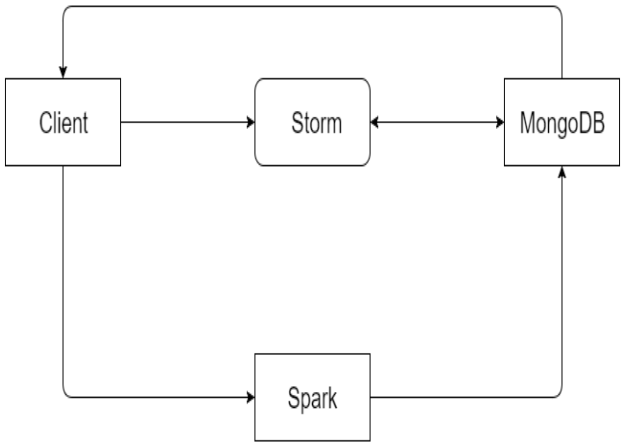



Here we upload a video and click on the submit. This is sent to the server. The response from the server is taken and displayed onto the UI.

**Project Flow:**

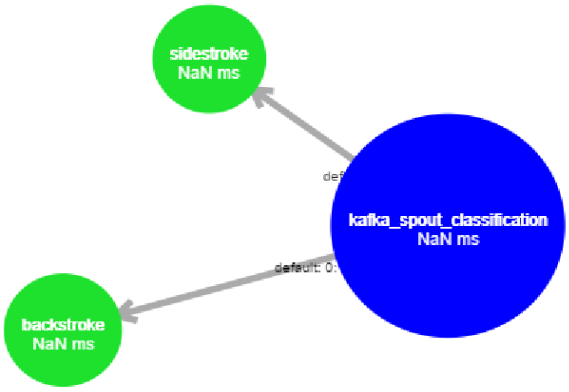
Initially, we take a video, extract its features, build a model and then test it. For building the model we take a swimming video, extract its features. We save these features in a text file and give it to the Spark. Here the model is built. We built a decision tree model and push this to the mongo database.

On the other hand, we built the Storm topology. We built a topology consisting of one spout and two bolts. Each bolt represents a swimming style. We have taken Backstroke and Sidestroke swimming styles into consideration. Now we take the model from the mongo database to the storm. When a new video comes in, it is given as input to the storm Spout which in turn is given to the Storm bolts. Now the first bolt checks for the backstroke swimming styles and returns true if the frame consists of backstroke swimming style else returns false. The second bolt also checks for the sidestroke swimming style and returns true if it is a sidestroke swimming style else returns false. We save these results on to the mongo database and display it on the UI.

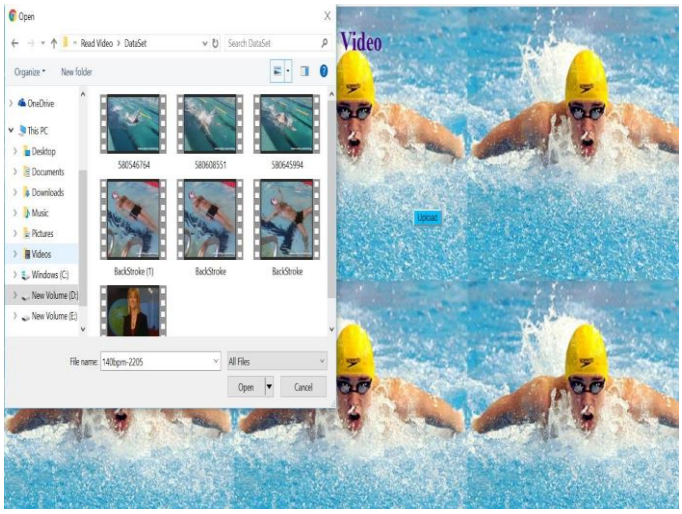



The above figure shows the flow of our project.

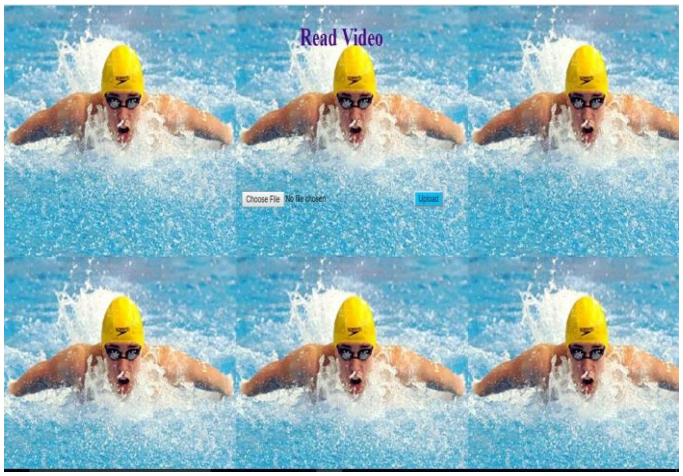
**Topology:**

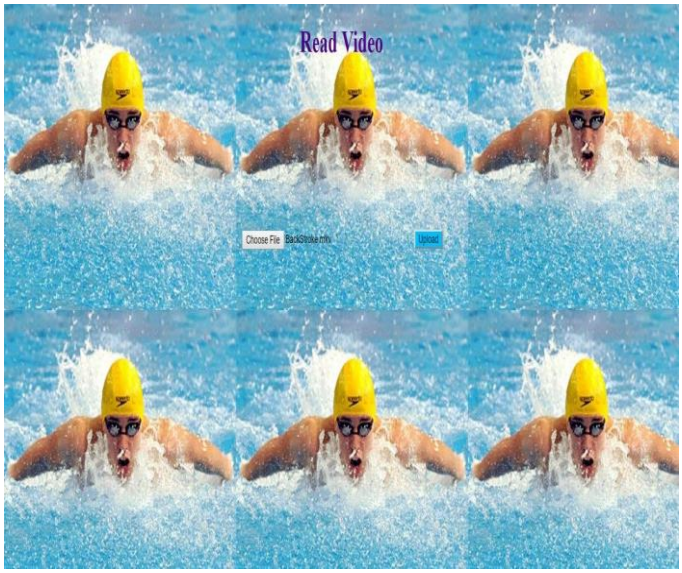


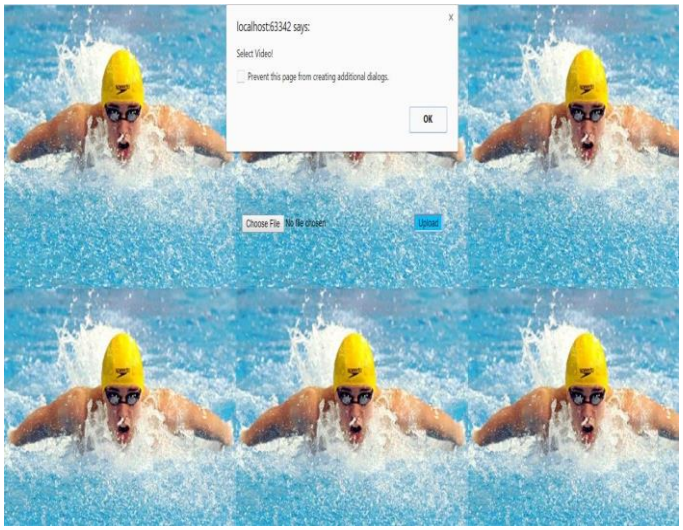
Our topology consists of a single Spout. This Spout gets the input from the Kafka. The topology consists of two bolts one for sidestroke swimming style and the other for backstroke swimming style. The sidestroke bolt takes the frames and returns true if the frame consists of sidestroke swimming style else the decision is false. The backstroke bolt returns true if the frames that came in consists of the backstroke swimming style else the decision is false.




## Results And Evaluation



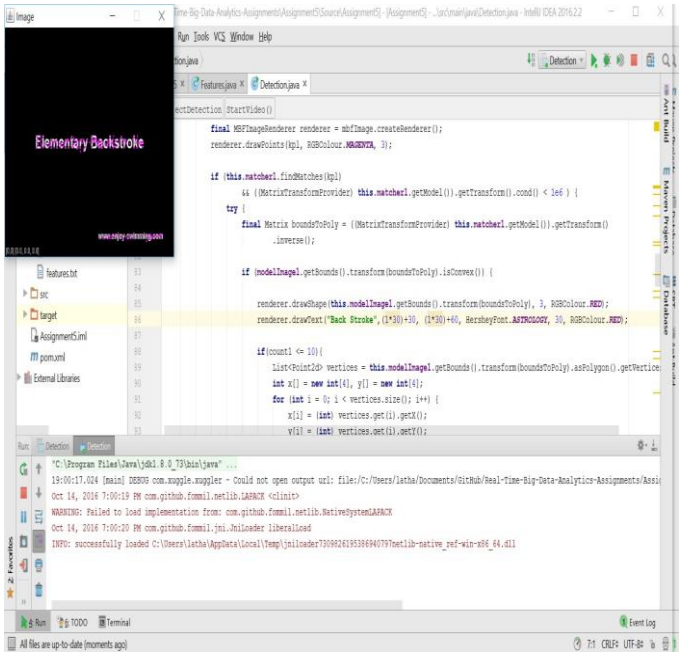


Image
Elementary Backstroke
www.enjoy-swimming.com

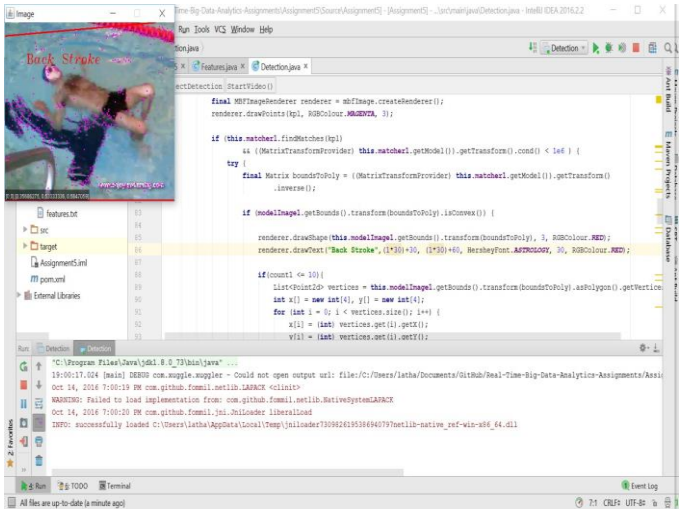


Image
Back Stroke

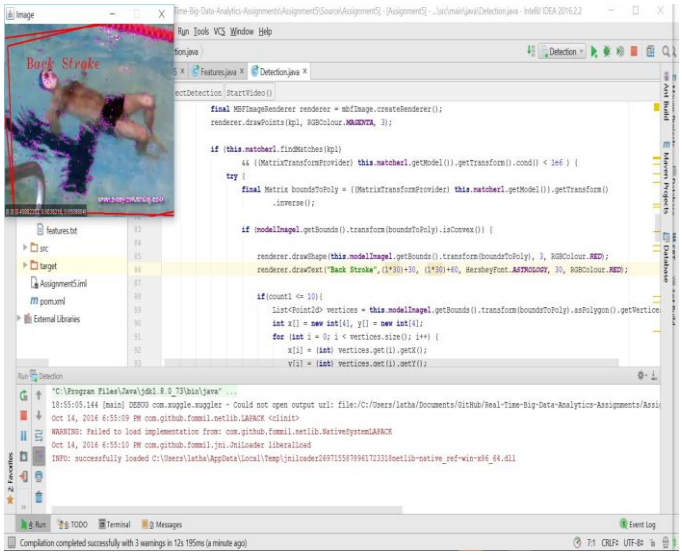


Image
Back Stroke

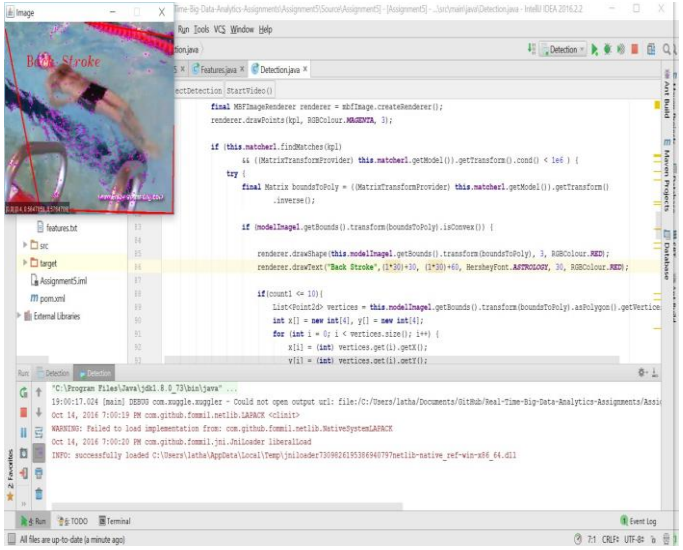
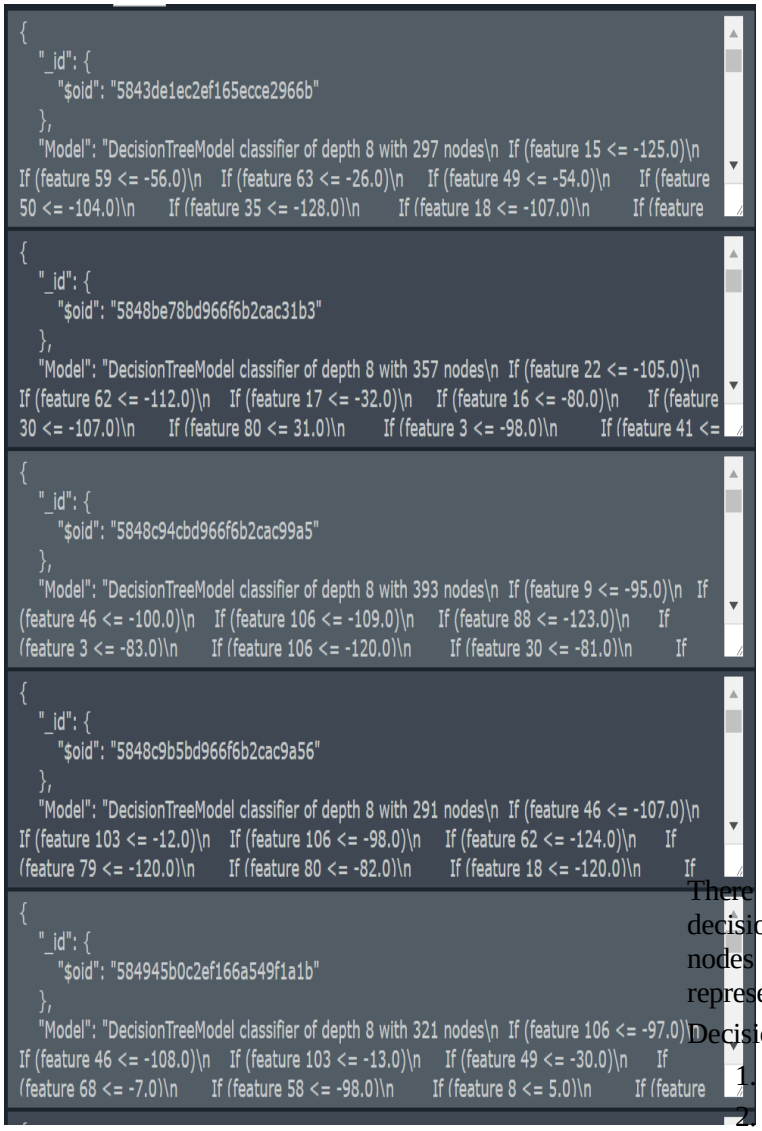


Image
Back Stroke




The above is the screenshot of the data that is stored in mongo database. In the below screenshot we can see the decision as true for few frames and false for some cases. For the one's the decision is true means that that particular frame belongs to that swimming style. If the decision is false it does not belong to that swimming style.


**Decision Tree Model**

In decision tree model each node is a test attribute. There are three kinds of nodes in decisions tree namely decision nodes, chance nodes and end nodes. Decision nodes are represented in squares. Chance nodes are represented in circles and end nodes by triangles.

Decision tree is of two types:

1. Classification tree
2. Regression tree

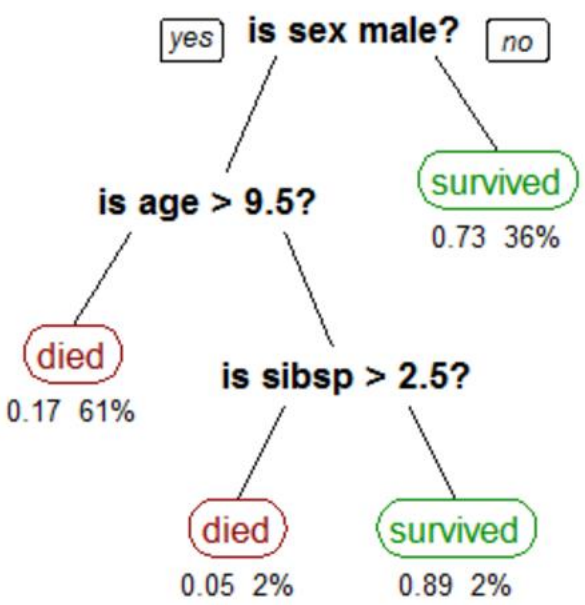
The techniques for the decision tree are

1. Bagging
2. Random forest
3. Boosted trees
4. Rotation forest

The advantages of using decision tree is

1. Simple to understand
2. Only little preparation is required
3. White box model is used
4. Robust
5. Performance is very high

Below is a sample decision tree.




**Project Management**

In the first increment we detected the key frames and tracked the objects. First the video is taken as input and divided into number of frames. Next the client side UI is built. Later we worked on the Storm and Spark to understand the work flow. We built a decision tree model using spark. We also built the topology with one spout and two bolts. The model from the mongo database is given to the storm where the video is analyzed. The results from these bolts are saved to the mongo database. These results are later displayed onto the UI.

**Confusion Matrix**

| | | Predicted | | |
|---|---|---|---|---|
| | | Cat | Dog | Rabbit |
| Actual class | Cat | 5 | 3 | 0 |
| | Dog | 2 | 3 | 1 |
| | Rabbit | 0 | 2 | 11 |

It is a two dimensional matrix consisting of actual and predicted data.

| n=165 | Predicted: NO | Predicted: YES |
|---|---|---|
| Actual: NO | 50 | 10 |
| Actual: YES | 5 | 100 |

**Accuracy**

We took the testing and training data and performed the evaluation test. We got the accuracy as 72%.

**Responsibility:** Both of us have divided the tasks equally and worked on it. At the end of each task we shared our knowledge in that task.

**Contributions:** Both of us worked on all the tasks equally

Akhilesh Gattu: 50 % & Latha Muddu: 50 %

**XVI. Future Scope:**

We could detect which event is going on and winning can be predicted.

**Github:**

https://github.com/AkhileshGattu/Read-Video/tree/master/Source

**References:**


1. Spark : Cluster Computing with working Sets by Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica University of California, Berkeley
2. Storm@twitter by Ankit Toshniwal, Siddarth Taneja, Amit Shukla, Karthik Ramasamy, Jignesh M. Patel*, Sanjeev Kulkarni, Jason Jackson, Krishna Gade,
3. Kafka: A distributed messaging system for log processing by Jay Kreps, Neha Narkhede, Jun Rao