DEEP LEARNING BASED ON FOOD IMAGE RECOGNITION AND PORTION SIZE DETERMINATION

Team Name : New Crew

TEAM MEMBERS

Shreyaa Sridhar (21)
Khushbu Kolhe (9)
Geovanni Nicque West (23)
Naga Venkata Satya Pranoop Mutha (15)

INDEX

	Topic	Page No
I	Project Objectives	3
II	Approach	4
III	Related Work	5
IV	Application Specification and Implementation	6
V	Project Management	17
VI	Bibliography	18

PROJECT OBJECTIVES

- ## Motivation

Good nutrition is an important part of leading a healthy lifestyle. Food choices that we make each day affect our health - how we feel today, tomorrow and in the future. In addition to that food portion also plays an important role because it allows for you to have a tight handle on how many calories you are presumably taking in. This means eating what your body needs instead of mindlessly overindulging!

- ## Significance

The key idea of this project is to train our model to recognize the food image we provide and identify calorie count. Also to predict the food portion size and recalculate calories based on the image.

- ## Features :Use Case/Scenario

  - Food image recognition
  - Calorie count
  - Food portion size

APPROACH

- ## Data Sources

Dataset - Food-101 dataset (https://www.vision.ee.ethz.ch/datasets_extra/food-101/)

- ## Analytic Tools

The analytic tool used is the increment one of the project is **Clarifai** and **OpenIMAJ**

- ## Analytical Tasks

We took our food dataset in the form of video(.mkv format) and then determined the keyframe in the video, later we passed the key frame to the Clarifai API. Clarifai API find the objects in the image and annotated objects according from higher to lower confidence score. Also, we have classified all our global warming data set and then trained model to predict the test data set in Client UI using Random forest model, Naive Bayes Model and Decision Tree Model.

- ## Expected Inputs/ Outputs

Input - food.mkv , Images from Food-101 dataset

Output - Key frames identified, Image Categorization, Image Prediction, Statistical parameters like accuracy, precision,confusion matrix etc.

- ## Algorithms

  - ★ K - Means
  - ★ Decision Tree
  - ★ Random Forest
  - ★ Naive Bayes

RELATED WORK

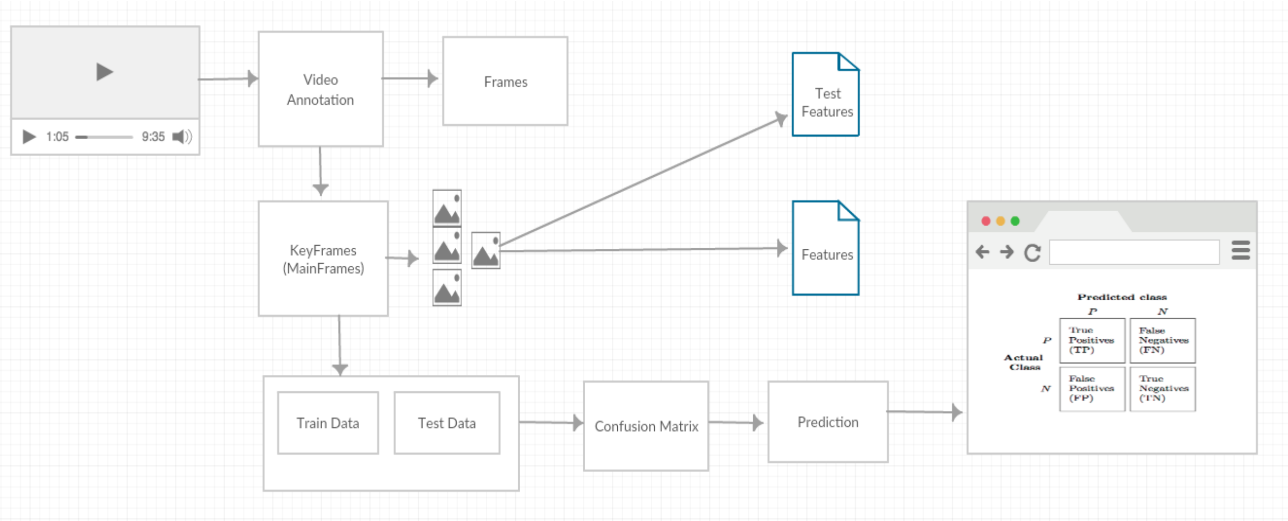
- **Literature Reviews**

http://oar.a-star.edu.sg/jspui/bitstream/123456789/2299/1/05-NTCIR13-LIFELOG-XuQ.pdf
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5537777/
https://arxiv.org/ftp/arxiv/papers/1606/1606.05675.pdf
http://ieeexplore.ieee.org/document/4649292/

APPLICATION SPECIFICATION & IMPLEMENTATION

System Specification (Big Data Analytics Server/Client)

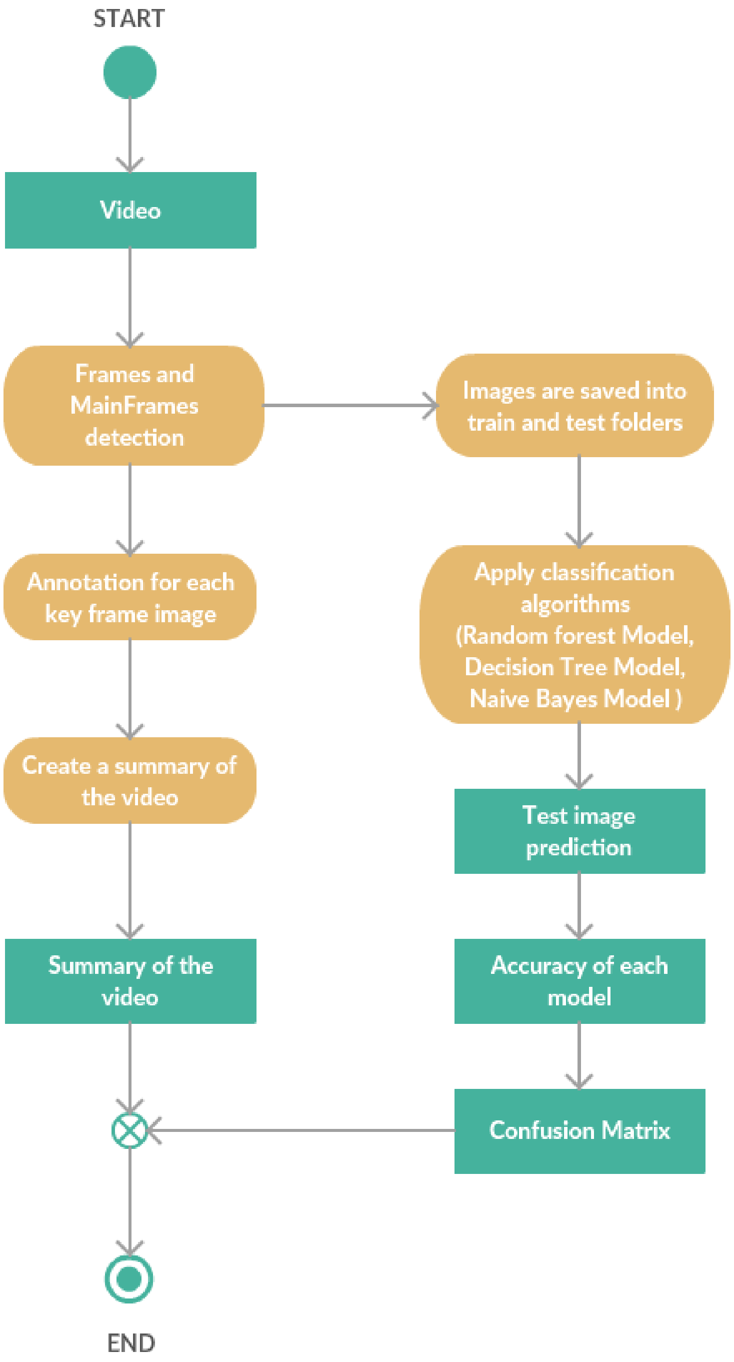
- **Software Architecture**



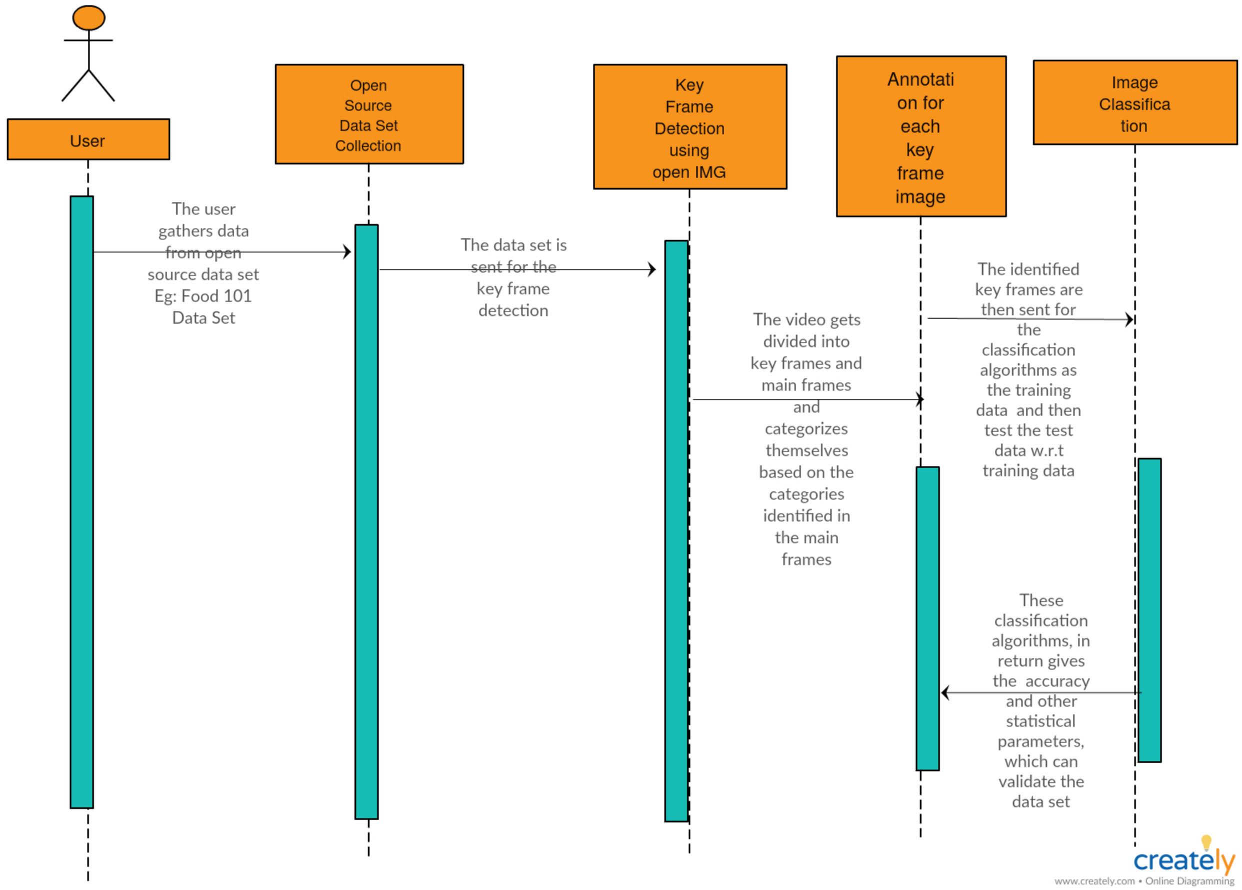
		Predicted class	
		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

- **Features, workflow, technologies**

★ Activity Diagram (workflow, data, task)


START
Video
Frames and MainFrames detection
Images are saved into train and test folders
Annotation for each key frame image
Apply classification algorithms (Random forest Model, Decision Tree Model, Naive Bayes Model)
Create a summary of the video
Test image prediction
Summary of the video
Accuracy of each model
Confusion Matrix
END

★ Sequence Diagram (interaction/collaboration)


User
Open Source Data Set Collection
Key Frame Detection using open IMG
Annotation for each key frame image
Image Classification
The user gathers data from open source data set Eg: Food 101 Data Set
The data set is sent for the key frame detection
The video gets divided into key frames and main frames and categorizes themselves based on the categories identified in the main frames
The identified key frames are then sent for the classification algorithms as the training data and then test the test data w.r.t training data
These classification algorithms, in return gives the accuracy and other statistical parameters, which can validate the data set
creately
www.creately.com • Online Diagramming

★ Feature Specification

- Food Recognition
- Food Quantity Comparison

★ Operation Specification (Input/output, exceptions)

Input - food.mkv , Images from Food-101 dataset

Output - Key frames identified, Image Categorization, Image Prediction, Statistical parameters like accuracy, precision, confusion matrix etc.

EXISTING APPLICATIONS/SERVICES USED: NAME, DESCRIPTION, URL

- We have used **Clarifai API service** which automatically tags the images or video file in such a way ensuring a quick organization, management and searching throughout the content.
  URL: https://www.clarifai.com/
- We have also used **OpenIMAG** for key frame detection.

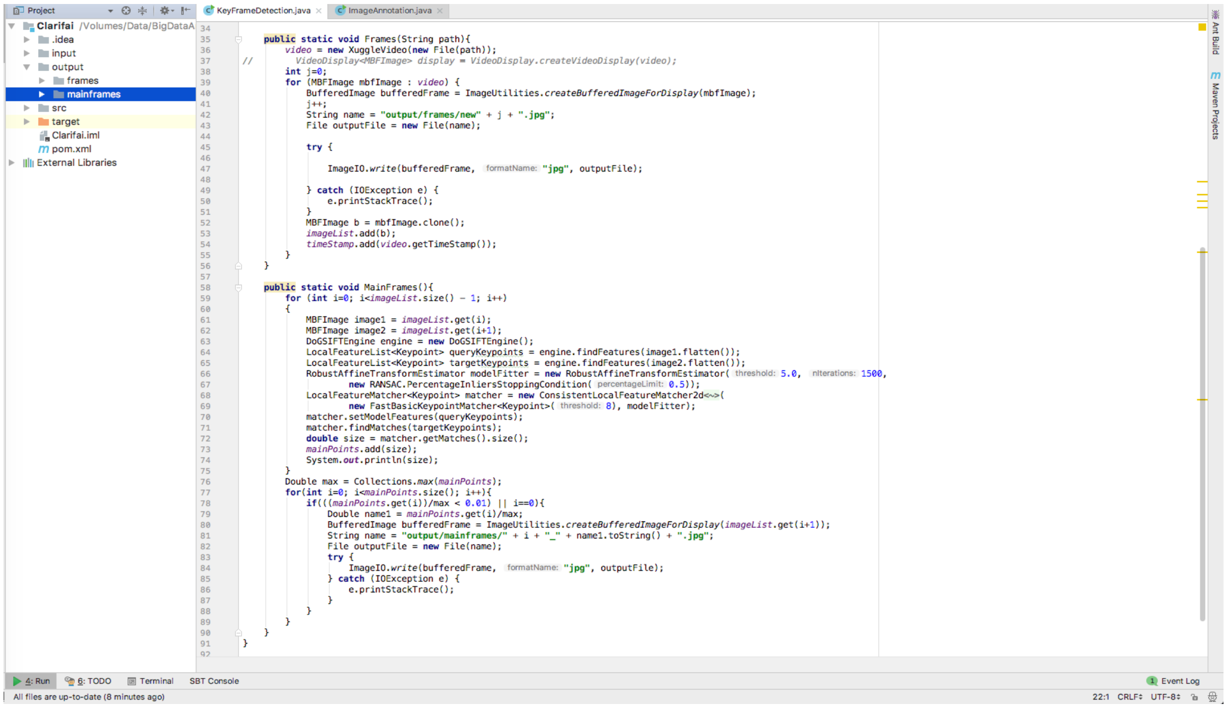
IMPLEMENTATION OF YOUR APPLICATION USING CLARIFAI API

First,we took our dataset of food in form of video(.mkv format) which contains variety of food and then determined the keyframe images from that video. Then we passed that key frame images to the Clarifai API. Clarifai API find the objects in the image and annotated objects which provides the summary of the video.

Steps Involved

1. Spark libraries are imported and spark is initialized.
2. Here we are analyzing video using Clarifai API and the video is split into keyframes and mainframes based on distinct frames.
3. Code ran successfully and output was generated.

Frames and MainFrames generated

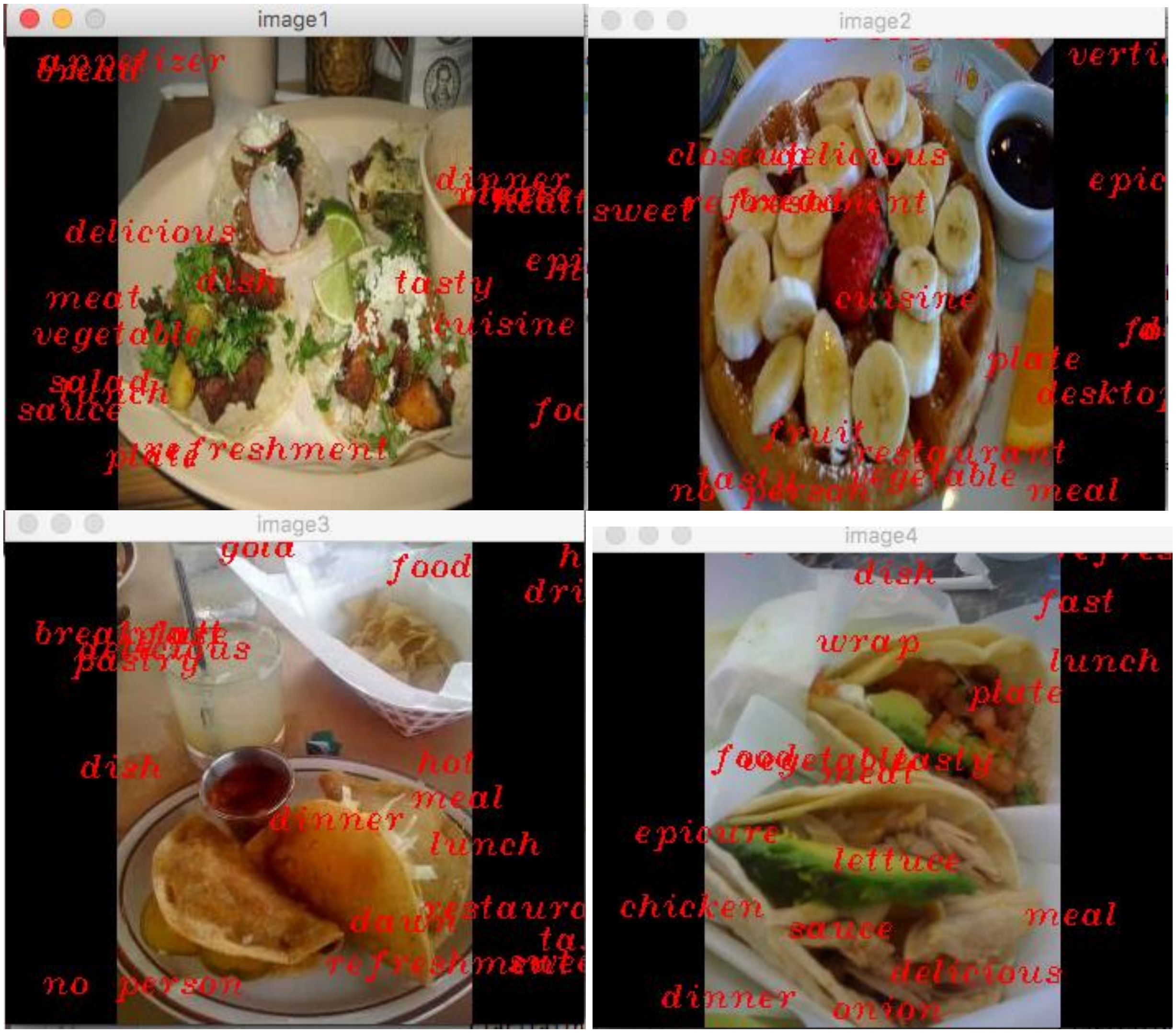


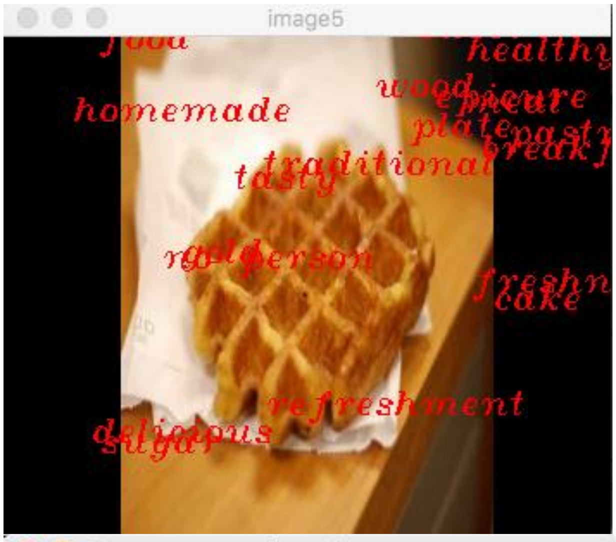
```java

    public static void Frames(String path){
        video = new XuggleVideo(new File(path));
//        VideoDisplay<MBFImage> display = VideoDisplay.createVideoDisplay(video);
        int j=0;
        for (MBFImage mbfImage : video) {
            BufferedImage bufferedFrame = ImageUtilities.createBufferedImageForDisplay(mbfImage);
            j++;
            String name = "output/frames/new" + j + ".jpg";
            File outputFile = new File(name);

            try {

                ImageIO.write(bufferedFrame,  formatName: "jpg", outputFile);

            } catch (IOException e) {
                e.printStackTrace();
            }
            MBFImage b = mbfImage.clone();
            imageList.add(b);
            timeStamp.add(video.getTimeStamp());
        }
    }

    public static void MainFrames(){
        for (int i=0; i<imageList.size() - 1; i++)
        {
            MBFImage image1 = imageList.get(i);
            MBFImage image2 = imageList.get(i+1);
            DoGSIFTEngine engine = new DoGSIFTEngine();
            LocalFeatureList<Keypoint> queryKeypoints = engine.findFeatures(image1.flatten());
            LocalFeatureList<Keypoint> targetKeypoints = engine.findFeatures(image2.flatten());
            RobustAffineTransformEstimator modelFitter = new RobustAffineTransformEstimator( threshold: 5.0,  nIterations: 1500,
                    new RANSAC.PercentageInliersStoppingCondition( percentageLimit: 0.5));
            LocalFeatureMatcher<Keypoint> matcher = new ConsistentLocalFeatureMatcher2d<~>(
                    new FastBasicKeypointMatcher<Keypoint>( threshold: 8), modelFitter);
            matcher.setModelFeatures(queryKeypoints);
            matcher.findMatches(targetKeypoints);
            double size = matcher.getMatches().size();
            mainPoints.add(size);
            System.out.println(size);
        }
        Double max = Collections.max(mainPoints);
        for(int i=0; i<mainPoints.size(); i++){
            if(((mainPoints.get(i))/max < 0.01) || i==0){
                Double name1 = mainPoints.get(i)/max;
                BufferedImage bufferedFrame = ImageUtilities.createBufferedImageForDisplay(imageList.get(i+1));
                String name = "output/mainframes/" + i + "_" + name1.toString() + ".jpg";
                File outputFile = new File(name);
                try {
                    ImageIO.write(bufferedFrame,  formatName: "jpg", outputFile);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}

```

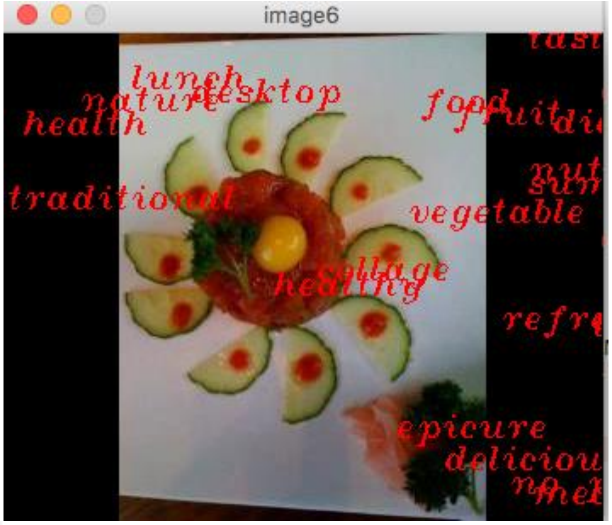
All files are up-to-date (8 minutes ago)

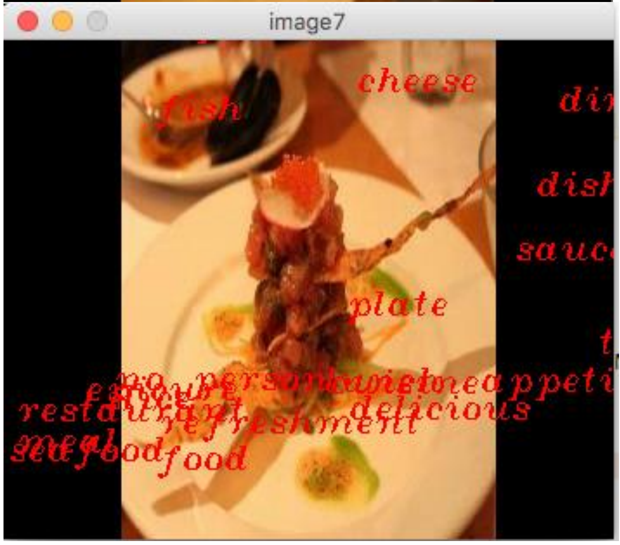
Output Generated

Clarifai API has detected these based on the image.









IMPLEMENTATION OF IMAGE CLASSIFICATION

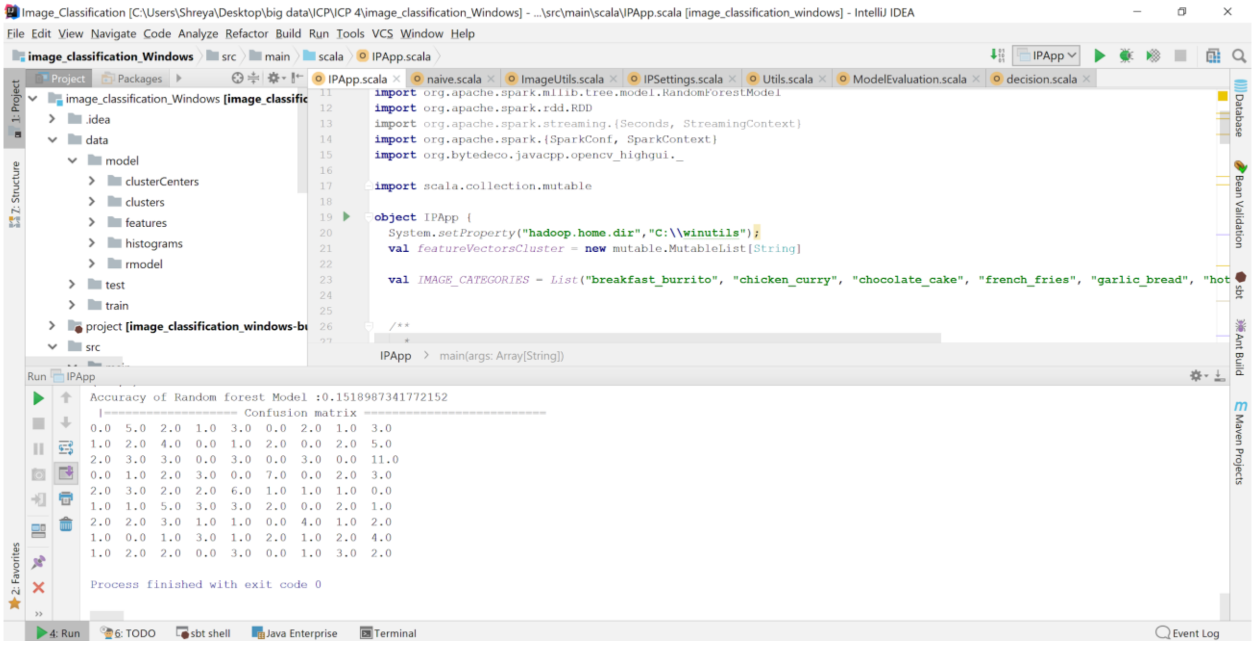
First, we have taken images from Food-101 dataset and have created a train and test folders which acts as the input. Model is generated based on the data provided. Histogram, accuracy, test image prediction and confusion matrix are obtained as output.

Here we are calculating the accuracies of the model using Random Forest , Naive Bayes and Decision Tree Models.

1. Spark libraries are imported and spark is initialized.
2. We have collected the input from Food 101 dataset .
3. Split data into training (70%) and test (30%) and function is applied.
4. Model is built and confusion matrix is obtained.

5. Accuracy of the Model is calculated.
6. Code ran successfully and outputs were generated. Also the model was saved.

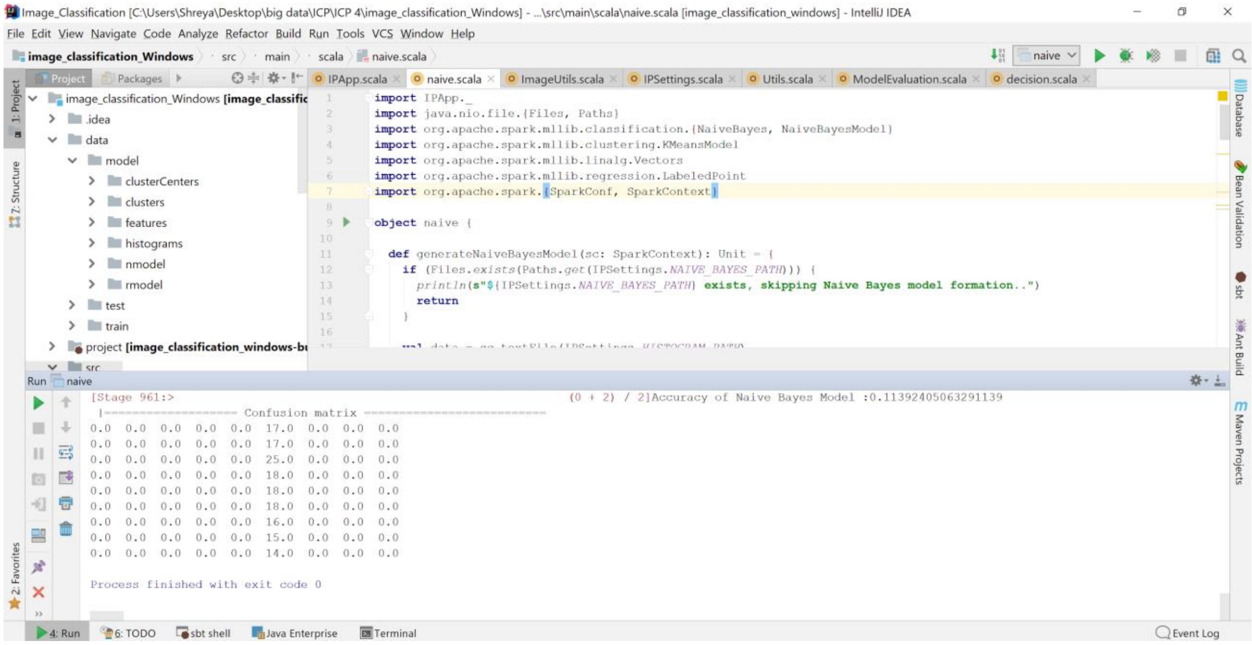
Random Forest Model


Image_Classification [C:\Users\Shreya\Desktop\big data\ICP\ICP 4\image_classification_Windows] - ...\src\main\scala\IPApp.scala [image_classification_windows] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
image_classification_Windows src main scala IPApp.scala
IPApp
Project Packages
IPApp.scala naive.scala ImageUtils.scala IPSettings.scala Utils.scala ModelEvaluation.scala decision.scala
image_classification_Windows [image_classific
.idea
data
model
clusterCenters
clusters
features
histograms
rmodel
test
train
project [image_classification_windows-bu
src
IPApp > main(args: Array[String])
Run IPApp
4: Run 6: TODO sbt shell Java Enterprise Terminal Event Log
1: Project 7: Structure 2: Favorites Database Bean Validation sbt Ant Build Maven Projects

```
import org.apache.spark.mllib.tree.model.RandomForestModel
import org.apache.spark.rdd.RDD
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark.{SparkConf, SparkContext}
import org.bytedeco.javacpp.opencv_highgui._

import scala.collection.mutable

object IPApp {
  System.setProperty("hadoop.home.dir","C:\\winutils");
  val featureVectorsCluster = new mutable.MutableList[String]

  val IMAGE_CATEGORIES = List("breakfast_burrito", "chicken_curry", "chocolate_cake", "french_fries", "garlic_bread", "hot


  /**
```

```
Accuracy of Random forest Model :0.1518987341772152
 |=================== Confusion matrix ==========================
0.0  5.0  2.0  1.0  3.0  0.0  2.0  1.0  3.0
1.0  2.0  4.0  0.0  1.0  2.0  0.0  2.0  5.0
2.0  3.0  3.0  0.0  3.0  0.0  3.0  0.0  11.0
0.0  1.0  2.0  3.0  0.0  7.0  0.0  2.0  3.0
2.0  3.0  2.0  2.0  6.0  1.0  1.0  1.0  0.0
1.0  1.0  5.0  3.0  3.0  2.0  0.0  2.0  1.0
2.0  2.0  3.0  1.0  1.0  0.0  4.0  1.0  2.0
1.0  0.0  1.0  3.0  1.0  2.0  1.0  2.0  4.0
1.0  2.0  2.0  0.0  3.0  0.0  1.0  3.0  2.0

Process finished with exit code 0
```

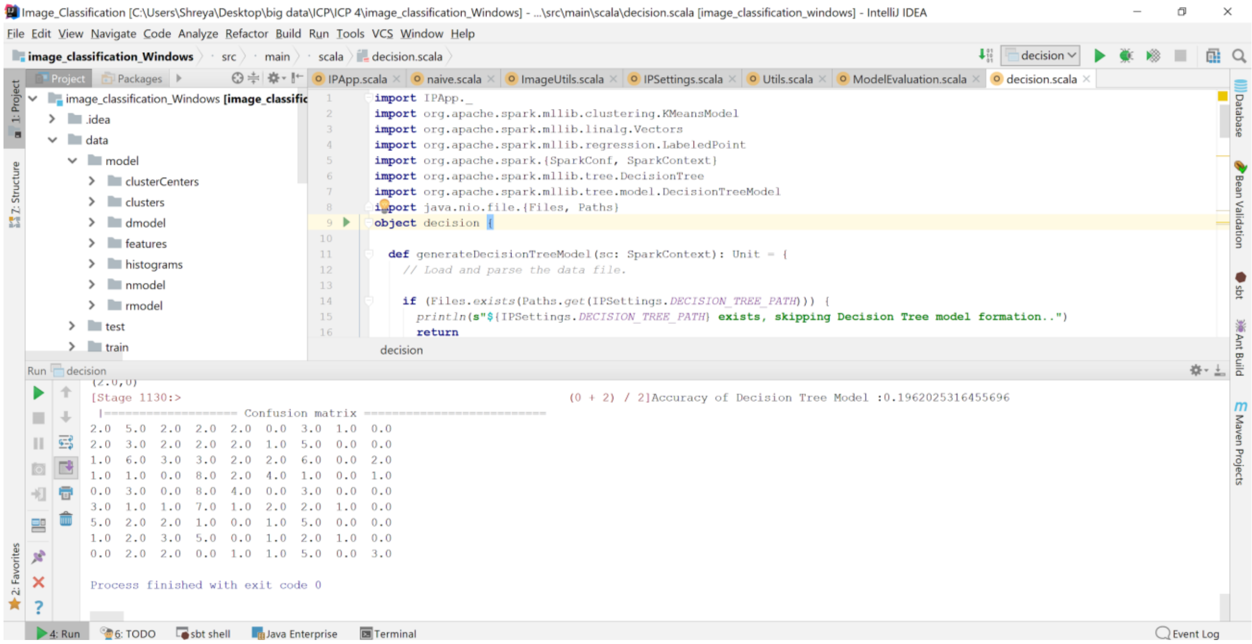
Naive Bayes Model


Image_Classification [C:\Users\Shreya\Desktop\big data\ICP\ICP 4\image_classification_Windows] - ...\src\main\scala\naive.scala [image_classification_windows] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
image_classification_Windows src main scala naive.scala
naive
Project Packages
IPApp.scala naive.scala ImageUtils.scala IPSettings.scala Utils.scala ModelEvaluation.scala decision.scala
image_classification_Windows [image_classific
.idea
data
model
clusterCenters
clusters
features
histograms
nmodel
rmodel
test
train
project [image_classification_windows-bu
src
Run naive
4: Run 6: TODO sbt shell Java Enterprise Terminal Event Log
1: Project 7: Structure 2: Favorites Database Bean Validation sbt Ant Build Maven Projects

```
import IPApp._
import java.nio.file.{Files, Paths}
import org.apache.spark.mllib.classification.{NaiveBayes, NaiveBayesModel}
import org.apache.spark.mllib.clustering.KMeansModel
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.{SparkConf, SparkContext}

object naive {

  def generateNaiveBayesModel(sc: SparkContext): Unit = {
    if (Files.exists(Paths.get(IPSettings.NAIVE_BAYES_PATH))) {
      println(s"${IPSettings.NAIVE_BAYES_PATH} exists, skipping Naive Bayes model formation..")
      return
    }

    val data = sc.textFile(IPSettings.HISTOGRAM_PATH)
```

```
[Stage 961:>                                                        (0 + 2) / 2]Accuracy of Naive Bayes Model :0.11392405063291139
 |=================== Confusion matrix ==========================
0.0  0.0  0.0  0.0  0.0  17.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  17.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  25.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  18.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  18.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  18.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  16.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  15.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  14.0  0.0  0.0  0.0

Process finished with exit code 0
```

Decision Tree Model



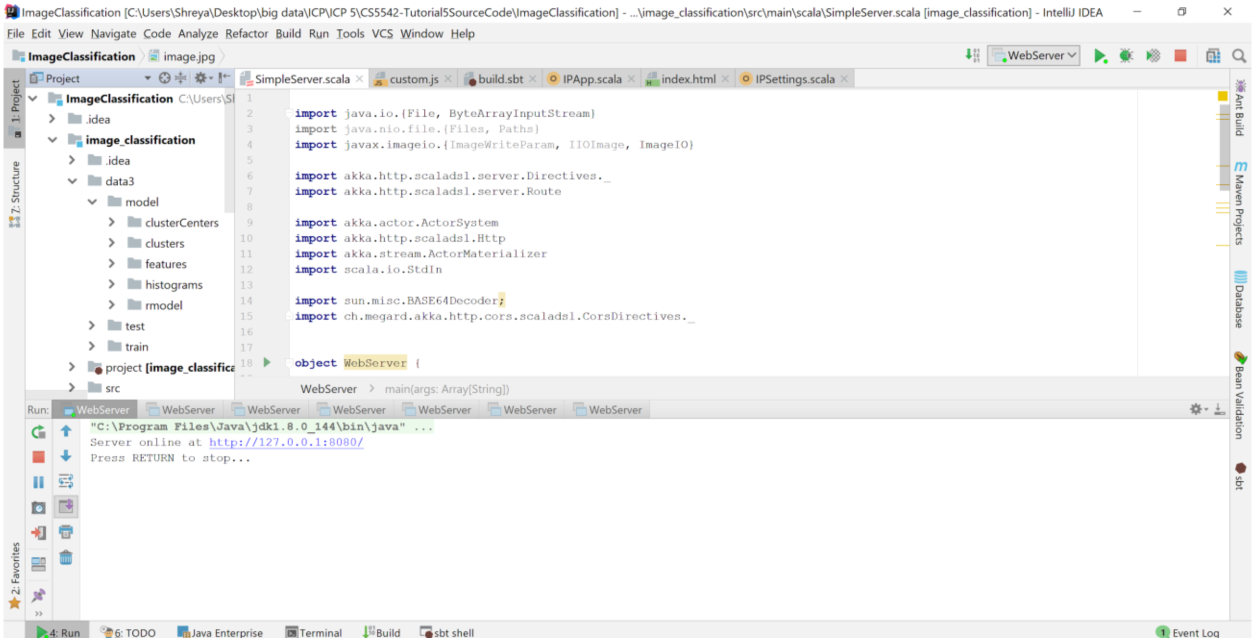
```scala
import IPApp._
import org.apache.spark.mllib.clustering.KMeansModel
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.mllib.tree.DecisionTree
import org.apache.spark.mllib.tree.model.DecisionTreeModel
import java.nio.file.{Files, Paths}
object decision {

  def generateDecisionTreeModel(sc: SparkContext): Unit = {
    // Load and parse the data file.

    if (Files.exists(Paths.get(IPSettings.DECISION_TREE_PATH))) {
      println(s"${IPSettings.DECISION_TREE_PATH} exists, skipping Decision Tree model formation..")
      return
```

```
(2.0,0)
[Stage 1130:>                                                    (0 + 2) / 2]Accuracy of Decision Tree Model :0.1962025316455696
 |================== Confusion matrix ==========================
2.0  5.0  2.0  2.0  2.0  0.0  3.0  1.0  0.0
2.0  3.0  2.0  2.0  2.0  1.0  5.0  0.0  0.0
1.0  6.0  3.0  3.0  2.0  2.0  6.0  0.0  2.0
1.0  1.0  0.0  8.0  2.0  4.0  1.0  0.0  1.0
0.0  3.0  0.0  8.0  4.0  0.0  3.0  0.0  0.0
3.0  1.0  1.0  7.0  1.0  2.0  2.0  1.0  0.0
5.0  2.0  2.0  1.0  0.0  1.0  5.0  0.0  0.0
1.0  2.0  3.0  5.0  0.0  1.0  2.0  1.0  0.0
0.0  2.0  2.0  0.0  1.0  1.0  5.0  0.0  3.0

Process finished with exit code 0
```

IMPLEMENTATION OF IMAGE PREDICTION

Here, image is predicted based on the model generated previously.

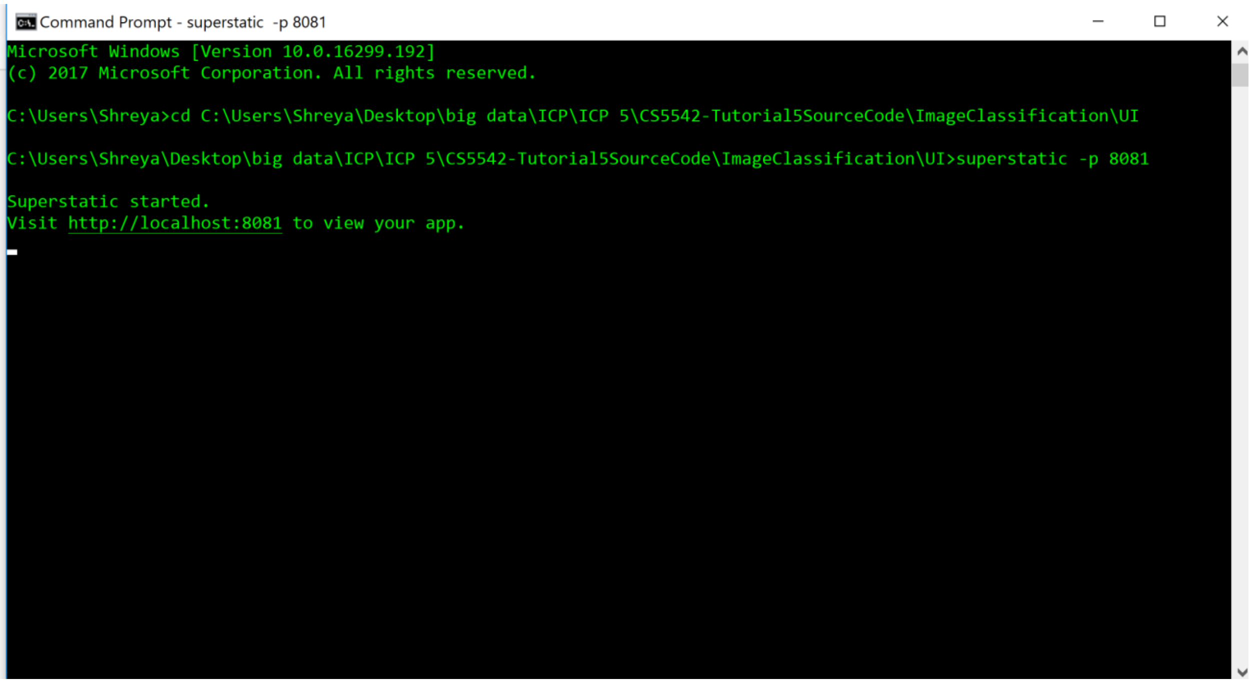
Server Launched



```scala

import java.io.{File, ByteArrayInputStream}
import java.nio.file.{Files, Paths}
import javax.imageio.{ImageWriteParam, IIOImage, ImageIO}

import akka.http.scaladsl.server.Directives._
import akka.http.scaladsl.server.Route

import akka.actor.ActorSystem
import akka.http.scaladsl.Http
import akka.stream.ActorMaterializer
import scala.io.StdIn

import sun.misc.BASE64Decoder;
import ch.megard.akka.http.cors.scaladsl.CorsDirectives._


object WebServer {
```

```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Server online at http://127.0.0.1:8080/
Press RETURN to stop...
```

Superstatic command run on Command Prompt



```
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Shreya>cd C:\Users\Shreya\Desktop\big data\ICP\ICP 5\CS5542-Tutorial5SourceCode\ImageClassification\UI

C:\Users\Shreya\Desktop\big data\ICP\ICP 5\CS5542-Tutorial5SourceCode\ImageClassification\UI>superstatic -p 8081

Superstatic started.
Visit http://localhost:8081 to view your app.
```

Prediction Test

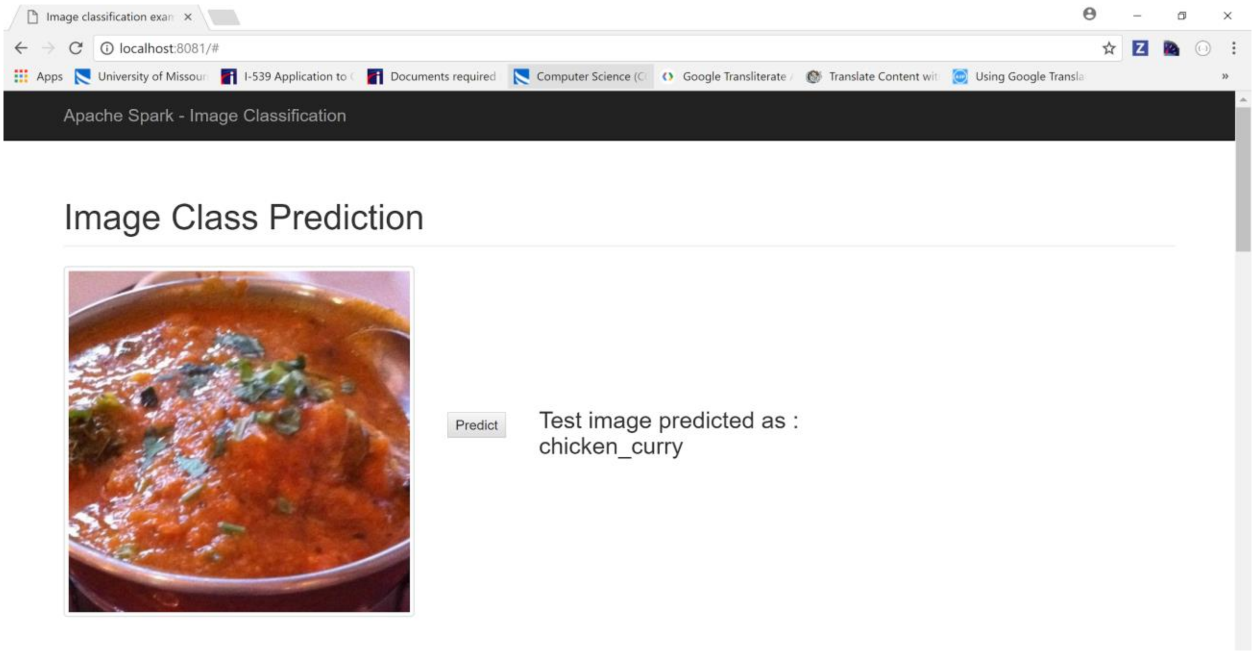
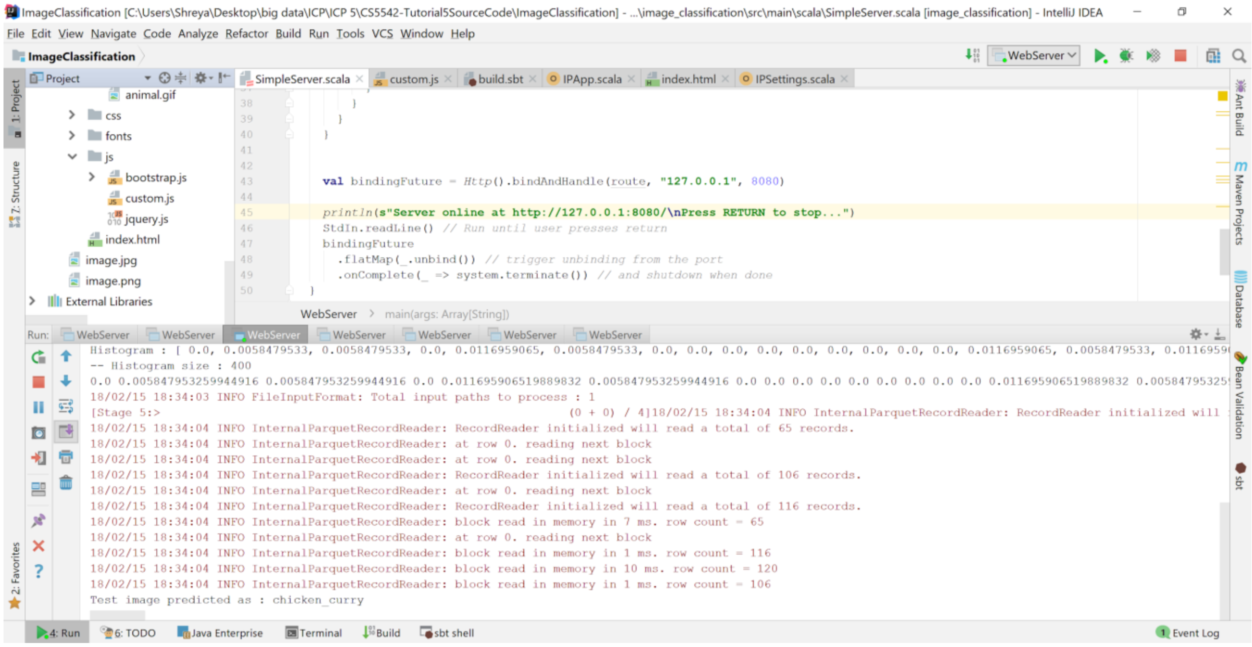


Image Class Prediction

Test image predicted as :
chicken_curry

Prediction Displayed on the server side



```
          }
        }
      }


    val bindingFuture = Http().bindAndHandle(route, "127.0.0.1", 8080)

    println(s"Server online at http://127.0.0.1:8080/\nPress RETURN to stop...")
    StdIn.readLine() // Run until user presses return
    bindingFuture
      .flatMap(_.unbind()) // trigger unbinding from the port
      .onComplete(_ => system.terminate()) // and shutdown when done
  }
```

```
Histogram : [ 0.0, 0.0058479533, 0.0058479533, 0.0, 0.0116959065, 0.0058479533, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0116959065, 0.0058479533, 0.0116959
-- Histogram size : 400
0.0 0.005847953259944916 0.005847953259944916 0.0 0.011695906519889832 0.005847953259944916 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.011695906519889832 0.00584795325
18/02/15 18:34:03 INFO FileInputFormat: Total input paths to process : 1
[Stage 5:>                                                          (0 + 0) / 4]18/02/15 18:34:04 INFO InternalParquetRecordReader: RecordReader initialized will
18/02/15 18:34:04 INFO InternalParquetRecordReader: RecordReader initialized will read a total of 65 records.
18/02/15 18:34:04 INFO InternalParquetRecordReader: at row 0. reading next block
18/02/15 18:34:04 INFO InternalParquetRecordReader: at row 0. reading next block
18/02/15 18:34:04 INFO InternalParquetRecordReader: RecordReader initialized will read a total of 106 records.
18/02/15 18:34:04 INFO InternalParquetRecordReader: at row 0. reading next block
18/02/15 18:34:04 INFO InternalParquetRecordReader: RecordReader initialized will read a total of 116 records.
18/02/15 18:34:04 INFO InternalParquetRecordReader: block read in memory in 7 ms. row count = 65
18/02/15 18:34:04 INFO InternalParquetRecordReader: at row 0. reading next block
18/02/15 18:34:04 INFO InternalParquetRecordReader: block read in memory in 1 ms. row count = 116
18/02/15 18:34:04 INFO InternalParquetRecordReader: block read in memory in 10 ms. row count = 120
18/02/15 18:34:04 INFO InternalParquetRecordReader: block read in memory in 1 ms. row count = 106
Test image predicted as : chicken_curry
```

Few more prediction examples

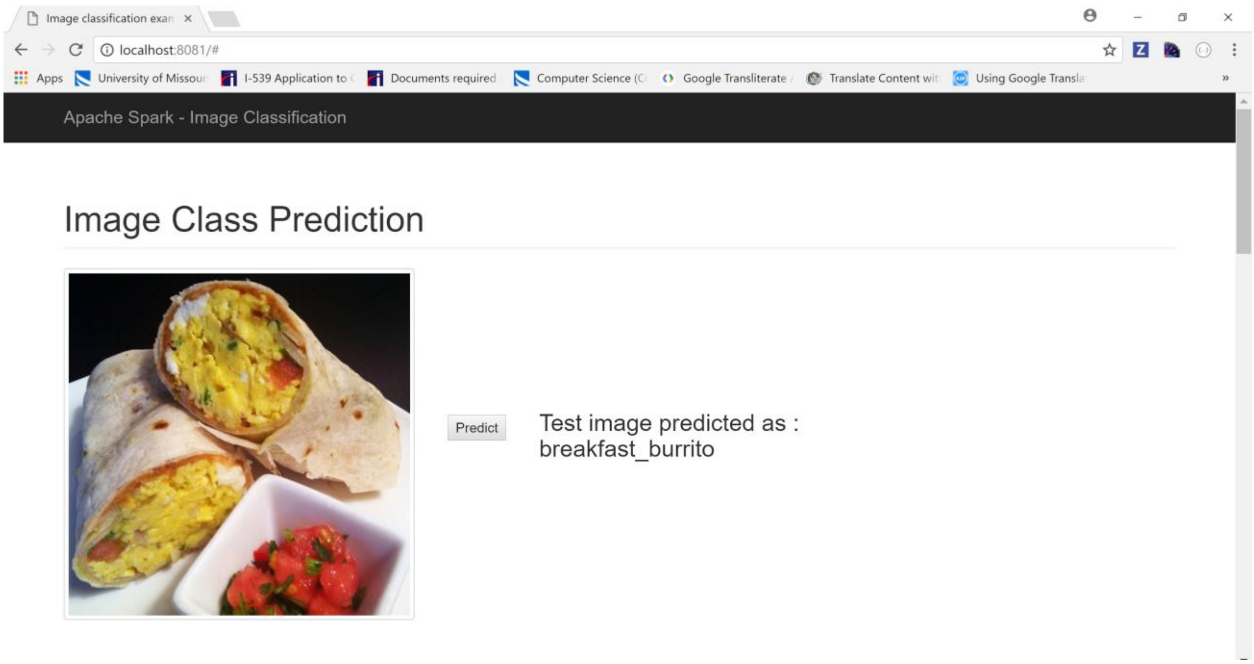
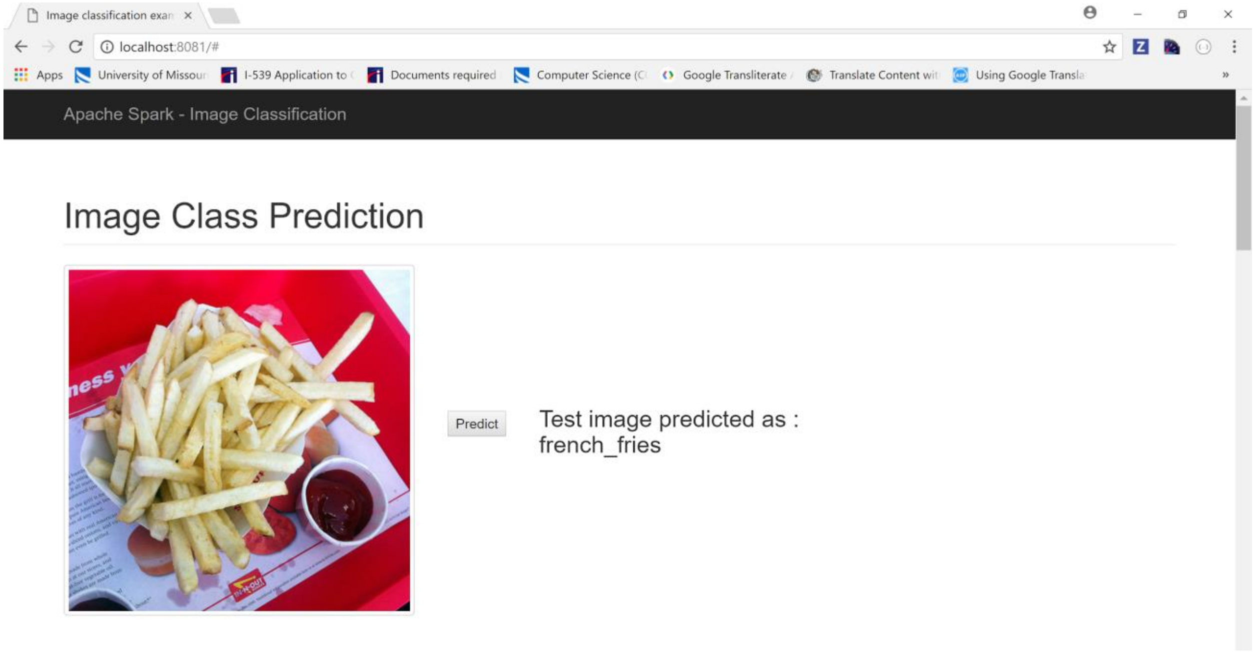


Image Class Prediction

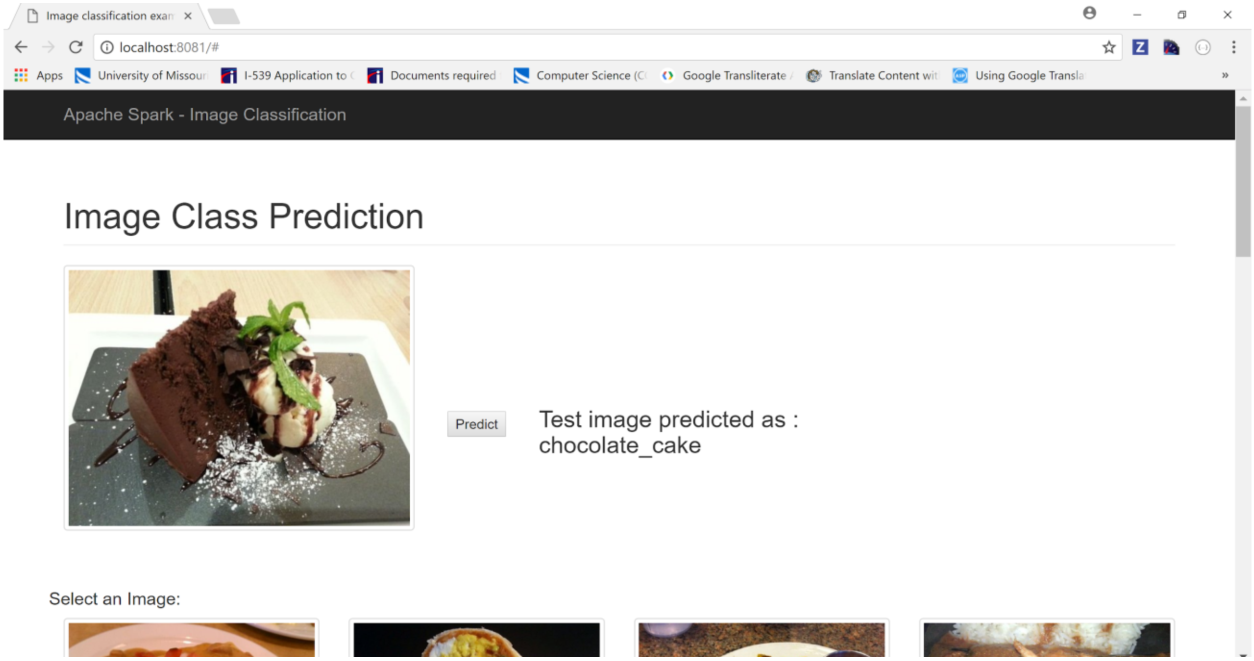
Test image predicted as :
breakfast_burrito



Apache Spark - Image Classification

Image Class Prediction

Test image predicted as :
french_fries



Apache Spark - Image Classification

Image Class Prediction

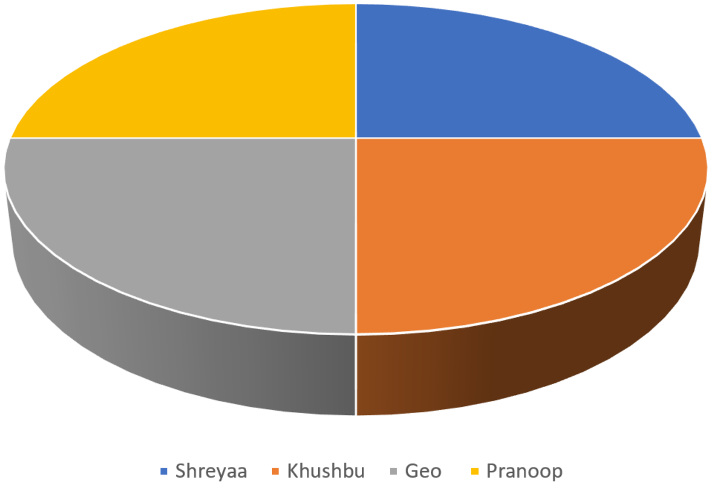
Test image predicted as :
chocolate_cake

Select an Image:

PROJECT MANAGEMENT

Plan & Project Timelines, Members, Task Responsibility

Shreyaa Sridhar (21) - 25%
Khushbu Kolhe (9) - 25%
Geovanni Nicque West (23) - 25%
Naga Venkata Satya Pranoop Mutha (15) - 25%



BIBLIOGRAPHY

http://oar.a-star.edu.sg/jspui/bitstream/123456789/2299/1/05-NTCIR13-LIFELOG-XuQ.pdf
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5537777/
https://arxiv.org/ftp/arxiv/papers/1606/1606.05675.pdf