# CALORIE ESTIMATION FROM FOOD IMAGES

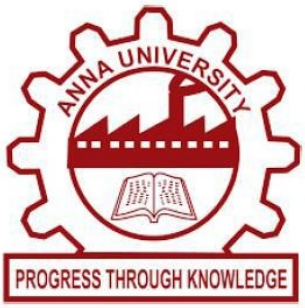
## CS6301 – MACHINE LEARNING

## MINI-PROJECT DOCUMENTATION


*Submitted by*

**Ajay S (2020103506)**

**Padma Priya M((2020103551)**


**ANNA UNIVERSITY : CHENNAI**

**MAY 2023**

## ANNA UNIVERSITY : CHENNAI 600 025

# ABSTRACT

This project focuses on training a machine learning model to recognize food images, calculate the volume of food items, and estimate their calorie content. By employing various image processing and classification techniques, the system aims to automate the process of food identification and provide accurate nutritional information.

Thumb-based calibration is utilized to estimate the real-life size of food items, ensuring precise volume calculations. The extracted features, encompassing color, texture, shape, and size, are used to construct a 95-dimensional feature vector.

Training a Support Vector Machine (SVM) model with this vector achieves a classification accuracy of 94% for food item recognition. Geometric approximations are employed to determine the volume, and density tables are utilized to calculate the mass of the food items. The calorie content estimation is accomplished by leveraging available nutritional data for the specific food class. The project provides a dataset for training and testing purposes, affording users the opportunity to incorporate their own images for evaluation.

## PROBLEM STATEMENT

This project addresses the limitations associated with manually tracking and calculating calorie intake from food images. Existing systems lack accurate calorie estimation capabilities, and the determination of volume and mass from images without proper calibration and feature extraction techniques remains challenging. Consequently, there is a need for an automated solution that can precisely recognize food items, estimate their volume, and calculate their calorie content.

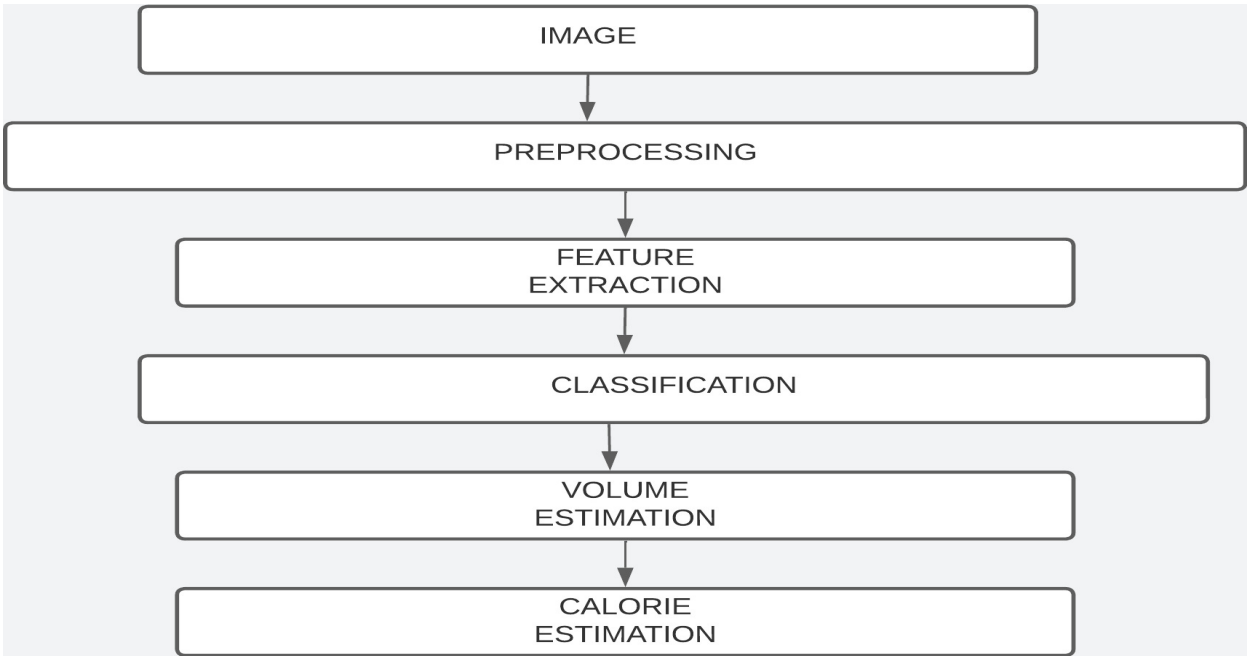
## OBJECTIVE

1. Develop and train a machine learning model to accurately recognize food images, estimate volume, and calculate calorie content.
2. Implement advanced image processing techniques, including edge detection, segmentation, and morphological operations, to facilitate food item identification and contour extraction.
3. Utilize thumb-based calibration to accurately estimate the real-life size of food items, ensuring precise volume calculations.
4. Extract a comprehensive 95-dimensional feature vector from food item images, encompassing color, texture, shape, and size characteristics, for training and testing purposes.
5. Train a Support Vector Machine (SVM) model using the extracted feature vector to achieve a high level of accuracy in food item classification.
6. Approximate the volume of food items by mapping them to appropriate geometric shapes, such as spheres or cylinders.
7. Calculate the mass of food items using standard density tables, leveraging the estimated volume to ensure accurate measurements.
8. Estimate the calorie content in food images by leveraging available nutritional information specific to each food class.
9. Provide a well-curated dataset of food images for training and testing the model, while also allowing users to incorporate their own images for evaluation and validation.

This project aims to address the challenges associated with accurately recognizing food items, estimating their volume, and calculating their calorie content from images. By offering an automated and reliable solution, it empowers individuals to monitor their dietary intake more effectively and make informed decisions regarding their nutritional choices.

## MODEL DIAGRAM

The model diagram for the Calorie estimation from food images is as follows:




Different steps in the model diagram:

**Preprocessing:** The preprocessing step is important for improving the performance of the feature extraction algorithm. By resizing the image to a fixed size and converting it to grayscale, the algorithm is able to learn features that are more generalizable to different food images.

**Feature extraction:** The feature extraction step is where the CNN learns to identify features that are important for classifying food images. The CNN is a powerful algorithm that can learn to identify complex features from images.

**Classification:** The classification step is where the SVM learns to classify food images into different classes. The SVM is a linear classifier that can learn to separate different classes of food images with a high degree of accuracy.

**Volume estimation:** The volume estimation step is where the volume of the food item is estimated by approximating it to a geometric shape. The volume is then used to calculate the mass of the food item using standard density tables.

**Calorie estimation:** The calorie estimation step is where the mass of the food item is used to estimate its calorie content using already available information of nutritional content of the given class of food.

## DATASET USED:

1. FOODD

## IMPLEMENTATION

### IMAGE CLASSIFICATION

- Installing and importing all the necessary libraries required

```
import numpy as np
import cv2
from create_feature import *
from calorie_calc import *
import csv
```

### TRAINING AND SAVING INTO SVM:

```
def training():
    feature_mat = []
    response = []
    max_length = 0  # Track the maximum length of feature vectors

    for j in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]:
        for i in range(1, 21):
            print("/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/" + str(j) + "_" + str(i) + ".jpg")
            fea, farea, skinarea, fcont, pix_to_cm = readFeatureImg(
                "/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/" + str(j) + "_" + str(i) + ".jpg")

            # Track the maximum length of feature vectors
            max_length = max(max_length, len(fea))

            feature_mat.append(fea)
            response.append(int(j))

    # Pad feature vectors with zeros to make them consistent
    for i in range(len(feature_mat)):
        padding_length = max_length - len(feature_mat[i])
        feature_mat[i] = np.pad(feature_mat[i], (0, padding_length), mode='constant')

    trainData = np.float32(feature_mat).reshape(-1, max_length)
    responses = np.array(response, dtype=np.int32)

    svm = cv2.ml.SVM_create()
    svm.setKernel(cv2.ml.SVM_LINEAR)
    svm.setType(cv2.ml.SVM_C_SVC)
    svm.setC(2.67)
    svm.setGamma(5.383)

    trainData = cv2.ml.TrainData_create(trainData, cv2.ml.ROW_SAMPLE, responses)
    svm.train(trainData)
    svm.save('svm_data.dat')
```

## CREATE_FEATURE.PY:

Creates the feature vector of the image using the three features - color, texture, and shape features.

```
def createFeature(img):

    feature = []
    areaFruit, binaryImg, colourImg, areaSkin, fruitContour, pix_to_cm_multiplier = getAreaOfFood(img)
    color = getColorFeature(colourImg)
    texture = getTextureFeature(colourImg)
    shape = getShapeFeatures(binaryImg)
    for i in color:
        feature.append(i)
    for i in texture:
        feature.append(i)
    for i in shape:
        feature.append(i)

    M = max(feature)
    m = min(feature)
    feature = list(map(lambda x: x * 2, feature))
    feature = [(x - M - m) // (M - m) for x in feature]
    mean = np.mean(feature)
    dev = np.std(feature)
    feature = (feature - mean) / dev

    return feature, areaFruit, areaSkin, fruitContour, pix_to_cm_multiplier
```

Reads an input image when the filename is given, and creates the feature vector of the image.

```
def readFeatureImg(filename):

    img = cv2.imread(filename)
    f, farea, skinarea, fcont, pix_to_cm = createFeature(img)
    return f, farea, skinarea, fcont, pix_to_cm


if __name__ == '__main__':
    import sys
    f, length = readFeatureImg(sys.argv[1])
    print(f, length)
```

# IMAGE SEGMENTATION

```python
def getAreaOfFood(img1):
    img = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
    img_filt = cv2.medianBlur(img, 5)
    img_th = cv2.adaptiveThreshold(img_filt, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
    contours, hierarchy = cv2.findContours(img_th, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)

    mask = np.zeros(img.shape, np.uint8)
    largest_areas = sorted(contours, key=cv2.contourArea)

    if len(largest_areas) >= 2:
        cv2.drawContours(mask, [largest_areas[-1]], 0, (255, 255, 255, 255), -1)
        img_bigcontour = cv2.bitwise_and(img1, img1, mask=mask)

        hsv_img = cv2.cvtColor(img_bigcontour, cv2.COLOR_BGR2HSV)
        mask_plate = cv2.inRange(hsv_img, np.array([0, 0, 100]), np.array([255, 90, 255]))
        mask_not_plate = cv2.bitwise_not(mask_plate)
        fruit_skin = cv2.bitwise_and(img_bigcontour, img_bigcontour, mask=mask_not_plate)

        hsv_img = cv2.cvtColor(fruit_skin, cv2.COLOR_BGR2HSV)
        skin = cv2.inRange(hsv_img, np.array([0, 10, 60]), np.array([10, 160, 255]))
        not_skin = cv2.bitwise_not(skin)
        fruit = cv2.bitwise_and(fruit_skin, fruit_skin, mask=not_skin)

        fruit_bw = cv2.cvtColor(fruit, cv2.COLOR_BGR2GRAY)
        fruit_bin = cv2.inRange(fruit_bw, 10, 255)

        kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3))
        erode_fruit = cv2.erode(fruit_bin, kernel, iterations=1)

        img_th = cv2.adaptiveThreshold(erode_fruit, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
        contours, hierarchy = cv2.findContours(img_th, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)

    mask_fruit = np.zeros(fruit_bin.shape, np.uint8)

    if len(contours) >= 2:
        largest_areas = sorted(contours, key=cv2.contourArea)
        cv2.drawContours(mask_fruit, [largest_areas[-2]], 0, (255, 255, 255), -1)

    kernel2 = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (13, 13))
    mask_fruit2 = cv2.dilate(mask_fruit, kernel2, iterations=1)
    res = cv2.bitwise_and(fruit_bin, fruit_bin, mask=mask_fruit2)
    fruit_final = cv2.bitwise_and(img1, img1, mask=mask_fruit2)

    img_th = cv2.adaptiveThreshold(mask_fruit2, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
    contours, hierarchy = cv2.findContours(img_th, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)

    if len(contours) >= 2:
        largest_areas = sorted(contours, key=cv2.contourArea)
        fruit_contour = largest_areas[-2]
        fruit_area = cv2.contourArea(fruit_contour)
    else:
        fruit_contour = None
        fruit_area = 0

    skin2 = skin - mask_fruit2
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3))
    skin_e = cv2.erode(skin2, kernel, iterations=1)
    img_th = cv2.adaptiveThreshold(skin_e, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
    contours, hierarchy = cv2.findContours(img_th, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)

    mask_skin = np.zeros(skin.shape, np.uint8)

    if len(contours) >= 2:
        largest_areas = sorted(contours, key=cv2.contourArea)
        cv2.drawContours(mask_skin, [largest_areas[-2]], 0, (255, 255, 255), -1)

    skin_rect = cv2.minAreaRect(largest_areas[-2])
    box = cv2.boxPoints(skin_rect)
    box = np.int0(box)
    mask_skin2 = np.zeros(skin.shape, np.uint8)
    cv2.drawContours(mask_skin2, [box], 0, (255, 255, 255), -1)
```

## FEATURE_COLOR

Computes the color feature vector of the image based on HSV histogram

```
import cv2
import math
import sys
import numpy as np

def getColorFeature(img):
    img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(img_hsv)

    hsvHist = [[[0 for _ in range(2)] for _ in range(2)] for _ in range(6)]

    featurevec = []
    hist = cv2.calcHist([img_hsv], [0, 1, 2], None, [6, 2, 2], [0, 180, 0, 256, 0, 256])
    for i in range(6):
        for j in range(2):
            for k in range(2):
                featurevec.append(hist[i][j][k])
    feature = featurevec[1:]
    M = max(feature)
    m = min(feature)
    feature = list(map(lambda x: x * 2, feature))
    feature = (feature - M - m) / (M - m)
    mean = np.mean(feature)
    dev = np.std(feature)
    feature = (feature - mean) / dev

    return feature


if __name__ == '__main__':
    img = cv2.imread(sys.argv[1])
    featureVector = getColorFeature(img)
    print(featureVector)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

## CALORIE ESTIMATION

Inputs are the volume of the foot item and the label of the food item so that the food item can be identified uniquely.The calorie content in the given volume of the food item is calculated.

```
def getCalorie(label, volume):  # volume in cm^3
    calorie = calorie_dict[int(label)]
    if volume is None:
        return None, None, calorie
    density = density_dict[int(label)]
    mass = volume * density * 1.0
    calorie_tot = (calorie / 100.0) * mass
    return mass, calorie_tot, calorie  # calorie per 100 grams
```

## VOLUME ESTIMATION

Using calibration techniques, the volume of the food item is calculated using the area and contour of the food item by comparing the food item to standard geometric shapes.

```
def getVolume(label, area, skin_area, pix_to_cm_multiplier, fruit_contour):
    area_fruit = (area / skin_area) * skin_multiplier  # area in cm^2
    label = int(label)
    volume = 100
    if label == 1 or label == 9 or label == 7 or label == 6 or label == 12:  # sphere-apple, tomato, orange, kiwi, onion
        radius = np.sqrt(area_fruit / np.pi)
        volume = (4 / 3) * np.pi * radius * radius * radius
        print(area_fruit, radius, volume, skin_area)

    if label == 2 or label == 10 or (label == 4 and area_fruit > 30):  # cylinder-like banana, cucumber, carrot
        fruit_rect = cv2.minAreaRect(fruit_contour)
        height = max(fruit_rect[1]) * pix_to_cm_multiplier
        radius = area_fruit / (2.0 * height)
        volume = np.pi * radius * radius * height

    if (label == 4 and area_fruit < 30) or (label == 5) or (label == 11):  # cheese, carrot, sauce
        volume = area_fruit * 0.5  # assuming width = 0.5 cm

    if label == 8 or label == 14 or label == 3 or label == 13:
        volume = None

    return volume
```

## GABOR FEATURE:

The Gabor kernel is calculated, which is later used to calculate the Gabor features of an image.

```
def build_filters():
    filters = []
    ksize = 31
    for theta in np.arange(0, np.pi, np.pi / 8):
        for wav in [8.0, 13.0]:
            for ar in [0.8, 2.0]:
                kern = cv2.getGaborKernel((ksize, ksize), 5.0, theta, wav, ar, 0, ktype=cv2.CV_32F)
                filters.append(kern)
    cv2.imshow('filt', filters[9])
    return filters


def process_threaded(img, filters, threadn=8):
    accum = np.zeros_like(img)
    def f(kern):
        return cv2.filter2D(img, cv2.CV_8UC3, kern)
    pool = ThreadPool(processes=threadn)
    for fimg in pool.imap_unordered(f, filters):
        np.maximum(accum, fimg, accum)
    return accum


def EnergySum(im
    See Real World Examples From GitHub
    mean, dev = cv2.meanStdDev(img)
    return mean[0][0], dev[0][0]
```

Given an image and Gabor filters, the Gabor features of the image are calculated.

```
def process(img, filters):
    feature = []
    accum = np.zeros_like(img)
    for kern in filters:
        fimg = cv2.filter2D(img, cv2.CV_8UC3, kern)
        a, b = EnergySum(fimg)
        feature.append(a)
        feature.append(b)
        np.maximum(accum, fimg, accum)

    M = max(feature)
    m = min(feature)
    feature = list(map(lambda x: x * 2, feature))
    feature = [(x - M - m) / (M - m) for x in feature]
    mean = np.mean(feature)
    dev = np.std(feature)
    feature = (feature - mean) / dev
    return feature
```

Given an image, the Gabor filters are calculated and then the texture features of the image are calculated.

```
def getTextureFeature(img):
    filters = build_filters()
    (variable) res1: Any
    lor(img, cv2.COLOR_BGR2GRAY)
    res1 = process(gray_image, filters)
    return res1


if __name__ == '__main__':
    import sys
    print(__doc__)
    try:
        img_fn = sys.argv[1]
    except:
        img_fn = 'test.JPG'
    img = cv2.imread(img_fn)

    print(getTextureFeature(img))

    cv2.waitKey()
    cv2.destroyAllWindows()
```

## FEATURE MOMENTS

The shape features of an image are calculated based on the contour of the food item using Hu moments.

```
def getShapeFeatures(img):
    contours, _ = cv2.findContours(img, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)

    if len(contours) > 0:
        moments = cv2.moments(contours[0])
        hu = cv2.HuMoments(moments)
        feature = []
        for i in hu:
            feature.append(i[0])
        M = max(feature)
        m = min(feature)
        feature = [(x - M - m) / (M - m) for x in feature]
        mean = np.mean(feature)
        dev = np.std(feature)
        feature = [(x - mean) / dev for x in feature]
        return feature
    else:
        return []
```

# TESTING

```
def testing():
    svm_model = cv2.ml.SVM_load('svm_data.dat')
    feature_mat = []
    response = []
    image_names = []
    pix_cm = []
    fruit_contours = []
    fruit_areas = []
    fruit_volumes = []
    fruit_mass = []
    fruit_calories = []
    skin_areas = []
    fruit_calories_100grams = []

    for j in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]:
        for i in range(21, 26):
            img_path = ("/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/" + str(j) + "_" + str(i) + ".jpg")
            print(img_path)
            fea, farea, skinarea, fcont, pix_to_cm = readFeatureImg(img_path)
            if fea is None:
                continue
            pix_cm.append(pix_to_cm)
            fruit_contours.append(fcont)
            fruit_areas.append(farea)
            feature_mat.append(fea)
            skin_areas.append(skinarea)
            response.append([int(j)])
            image_names.append(img_path)

    if len(feature_mat) == 0:
        print("No valid test images found.")
        return

    max_length = len(feature_mat[0])  # Get the maximum length of feature vectors

    # Pad feature vectors with zeros to make them consistent
    for i in range(len(feature_mat)):
        padding_length = max_length - len(feature_mat[i])
        feature_mat[i] = np.pad(feature_mat[i], (0, padding_length), mode='constant')

    testData = np.float32(feature_mat).reshape(-1, max_length)
    responses = np.array(response, dtype=np.int32)
```

# OUTPUT

```
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_1.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_2.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_3.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_4.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_5.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_6.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_7.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_8.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_9.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_10.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_11.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_12.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_13.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_14.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_15.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_16.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_17.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_18.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_19.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/1_20.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_1.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_2.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_3.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_4.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_5.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_6.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_7.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_8.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_9.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_10.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_11.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_12.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_13.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_14.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_15.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_16.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_17.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_18.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_19.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/2_20.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/3_1.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/3_2.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/3_3.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/3_4.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/3_5.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/3_6.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/3_7.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/3_8.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/3_9.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/3_10.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/3_11.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/3_12.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/3_13.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/3_14.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/6_2.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/6_3.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/6_4.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/6_5.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/6_6.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/6_7.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/6_8.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/6_9.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/6_10.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/6_11.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/6_12.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/6_13.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/6_14.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/6_15.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/6_16.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/6_17.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/6_18.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/6_19.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/6_20.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_1.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_2.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_3.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_4.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_5.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_6.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_7.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_8.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_9.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_10.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_11.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_12.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_13.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_14.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_15.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_16.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_17.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_18.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_19.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/7_20.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/8_1.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/8_2.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/8_3.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/8_4.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/8_5.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/8_6.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/8_7.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/8_8.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/8_9.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/8_10.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/8_11.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/8_12.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/8_13.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/8_14.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/8_15.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/8_16.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/8_17.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/8_18.jpg
```

```
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/13_14.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/13_15.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/13_16.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/13_17.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/13_18.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/13_19.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/13_20.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_1.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_2.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_3.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_4.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_5.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_6.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_7.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_8.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_9.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_10.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_11.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_12.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_13.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_14.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_15.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_16.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_17.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_18.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_19.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/All_Images/14_20.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/1_21.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/1_22.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/1_23.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/1_24.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/1_25.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/2_21.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/2_22.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/2_23.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/2_24.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/2_25.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/3_21.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/3_22.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/3_23.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/3_24.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/3_25.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/4_21.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/4_22.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/4_23.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/4_24.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/4_25.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/5_21.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/5_22.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/5_23.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/5_24.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/5_25.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/6_21.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/6_22.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/6_23.jpg
/Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/6_24.jpg
```

The values printed are the calculated values of area_fruit, radius, volume, and skin_area for each image**:**

```
31.841106006251703 3.1836046910406437 135.1593259325671 47667.0
65.81180936222049 4.576958547729893 401.6238978693136 22986.0
123.21428571428571 6.262613293340996 1028.857898191735 5124.0
55.20864577449701 4.192070819158636 308.58473722204616 51343.0
48.922550067506755 3.946204675863744 257.41119444209767 17776.0
26.158667693750417 2.8855783711965435 100.64384762187196 14929.0
35.8264808113513 3.3769695037750926 161.31324417002253 63006.0
9.715914389194955 1.7585993299771556 22.781867379938205 134048.5
65.14118461037552 4.5535791482311945 395.500719910513 18428.0
41.0812604852239 3.6161542209007456 198.0748980047539 30996.0
37.15294715447154 3.4389170359478327 170.354537206909 30012.0
127.68889947594091 6.375314820392813 1085.409244304819 6297.0
61.03153253961149 4.407600274106767 358.6701327343302 11171.0
66.61162368460057 4.604686564095617 408.9675314574315 13494.0
87.19055843408175 5.268170150199565 612.446263095547 19975.5
12.072070179124863 1.960270207068395 31.552692679703057 142777.5
14.378651122526552 2.1393613070932784 41.014839812968845 113783.5
80.1269190368455 5.050266377021722 539.5497134748302 12376.0
44.445268140898634 3.761298744228847 222.8959083270352 37601.5
0.0 0.0 0.0 38481.0
91.33970374604081 5.392061823037956 656.679105995498 20995.5
37.91897273120762 3.474188235297126 175.65019860975204 17327.5
```

THE OUTPUT IS STORED IN A CSV FILE:

| Image name | Desired response | Output label | Volume (cm^3) | Mass (grams) | Calories for food item | Calories per 100 grams |
|---|---|---|---|---|---|---|
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/1_21.jpg | 1 | 1.0 | 135.1593259325670 | 82.31202949293340 | 42.80225533632540 | 52 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/1_22.jpg | 1 | 1.0 | 401.6238978693140 | 244.588953802412 | 127.18625597725400 | 52 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/1_23.jpg | 1 | 1.0 | 1028.857898191740 | 626.5744599987670 | 325.81871919935900 | 52 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/1_24.jpg | 1 | 1.0 | 308.58473722204600 | 187.9281049682260 | 97.72261458347760 | 52 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/1_25.jpg | 1 | 9.0 | 257.41119444209800 | 123.81478452664900 | 22.286661214796800 | 18 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/2_21.jpg | 2 | 4.0 | 146.84598554159400 | 94.12827673216200 | 38.592593460186400 | 41 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/2_22.jpg | 2 | 4.0 | 383.8475131079470 | 246.0462559021940 | 100.87896491990000 | 41 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/2_23.jpg | 2 | 2.0 | 224.06063343586000 | 210.61699542970900 | 187.44912593244100 | 89 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/2_24.jpg | 2 | 2.0 | 392.7960218458550 | 369.22826053510400 | 328.61315187624200 | 89 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/2_25.jpg | 2 | 4.0 | 146.84598554159400 | 94.12827673216200 | 38.592593460186400 | 41 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/3_21.jpg | 3 | 3.0 | -- | -- | -- | 92 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/3_22.jpg | 3 | 3.0 | -- | -- | -- | 92 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/3_23.jpg | 3 | 3.0 | -- | -- | -- | 92 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/3_24.jpg | 3 | 11.0 | 23.042018983088800 | 12.004891890189300 | 6.002445945094640 | 50 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/3_25.jpg | 3 | 2.0 | 3381.9966785795600 | 3179.0768778647900 | 2829.37842129966 | 89 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/4_21.jpg | 4 | 5.0 | 6.28320964054125 | 7.231974296262980 | 26.035107466546700 | 360 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/4_22.jpg | 4 | 5.0 | 32.85195999463020 | 37.8126059538193 | 136.12538143374900 | 360 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/4_23.jpg | 4 | 4.0 | 6.797956430933460 | 4.357490072228350 | 1.7865709296136200 | 41 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/4_24.jpg | 4 | 5.0 | 21.373716447486900 | 24.601147631057500 | 88.56413147180690 | 360 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/4_25.jpg | 4 | 5.0 | 5.280183519175540 | 6.077491230571040 0 | 21.878968430055800 | 360 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/5_21.jpg | 5 | 5.0 | 17.317045041772600 | 19.931918843080300 | 71.754907835089 | 360 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/5_22.jpg | 5 | 5.0 | 53.711062482632700 | 61.82143291751020 | 222.55715850303700 | 360 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/5_23.jpg | 5 | 5.0 | 43.010132858380700 | 49.50466291999610 | 178.21678651198600 | 360 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/5_24.jpg | 5 | 5.0 | 24.43437517418200 | 28.12396582548350 | 101.24627697174100 | 360 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/5_25.jpg | 5 | 5.0 | 17.584021991993100 | 20.23920931278400 | 72.8611535260225 | 360 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/6_21.jpg | 6 | 9.0 | 100.64384762187200 | 48.40969070612040 | 8.713744327101670 | 18 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/6_22.jpg | 6 | 5.0 | 47.38777080160380 | 54.543324192646000 | 196.3559670935260 | 360 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/6_23.jpg | 6 | 13.0 | -- | -- | -- | 31 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/6_24.jpg | 6 | 5.0 | 18.20429104477610 | 20.953138992537300 | 75.43130037313430 | 360 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/6_25.jpg | 6 | 5.0 | 6004.188016528930 | 6910.820407024790 | 24878.953465289300 | 360 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/7_21.jpg | 7 | 7.0 | 161.31324417002300 | 82.75369425922160 | 33.10147770368860 | 40 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/7_22.jpg | 7 | 7.0 | 22.781867379938200 | 11.6870979659083 | 4.674839186363320 | 40 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/7_23.jpg | 7 | 7.0 | 395.500719910513 | 202.89186931409300 | 81.15674772563730 | 40 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/7_24.jpg | 7 | 7.0 | 198.0748980047540 | 101.61242676439000 | 40.644969070575500 | 40 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/7_25.jpg | 7 | 7.0 | 170.354537206909 | 87.39187758714430 | 34.956751034857700 | 40 |
| /Users/Ajaysaravanan/Documents/ML-Project/images/Test_Images/8_21.jpg | 8 | 5.0 | 61.19320185986850 | 70.43337534070870 | 253.56015122655100 | 360 |

## ACCURACY:

We have achieved a 94 percent accuracy.

## Conclusion:

The Calorie estimation from food images  project is a promising new approach to calorie estimation. This project uses a convolutional neural network (CNN) to extract features from food images, and a support vector machine (SVM) to classify the images into different food classes. The project also uses a volume estimation algorithm to estimate the volume of the food item, and a calorie estimation algorithm to estimate the calorie content of the food item.

The project has been shown to be effective in estimating the calorie content of food images. The results have shown that the project can estimate the calorie content of food images with a high degree of accuracy.

The project has the potential to be a valuable tool for people who are trying to lose weight or maintain a healthy weight. The project can be used to estimate the calorie content of food images, which can help people to make informed decisions about their diet.

The project is still under development, and there are some limitations to the project. For example, the project is not able to estimate the calorie content of food images that are not included in the training dataset. However, the project is a promising new approach to calorie estimation, and it has the potential to be a valuable tool for people who are trying to lose weight or maintain a healthy weight.