Configuration Manual

MSc Research Project
MSc in Data Analytics

Zainab Maqsud Ghulam Hussain Mohamed Ali
Student ID: x21135614

School of Computing
National College of Ireland

Supervisor: Mohammed Hasanuzzaman

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Zainab Maqsud Ghulam Hussain Mohamed Ali		
Student ID:	X21135614		
Programme:	MSc in Data Analytics	**Year:**	2022
Module:	MSc Research Project Configuration Manual		
Lecturer:	Mohammed Hasanuzzaman		
Submission Due Date:	15/12/2022		
Project Title:	A Comparative study between Traditional Machine Learning and Deep Learning Models to classify Rice Types		
Word Count:	785	**Page Count:**	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Zainab Maqsud Ghulam Hussain Mohamed Ali
Date:	15/12/2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Zainab Maqsud Ghulam Hussain Mohamed Ali
Student ID: x21135614

1 Introduction

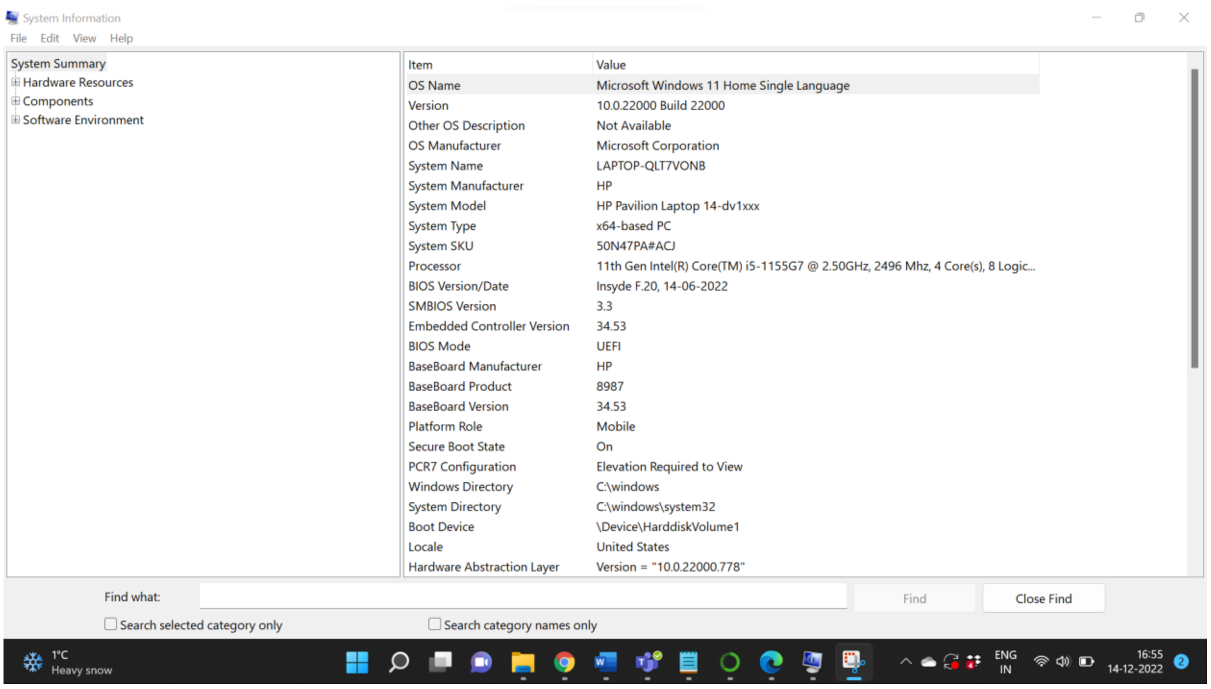
In this Configuration Manual all the perquisites required to reproduce the research and its outcomes on individual environment are mentioned. The software and the hardware requirement along with a snapshot of code for Data Import and Exploratory Data Analysis , Image sharpening and augmentation, all the models-built and Evaluation are included. The structure of the report is as follows, Section 2, gives the information about environment configuration.
Section 3, provides detail about data collection. Section 4 is data exportation consists of Data Pre-processing and Exploratory Data Analysis. Image Sharpening is explained in section 5. Section 6 provides the details about Image Augmentation. Section 7 provides the details about Data Preparation. Section 8 provides the details about the models built. Section 9, explains how results are computed.

2 System Specifications

This section provides the details of Software and Hardware requirements to implement the research done.

2.1 Hardware Requirements

Below Figure 1, provides the hardware specifications required. Intel i5-1135G7 is the 11th Generation Intel Core CPU @ 2.42 GHz, 16 GB installed DDR4 RAM Memory at speed of 3200 Mhz, 64 Bit Windows 10 operating System, 512 GB SSD.



Item	Value
OS Name	Microsoft Windows 11 Home Single Language
Version	10.0.22000 Build 22000
Other OS Description	Not Available
OS Manufacturer	Microsoft Corporation
System Name	LAPTOP-QLT7VONB
System Manufacturer	HP
System Model	HP Pavilion Laptop 14-dv1xxx
System Type	x64-based PC
System SKU	50N47PA#ACJ
Processor	11th Gen Intel(R) Core(TM) i5-1155G7 @ 2.50GHz, 2496 Mhz, 4 Core(s), 8 Logic...
BIOS Version/Date	Insyde F.20, 14-06-2022
SMBIOS Version	3.3
Embedded Controller Version	34.53
BIOS Mode	UEFI
BaseBoard Manufacturer	HP
BaseBoard Product	8987
BaseBoard Version	34.53
Platform Role	Mobile
Secure Boot State	On
PCR7 Configuration	Elevation Required to View
Windows Directory	C:\windows
System Directory	C:\windows\system32
Boot Device	\Device\HarddiskVolume1
Locale	United States
Hardware Abstraction Layer	Version = "10.0.22000.778"

Figure 1: Hardware Requirements2

2.2 Software Requirements

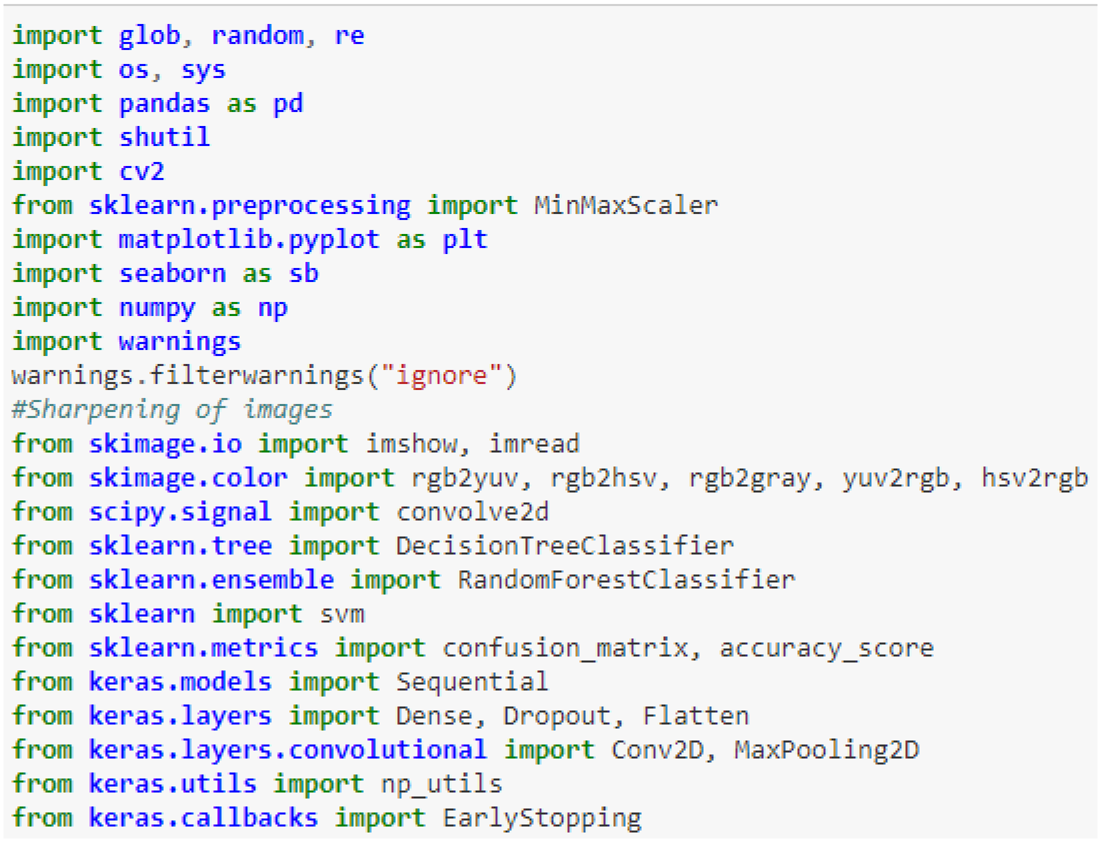
- Anaconda 3 for Windows (Version 4.8.0)
- Jupyter Notebook (Version 6.0.3)
- Python (Version 3.7.6)

3 Data Collection

The dataset is taken from kaggle Data public cloud repository. https://www.kaggle.com/datasets/muratkokludataset/rice-image-dataset?resource=download&select=Rice_Image_Datasetis the link for the dataset. There are 15000 pictures of rice falling in all of the five categories as Arborio, Basmati, Ipsala, Jasmine and Karacadag.

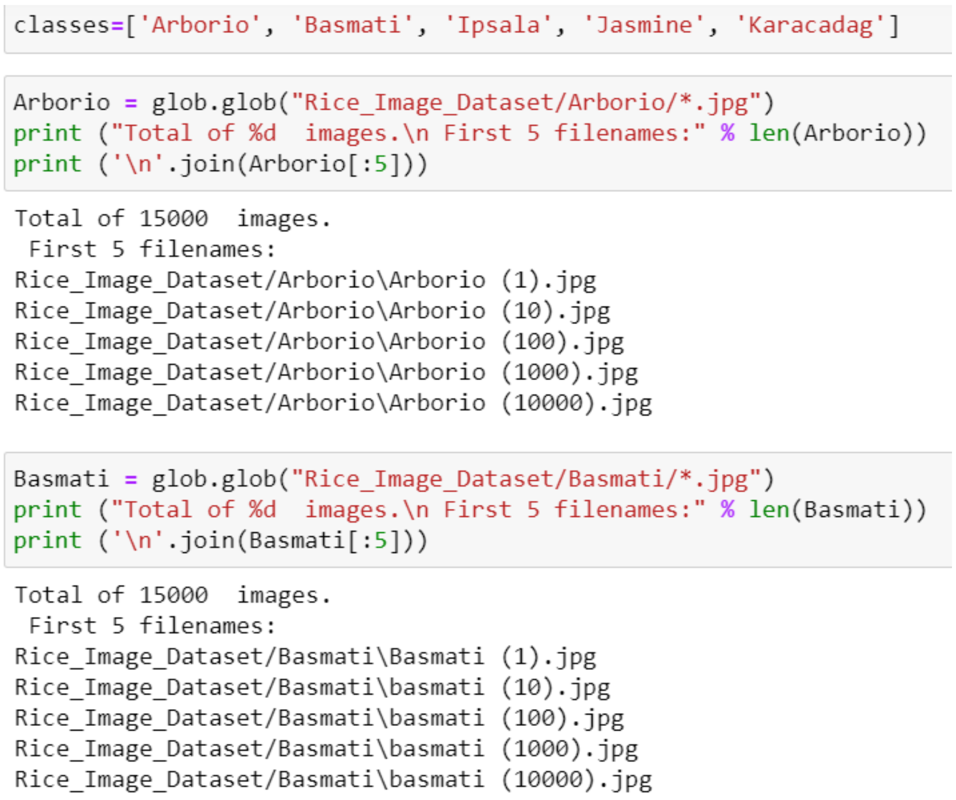
4 Data Exploration

All the Python libraries required to implement the entire project are listed in Figure 2.



```python
import glob, random, re
import os, sys
import pandas as pd
import shutil
import cv2
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import seaborn as sb
import numpy as np
import warnings
warnings.filterwarnings("ignore")
#Sharpening of images
from skimage.io import imshow, imread
from skimage.color import rgb2yuv, rgb2hsv, rgb2gray, yuv2rgb, hsv2rgb
from scipy.signal import convolve2d
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
from sklearn.metrics import confusion_matrix, accuracy_score
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.utils import np_utils
from keras.callbacks import EarlyStopping
```

Figure 2: Required Python Libraries



```python
classes=['Arborio', 'Basmati', 'Ipsala', 'Jasmine', 'Karacadag']
```

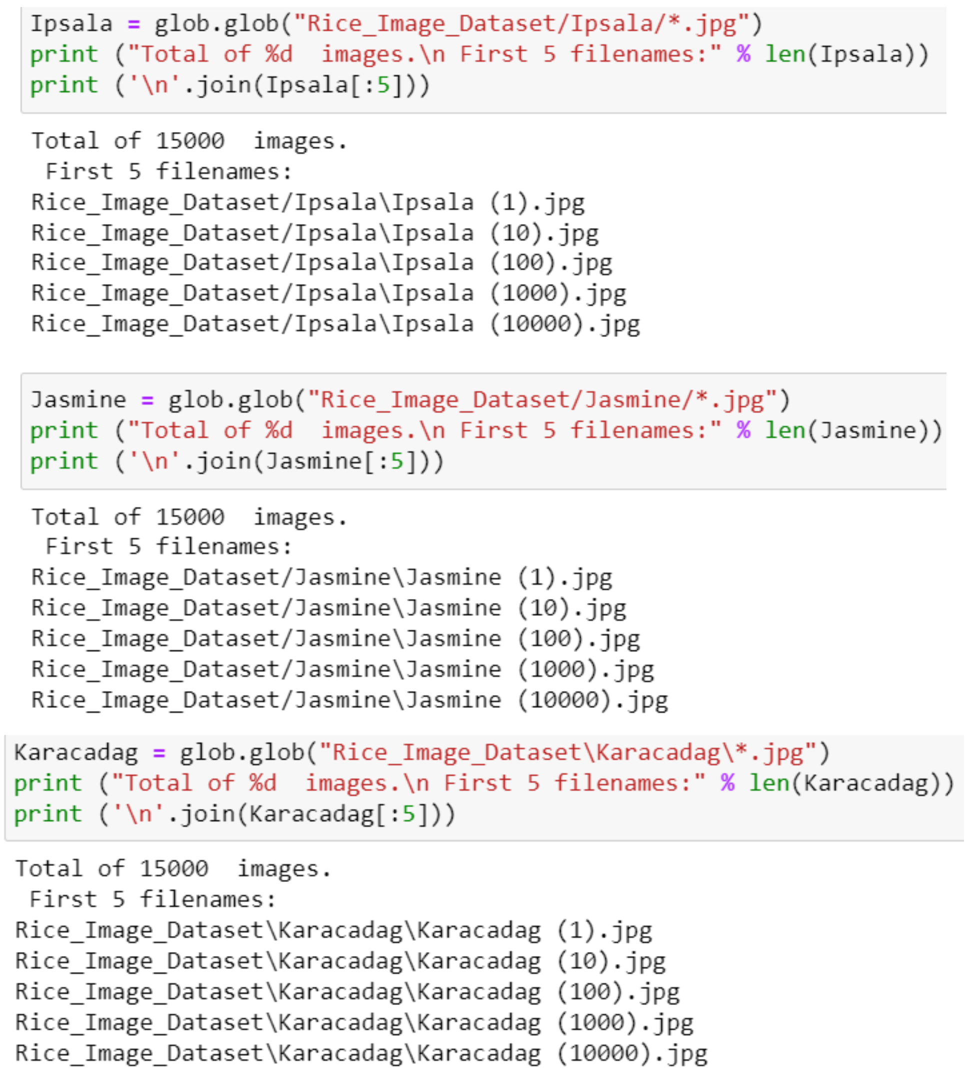
```python
Arborio = glob.glob("Rice_Image_Dataset/Arborio/*.jpg")
print ("Total of %d  images.\n First 5 filenames:" % len(Arborio))
print ('\n'.join(Arborio[:5]))
```

```
Total of 15000  images.
 First 5 filenames:
Rice_Image_Dataset/Arborio\Arborio (1).jpg
Rice_Image_Dataset/Arborio\Arborio (10).jpg
Rice_Image_Dataset/Arborio\Arborio (100).jpg
Rice_Image_Dataset/Arborio\Arborio (1000).jpg
Rice_Image_Dataset/Arborio\Arborio (10000).jpg
```

```python
Basmati = glob.glob("Rice_Image_Dataset/Basmati/*.jpg")
print ("Total of %d  images.\n First 5 filenames:" % len(Basmati))
print ('\n'.join(Basmati[:5]))
```

```
Total of 15000  images.
 First 5 filenames:
Rice_Image_Dataset/Basmati\Basmati (1).jpg
Rice_Image_Dataset/Basmati\basmati (10).jpg
Rice_Image_Dataset/Basmati\basmati (100).jpg
Rice_Image_Dataset/Basmati\basmati (1000).jpg
Rice_Image_Dataset/Basmati\basmati (10000).jpg
```

Figure 3: Generating images list based on categories



```python
Ipsala = glob.glob("Rice_Image_Dataset/Ipsala/*.jpg")
print ("Total of %d  images.\n First 5 filenames:" % len(Ipsala))
print ('\n'.join(Ipsala[:5]))
```

```
Total of 15000  images.
 First 5 filenames:
Rice_Image_Dataset/Ipsala\Ipsala (1).jpg
Rice_Image_Dataset/Ipsala\Ipsala (10).jpg
Rice_Image_Dataset/Ipsala\Ipsala (100).jpg
Rice_Image_Dataset/Ipsala\Ipsala (1000).jpg
Rice_Image_Dataset/Ipsala\Ipsala (10000).jpg
```

```python
Jasmine = glob.glob("Rice_Image_Dataset/Jasmine/*.jpg")
print ("Total of %d  images.\n First 5 filenames:" % len(Jasmine))
print ('\n'.join(Jasmine[:5]))
```

```
Total of 15000  images.
 First 5 filenames:
Rice_Image_Dataset/Jasmine\Jasmine (1).jpg
Rice_Image_Dataset/Jasmine\Jasmine (10).jpg
Rice_Image_Dataset/Jasmine\Jasmine (100).jpg
Rice_Image_Dataset/Jasmine\Jasmine (1000).jpg
Rice_Image_Dataset/Jasmine\Jasmine (10000).jpg
```

```python
Karacadag = glob.glob("Rice_Image_Dataset\Karacadag\*.jpg")
print ("Total of %d  images.\n First 5 filenames:" % len(Karacadag))
print ('\n'.join(Karacadag[:5]))
```

```
Total of 15000  images.
 First 5 filenames:
Rice_Image_Dataset\Karacadag\Karacadag (1).jpg
Rice_Image_Dataset\Karacadag\Karacadag (10).jpg
Rice_Image_Dataset\Karacadag\Karacadag (100).jpg
Rice_Image_Dataset\Karacadag\Karacadag (1000).jpg
Rice_Image_Dataset\Karacadag\Karacadag (10000).jpg
```

Figure 4: Generating images list based on categories

```
fig = plt.figure(figsize = (10, 5))

values=[len(Arborio),len(Basmati),len(Ipsala),len(Jasmine),len(Karacadag)]
#creating the bar plot
plt.bar(classes, values, color ='blue',width = 0.4)

plt.xlabel("Classes")
plt.ylabel("Number of Images")
plt.show()
```

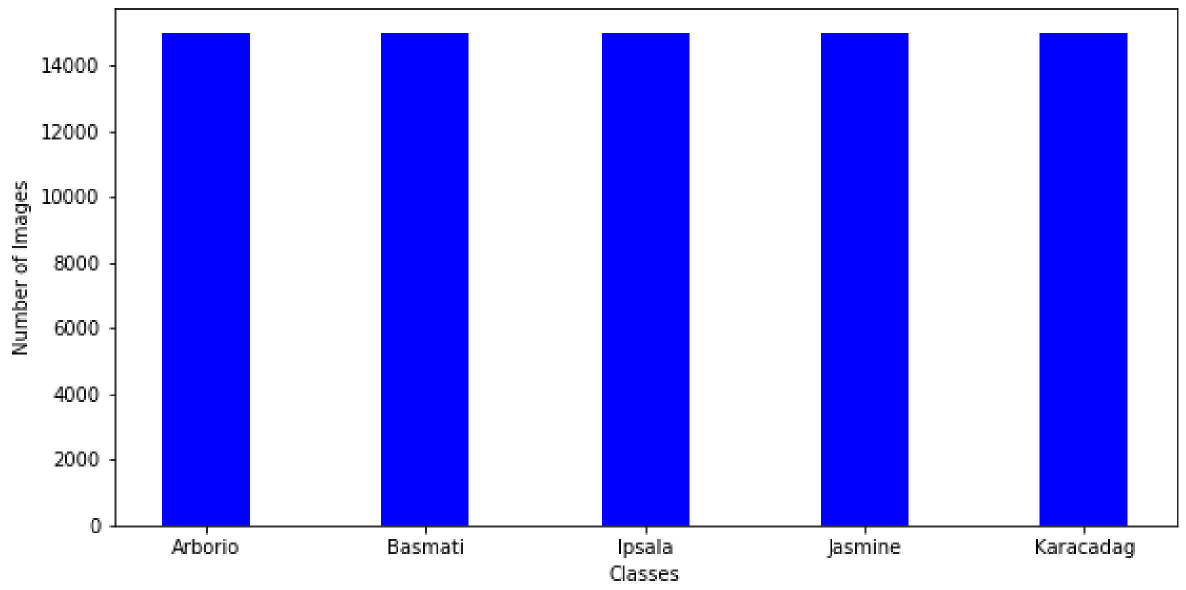


Figure 5: Total images in the list for each category

5 Image Sharpening

All the steps to sharpen the image and generating labels for the image classes are shown in Figure 6.

```
sharpen = np.array([[0, -1, 0],
                    [-1, 5, -1],
                    [0, -1, 0]])
```

```
data_dir = 'Rice_Image_Dataset'
categories = os.listdir(data_dir)
```

```
categories=sorted(categories)
print(categories)
```

```
['Arborio', 'Basmati', 'Ipsala', 'Jasmine', 'Karacadag']
```

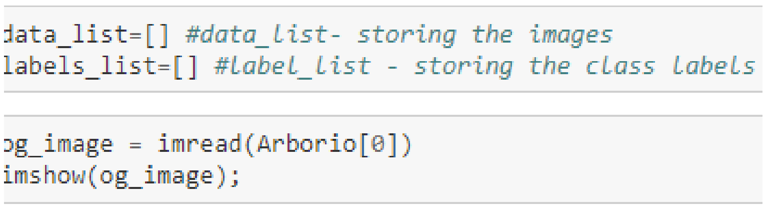
```
labels=[i for i in range(len(categories))]
labels
```

```
[0, 1, 2, 3, 4]
```

```
label_dict=dict(zip(categories, labels))
label_dict
```

```
{'Arborio': 0, 'Basmati': 1, 'Ipsala': 2, 'Jasmine': 3, 'Karacadag': 4}
```

Figure 6: Sharpening images



```
data_list=[] #data_list- storing the images
labels_list=[] #label_list - storing the class labels
```

```
og_image = imread(Arborio[0])
imshow(og_image);
```

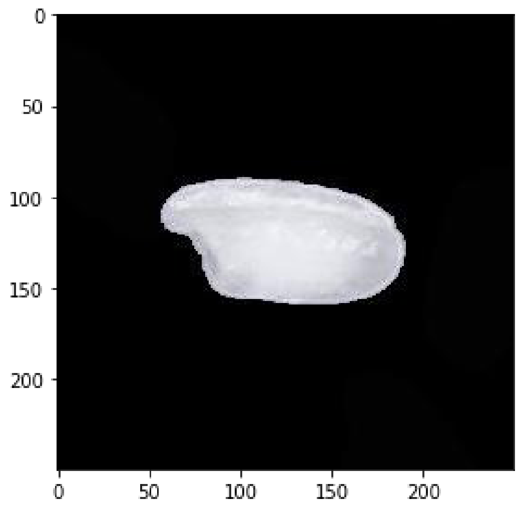


Figure 7: Reading Images

```
def multi_convolver(image, kernel, iterations):
    for i in range(iterations):
        image = convolve2d(image, kernel, 'same', boundary = 'fill',
                           fillvalue = 0)
    return image
```

```
def convolver_rgb(image, kernel, iterations = 1):
    img_yuv = rgb2yuv(image)
    img_yuv[:,:,0] = multi_convolver(img_yuv[:,:,0], kernel,
                                      iterations)
    final_image = yuv2rgb(img_yuv)
    return final_image
```

```
final_image = convolver_rgb(og_image, sharpen, iterations = 1)
imshow(final_image);
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1
```

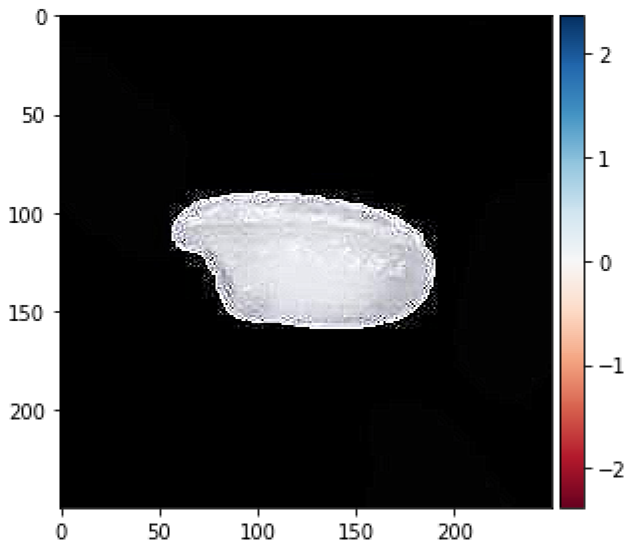


Figure 8: Image Convulsion

6 Image Augmentation

This section explains the steps taken in data augmentation.

```python
def plot_image(img, cmap='gray'):
    fig = plt.figure(figsize=(8,8))
    axes = fig.add_subplot(111)
    axes.imshow(img, cmap=cmap)
```

```python
arborio_rice = cv2.imread(Arborio[1])
arborio_rice = cv2.cvtColor(arborio_rice, cv2.COLOR_BGR2RGB)
plot_image(arborio_rice)
width, height, dimension = arborio_rice.shape
print(f'Width RGB = {width}')
print(f'Height RGB = {height}')
print(f'Dimension RGB = {dimension}')
```

```
Width RGB = 250
Height RGB = 250
Dimension RGB = 3
```

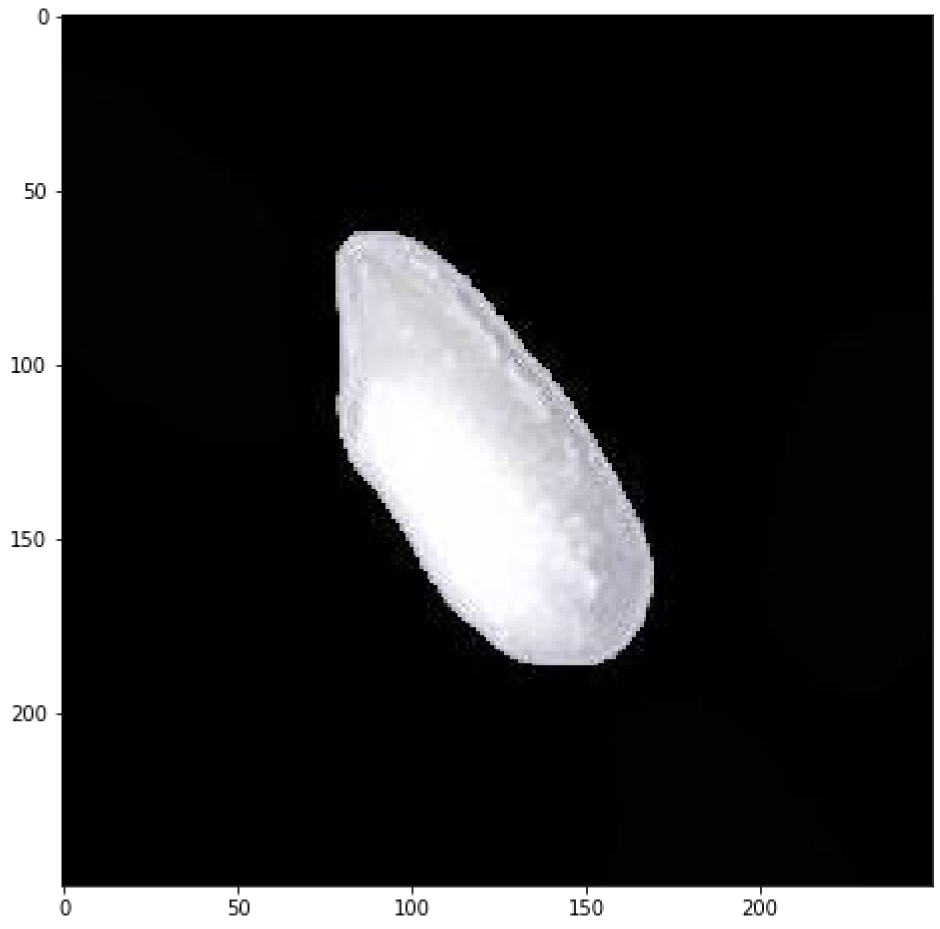


Figure 9: Loading image and visualizing function

```
arborio_rice_gray = cv2.cvtColor(arborio_rice, cv2.COLOR_RGB2GRAY)
plot_image(arborio_rice_gray)
width, height = arborio_rice_gray.shape
print(f'Width Grayscale = {width}')
print(f'Height Grayscale = {height}')
print(f'Image Shape Grayscale {arborio_rice_gray.shape}')
```

```
Width Grayscale = 250
Height Grayscale = 250
Image Shape Grayscale (250, 250)
```

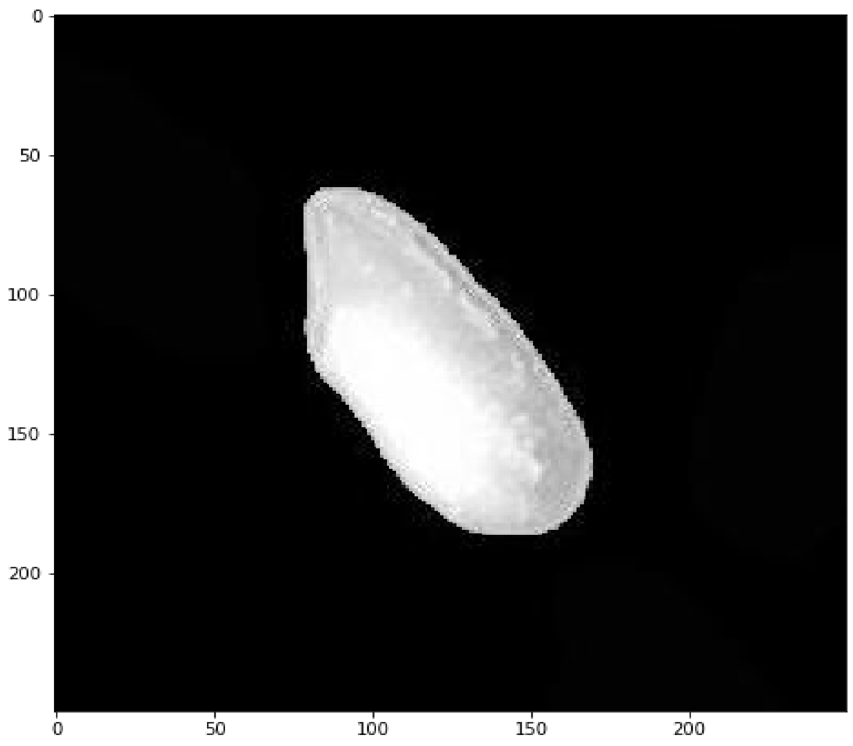


Figure 10: Checking image in gray scale

```
arborio_rice_thresh = cv2.adaptiveThreshold(arborio_rice_gray,5,cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV,11,3)
plot_image(arborio_rice_thresh)
```

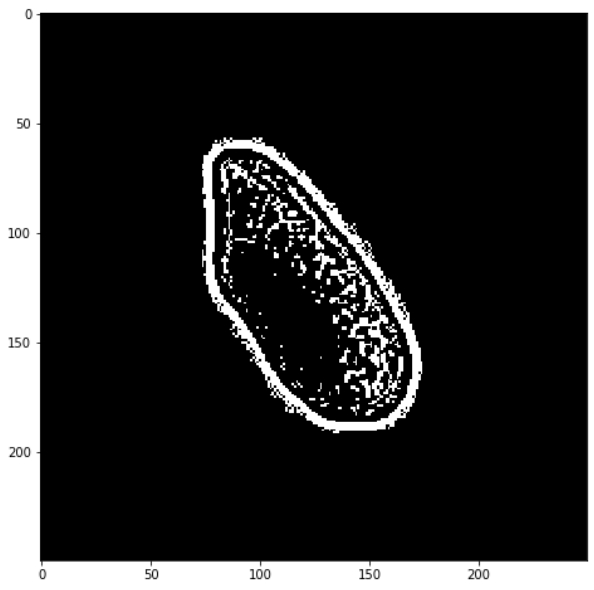


Figure 11: Checking adaptive threshold of images

```
contours = cv2.findContours(arborio_rice_thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
contours = contours[0] if len(contours) == 2 else contours[1]
contours = sorted(contours, key=cv2.contourArea, reverse=True)
for c in contours:
    x,y,w,h = cv2.boundingRect(c)
    arborio_rice_ROI = arborio_rice[y:y+h, x:x+w]
    break
plot_image(arborio_rice_ROI)
width, height, dimension = arborio_rice_ROI.shape
print(f'Width = {width}')
print(f'Height = {height}')
print(f'Dimension = {dimension}')
```

```
Width = 135
Height = 101
Dimension = 3
```

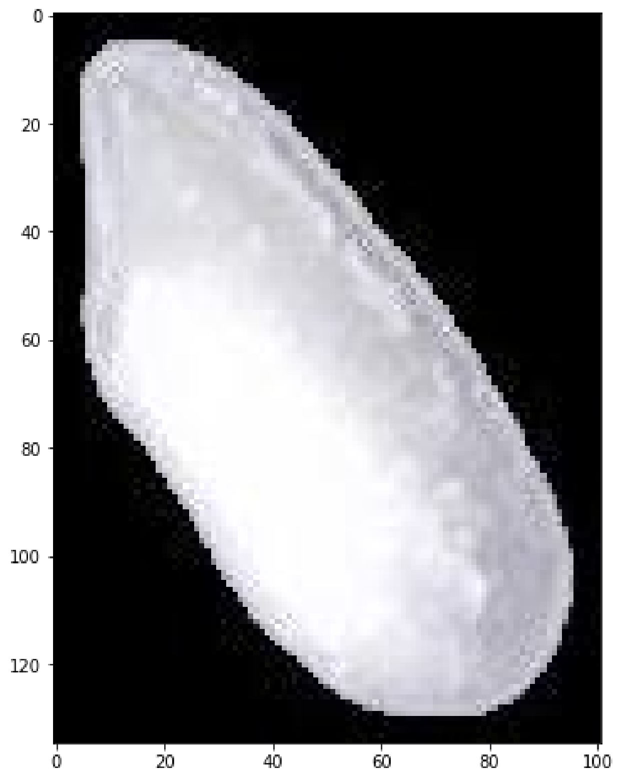


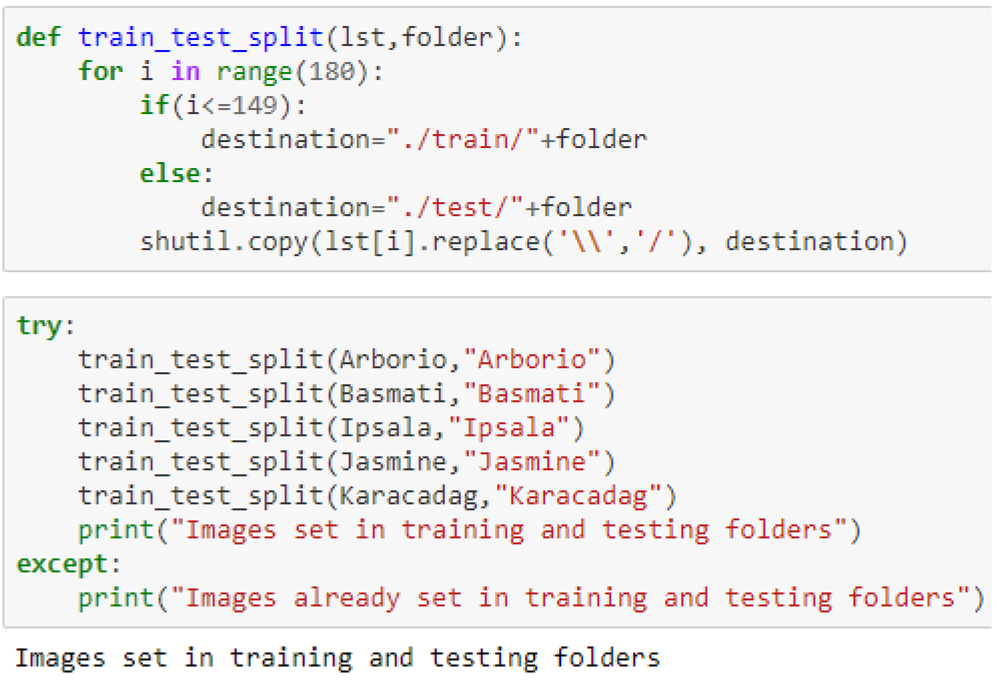
Figure 12: Checking contours in images

7 Data Preparation

```
try:
    os.makedirs("train")
    os.makedirs("test")
    for category in classes:
        path = os.path.join("./train", category)
        os.makedirs(path)
        path = os.path.join("./test", category)
        os.makedirs(path)
    print("Folders created")
except:
    print("Folders already created")
```

```
Folders already created
```

Figure 13: Image Directory train and test

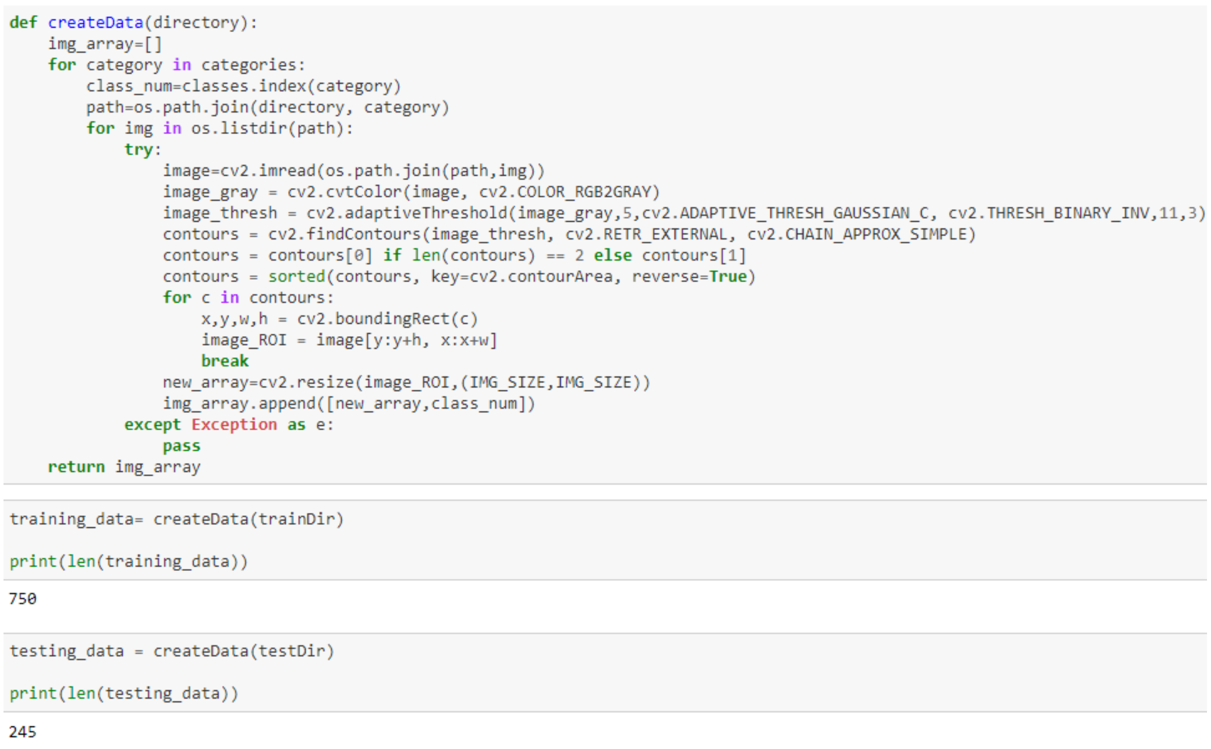


```python
def train_test_split(lst,folder):
    for i in range(180):
        if(i<=149):
            destination="./train/"+folder
        else:
            destination="./test/"+folder
        shutil.copy(lst[i].replace('\\','/'), destination)
```

```python
try:
    train_test_split(Arborio,"Arborio")
    train_test_split(Basmati,"Basmati")
    train_test_split(Ipsala,"Ipsala")
    train_test_split(Jasmine,"Jasmine")
    train_test_split(Karacadag,"Karacadag")
    print("Images set in training and testing folders")
except:
    print("Images already set in training and testing folders")
```

```
Images set in training and testing folders
```

Figure 14: Creating train and test split



```python
def createData(directory):
    img_array=[]
    for category in categories:
        class_num=classes.index(category)
        path=os.path.join(directory, category)
        for img in os.listdir(path):
            try:
                image=cv2.imread(os.path.join(path,img))
                image_gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
                image_thresh = cv2.adaptiveThreshold(image_gray,5,cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV,11,3)
                contours = cv2.findContours(image_thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
                contours = contours[0] if len(contours) == 2 else contours[1]
                contours = sorted(contours, key=cv2.contourArea, reverse=True)
                for c in contours:
                    x,y,w,h = cv2.boundingRect(c)
                    image_ROI = image[y:y+h, x:x+w]
                    break
                new_array=cv2.resize(image_ROI,(IMG_SIZE,IMG_SIZE))
                img_array.append([new_array,class_num])
            except Exception as e:
                pass
    return img_array
```

```python
training_data= createData(trainDir)

print(len(training_data))
```

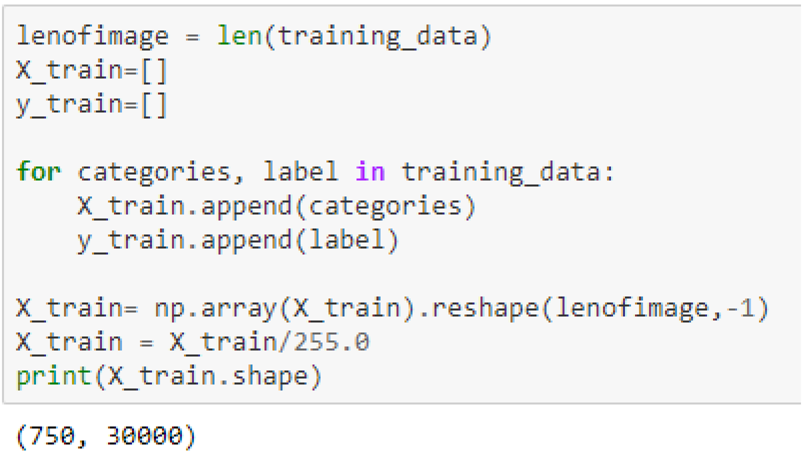
```
750
```

```python
testing_data = createData(testDir)

print(len(testing_data))
```

```
245
```

Figure 15: Creating image matrix



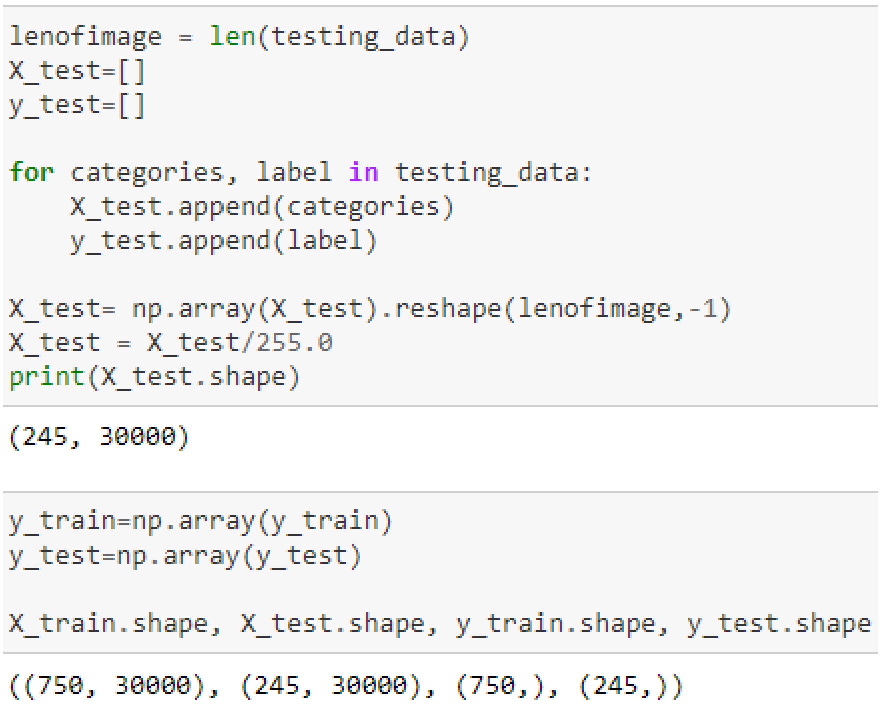
```python
lenofimage = len(training_data)
X_train=[]
y_train=[]

for categories, label in training_data:
    X_train.append(categories)
    y_train.append(label)

X_train= np.array(X_train).reshape(lenofimage,-1)
X_train = X_train/255.0
print(X_train.shape)
```

```
(750, 30000)
```

Figure 16: Transforming training data to 1D



```python
lenofimage = len(testing_data)
X_test=[]
y_test=[]

for categories, label in testing_data:
    X_test.append(categories)
    y_test.append(label)

X_test= np.array(X_test).reshape(lenofimage,-1)
X_test = X_test/255.0
print(X_test.shape)
```

```
(245, 30000)
```

```python
y_train=np.array(y_train)
y_test=np.array(y_test)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((750, 30000), (245, 30000), (750,), (245,))
```

Figure 17: Transforming testing data to 1D

8 Machine Learning Models

8.1 SVM

```
model=  svm.SVC(gamma='scale', C=20, kernel = 'linear')
```

```
model.fit(X_train,y_train)
```

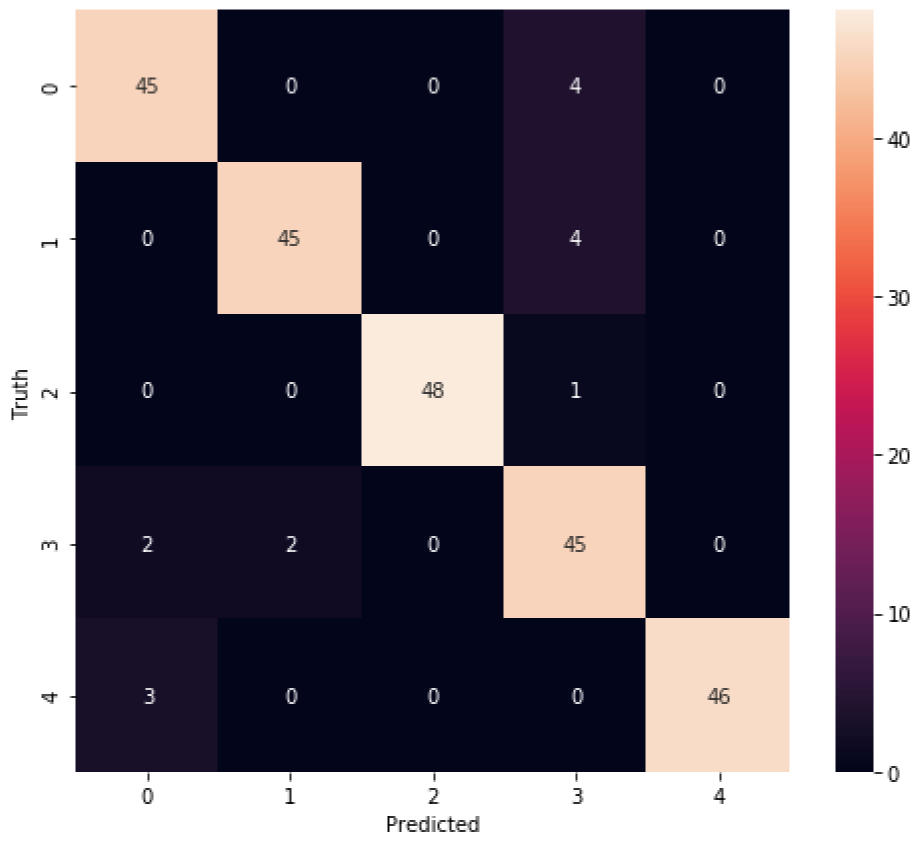
```
SVC(C=20, kernel='linear')
```

```
accuracy = model.score(X_test,y_test)*100
accuracy
```

```
93.46938775510203
```

```
y_predicted = model.predict(X_test)
cm = confusion_matrix(y_test,y_predicted)
plt.figure(figsize = (8,7))
sb.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(51.0, 0.5, 'Truth')
```



```
score = []
score.append(["SVM" , accuracy])
```

Figure 18: Implementation of SVM

8.2 Decision Trees

```
model =  DecisionTreeClassifier()
```

```
model.fit(X_train,y_train)
```

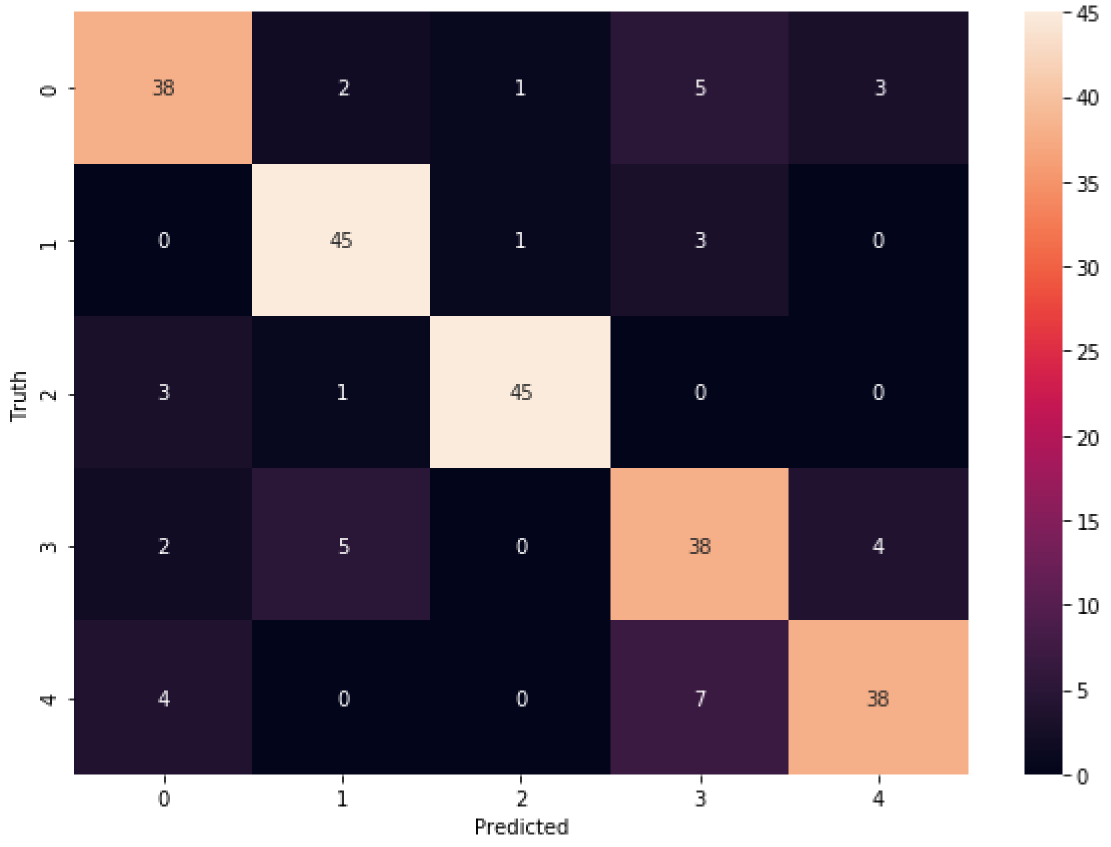
```
DecisionTreeClassifier()
```

```
accuracy = model.score(X_test,y_test)*100
accuracy
```

```
83.26530612244898
```

```
y_predicted = model.predict(X_test)
cm = confusion_matrix(y_test,y_predicted)
plt.figure(figsize = (10,7))
sb.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(69.0, 0.5, 'Truth')
```



```
score.append(["Decision Tree" , accuracy])
```

Figure 19: Implementation of Desicion Trees

8.3 Random Forest Trees

```
model =  RandomForestClassifier()
```

```
model.fit(X_train,y_train)
```

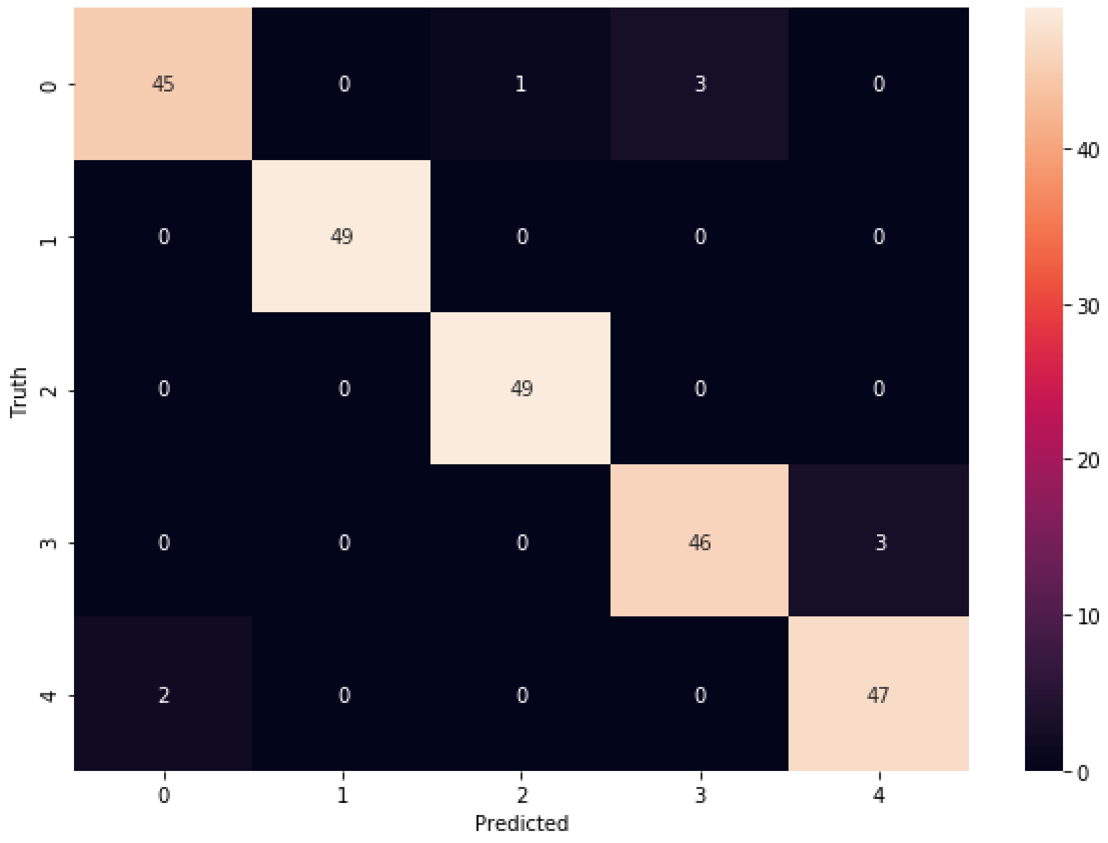
```
RandomForestClassifier()
```

```
accuracy = model.score(X_test,y_test)*100
accuracy
```

```
96.3265306122449
```

```
y_predicted = model.predict(X_test)
cm = confusion_matrix(y_test,y_predicted)
plt.figure(figsize = (10,7))
sb.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

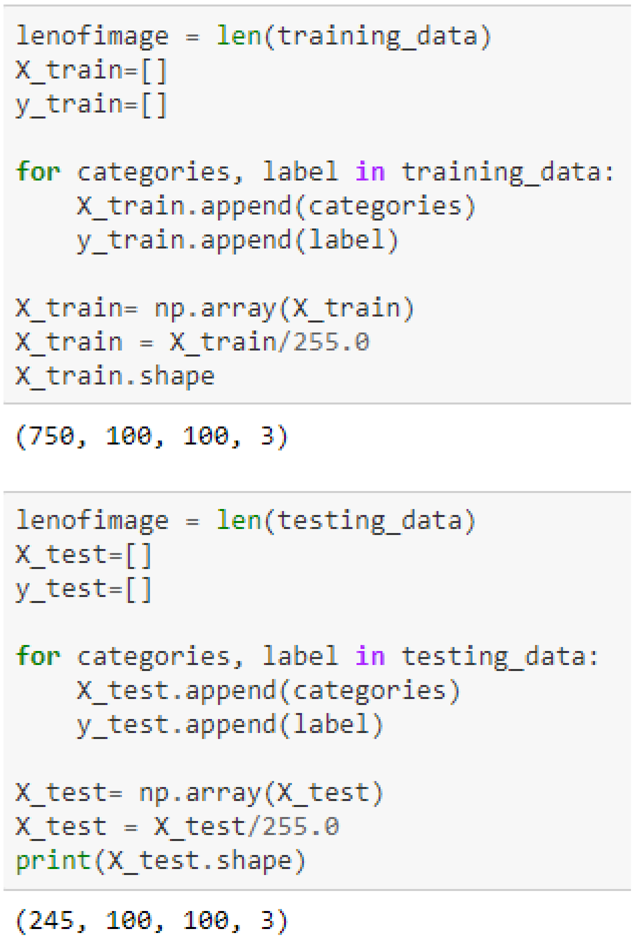
```
Text(69.0, 0.5, 'Truth')
```



```
score.append(["Random Forest Tree" , accuracy])
```

Figure 20: Implementation of Random Forest Tree

8.4 CNN



```
lenofimage = len(training_data)
X_train=[]
y_train=[]

for categories, label in training_data:
    X_train.append(categories)
    y_train.append(label)

X_train= np.array(X_train)
X_train = X_train/255.0
X_train.shape
```

```
(750, 100, 100, 3)
```

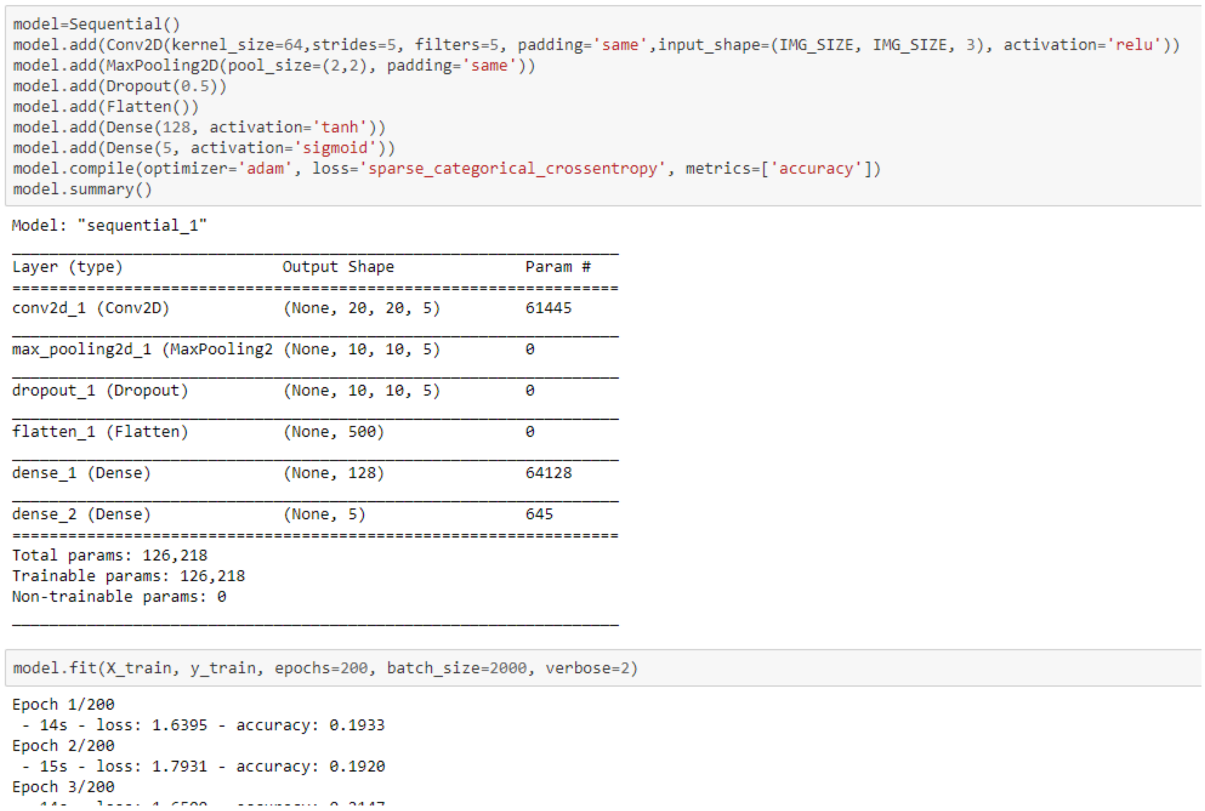
```
lenofimage = len(testing_data)
X_test=[]
y_test=[]

for categories, label in testing_data:
    X_test.append(categories)
    y_test.append(label)

X_test= np.array(X_test)
X_test = X_test/255.0
print(X_test.shape)
```

```
(245, 100, 100, 3)
```

Figure 21: Generating 2D Image data for CNN



```
model=Sequential()
model.add(Conv2D(kernel_size=64,strides=5, filters=5, padding='same',input_shape=(IMG_SIZE, IMG_SIZE, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), padding='same'))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, activation='tanh'))
model.add(Dense(5, activation='sigmoid'))
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 20, 20, 5)         61445
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 10, 10, 5)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 10, 10, 5)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 500)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               64128
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 645
=================================================================
Total params: 126,218
Trainable params: 126,218
Non-trainable params: 0
_________________________________________________________________
```

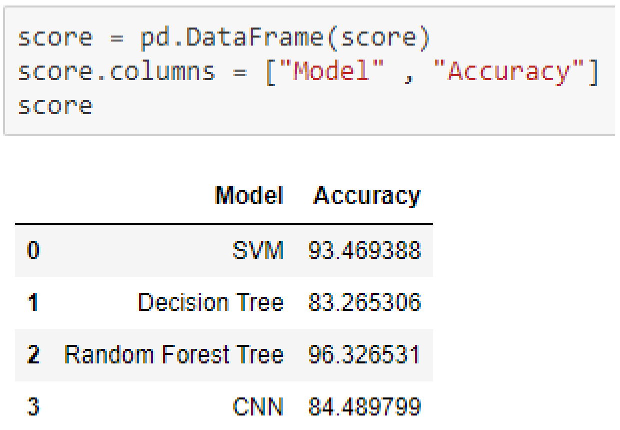
```
model.fit(X_train, y_train, epochs=200, batch_size=2000, verbose=2)
```

```
Epoch 1/200
 - 14s - loss: 1.6395 - accuracy: 0.1933
Epoch 2/200
 - 15s - loss: 1.7931 - accuracy: 0.1920
Epoch 3/200
 - 14s - loss: 1.6500 - accuracy: 0.2147
```

Figure 22: Implementation of CNN

9 Model result

This section explains the performance of the models.



```
score = pd.DataFrame(score)
score.columns = ["Model" , "Accuracy"]
score
```

	Model	Accuracy
0	SVM	93.469388
1	Decision Tree	83.265306
2	Random Forest Tree	96.326531
3	CNN	84.489799

Figure 23: Model Performance

```
plt.figure(figsize=(8,7))
plt.bar(score['Model'], score['Accuracy'])
plt.plot(score['Model'], score['Accuracy'], marker='o')
```

```
[<matplotlib.lines.Line2D at 0x1cedcdf0fc8>]
```

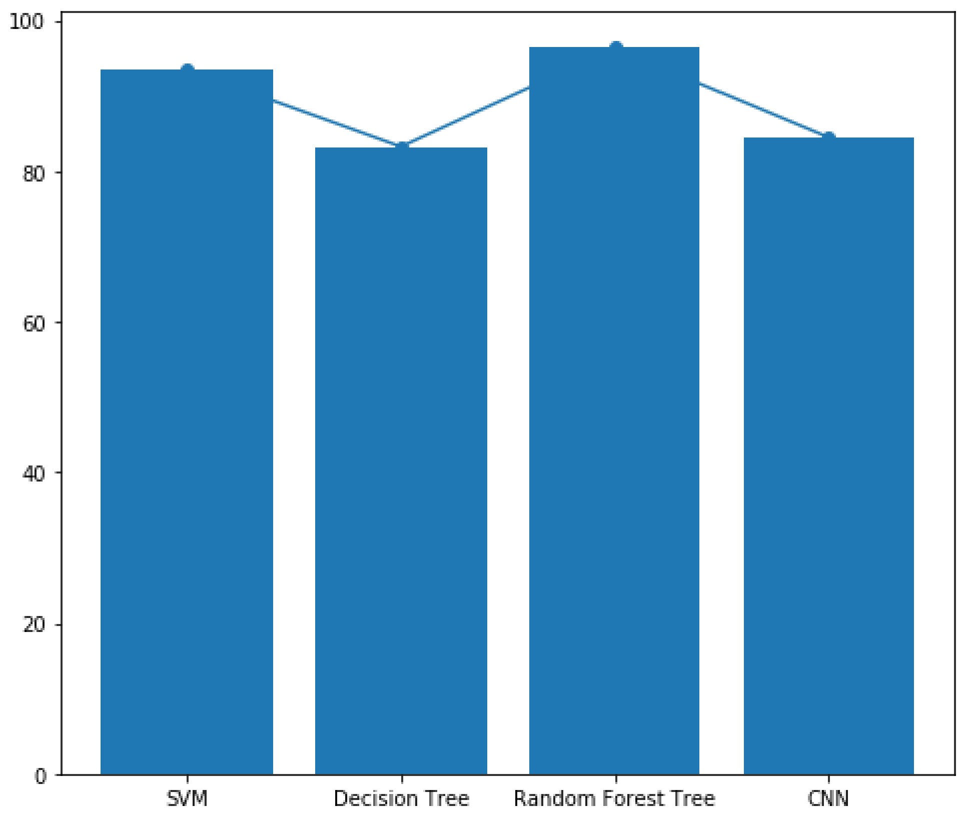


Figure 24: Plot for Model Accuracy Scores

References

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

https://www.tensorflow.org/tutorials/images/cnn