RAMADAN INTERNSHIP PROGRAM 2023

INTERNSHIP PROGRESS REPORT

PERSONAL DETAILS

STUDENT NAME	Ushna Hamza
ROLL NO	CS-016
BATCH	2020
DEPARTMENT	Computer and Information Systems Engineering
CONTACT NO.	034351993101
EMAIL ID	ushnahamza2001@gmail.com
INTERNSHIP DURATION	4 weeks
NAME OF PROJECT	Writers' recognition
MENTOR/TRAINER	Sir Rafay Mustafa

TASK ASSIGNED

WEEK#	OBJECTIVE
WEEK-1	Skew correction Study task Manual Text Line Segmentation
WEEK-2	Pixel padding Filtering Patch scanning Data augmentation
WEEK-3	Image cleaning Test and train folders split
WEEK-4	Feature extraction and classification

Please add additional rows as required

TASK COMPLETION DETAILS

Brief Description about the project :

NLP (natural language processing) and Machine learning, in many cases are used together to develop more advanced models that can understand and interpret human language in complex ways. This project also combines techniques from computer vision, natural language processing and machine learning to develop an effective writer recognition model that has the potential to be used in a wide range of applications.

The model is designed to analyze the handwritten Urdu paragraphs of different individuals and recognize handwriting from digital images.

To design this model, first we performed line segmentation and then applied various image processing techniques to clean the data. This data is then used to train a deep neural network which then be used to identify the writer behind that sample .

Write a Detailed Description of the assigned task

(Attach relevant document/references/code/Images/results)

TASK 1: Research

IMAGE PROCESSING

Image Processing is the use of algorithms and techniques to analyze and manipulate digital images, often to improve their quality and extract some useful information from them.

IMAGE PROCESSING TECHNIQUES

FILTERING

In image processing, filtering is a process of modifying an image by applying mathematical operation or function called filter.

Purpose

The goal of using filters is to modify or enhance image properties and/or to extract valuable information from the pictures such as edges or corners

SEGMENTATION

Segmentation in image processing refers to the process of converting an image into a collection of segments or regions, with each segment representing a meaningful object or region of interest.

Purpose

The goal of segmentation is to simplify an image and make it easier to analyze or process by separating the relevant objects or regions from the background or noise.

IMAGE AUGMENTATION

Image augmentation is a technique in image processing that involves creating new, modified versions of existing images by applying a series of transformations or manipulations to the original images.

Purpose

Image augmentation artificially expands the size and diversity of a training dataset, which can improve the accuracy and generalization of machine learning models.

PATCH SCANNING

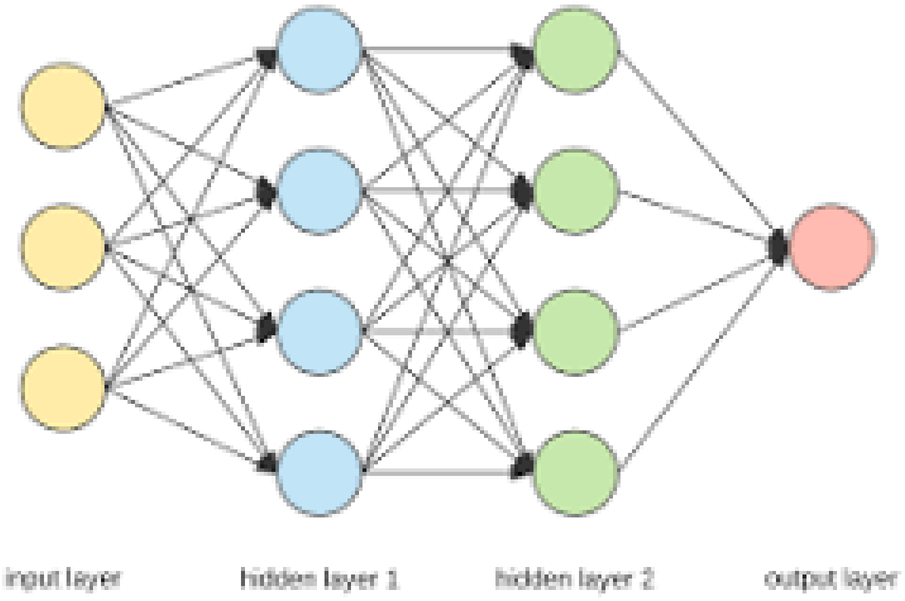
In image processing, patch scanning is a technique that involves dividing an image into small patches, and then scanning each patch to extract relevant features or information.

Purpose

The purpose of patch scanning is to reduce complexity of the image and identify specific features within an image that are relevant to a particular task or application.

ARCHITECTURE OF NEURAL NETWORK

Neural networks are a type of machine learning model inspired by the structure and function of the human brain. They are composed of artificial neurons that are interconnected and work together to perform tasks such as classification, regression, and pattern recognition.



SOME COMMON NEURAL NETWORK

1.Alexnet

AlexNet is a convolutional neural network (CNN) architecture proposed in 2012 in the research paper by Alex Krizhevsky and his colleagues. It won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 and was the first deep learning model to achieve significant breakthroughs in computer vision tasks such as object recognition and image classification.

3.YOLO

YOLO is a popular deep learning model that can detect objects in images and videos in real-time, with high accuracy and speed.

3.VGG

VGG (Visual Geometry Group) is a standard deep Convolutional Neural Network (CNN) architecture with multiple layers for image classification tasks. The “deep” refers to the number of layers with VGG-16 or VGG-19 consisting of 16 and 19 convolutional layers.

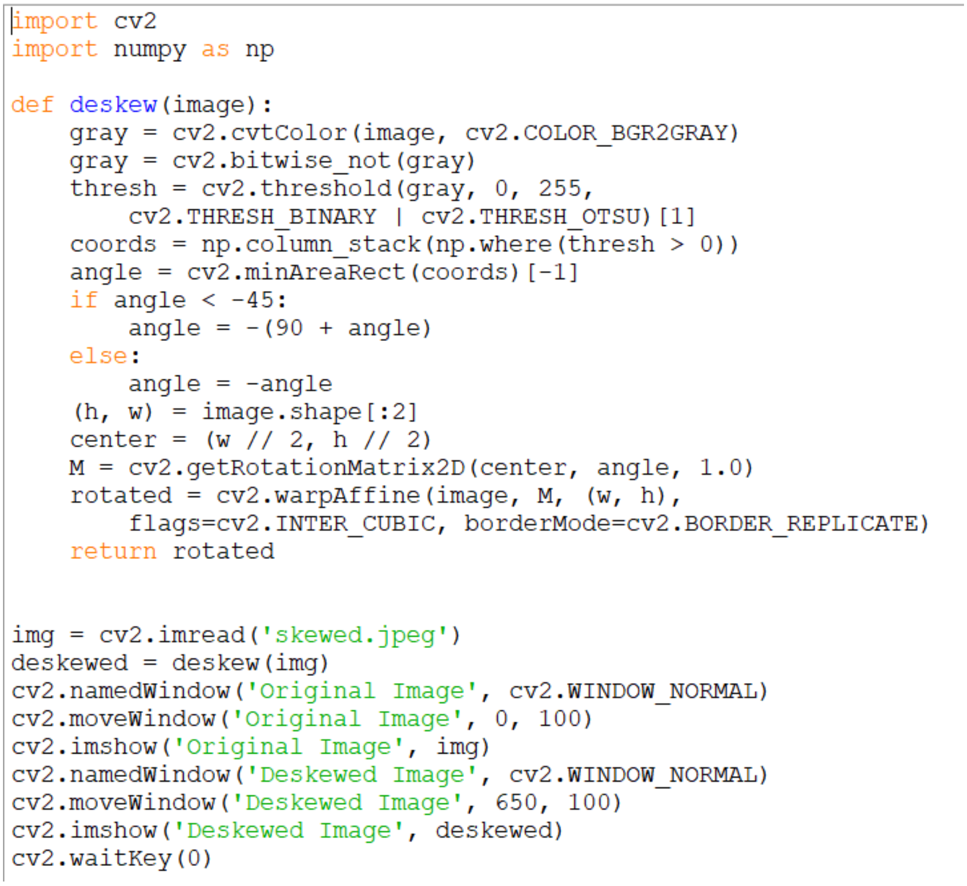
4.RESNET

ResNet is a deep neural network architecture that was introduced by Microsoft Research in 2015. ResNet won the ImageNet competition in 2015, and its performance has been further improved with subsequent versions.

Task 2: Skew Correct an Image

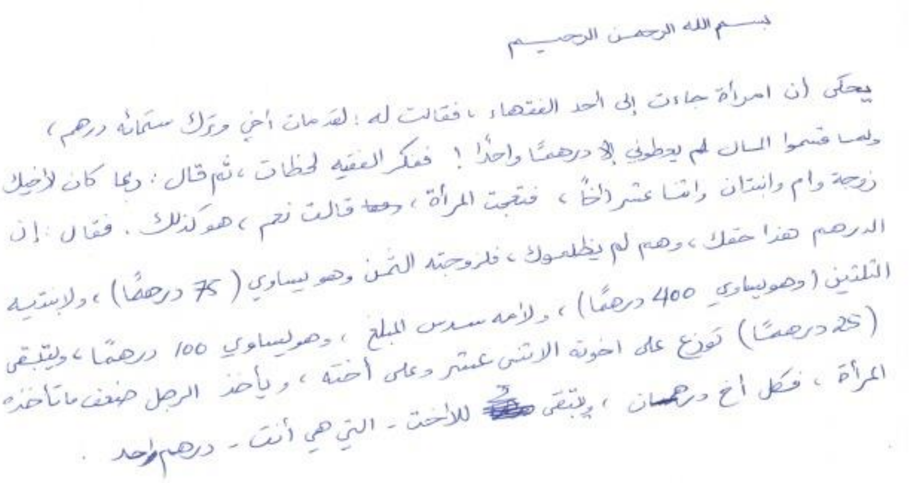
Given a skewed image, perform skew correction using appropriate algorithm

CODE



```
import cv2
import numpy as np

def deskew(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray = cv2.bitwise_not(gray)
    thresh = cv2.threshold(gray, 0, 255,
        cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]
    coords = np.column_stack(np.where(thresh > 0))
    angle = cv2.minAreaRect(coords)[-1]
    if angle < -45:
        angle = -(90 + angle)
    else:
        angle = -angle
    (h, w) = image.shape[:2]
    center = (w // 2, h // 2)
    M = cv2.getRotationMatrix2D(center, angle, 1.0)
    rotated = cv2.warpAffine(image, M, (w, h),
        flags=cv2.INTER_CUBIC, borderMode=cv2.BORDER_REPLICATE)
    return rotated


img = cv2.imread('skewed.jpeg')
deskewed = deskew(img)
cv2.namedWindow('Original Image', cv2.WINDOW_NORMAL)
cv2.moveWindow('Original Image', 0, 100)
cv2.imshow('Original Image', img)
cv2.namedWindow('Deskewed Image', cv2.WINDOW_NORMAL)
cv2.moveWindow('Deskewed Image', 650, 100)
cv2.imshow('Deskewed Image', deskewed)
cv2.waitKey(0)
```

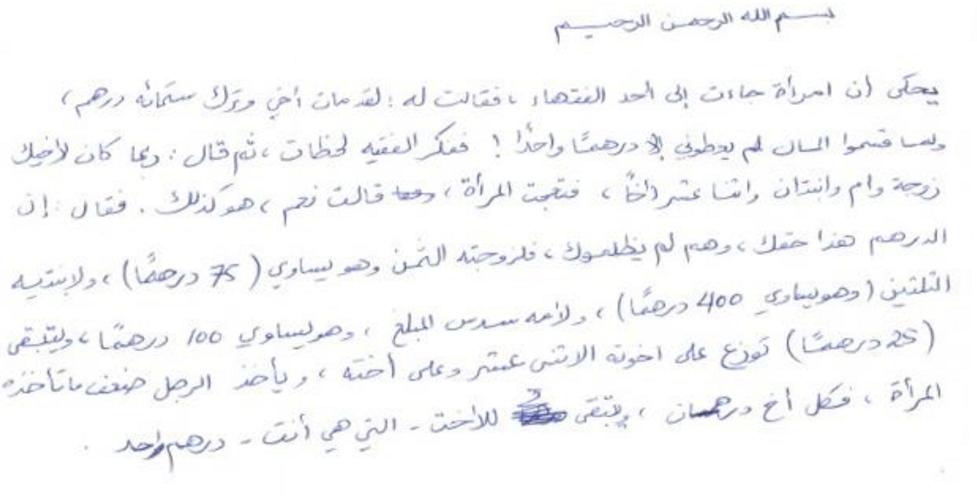
ORIGINAL IMAGE



بسم الله الرحمن الرحيم

يحكى أن امرأة جاءت إلى احد الفقهاء ، فقالت له : لقد مات أخي وترك ستمائة درهم ،
ولما قسموا المال لم يعطوني إلا درهمًا واحدًا ! ففكر الفقيه لحظات ، ثم قال : ربما كان لأخيك
زوجة وأم وابنتان واثنا عشر أخًا ، فتعجبت المرأة ، وهنا قالت نعم ، هو كذلك . فقال : إن
الدرهم هذا حقك ، وهم لم يظلموك ، فلزوجته الثمن وهو يساوي (75 درهمًا) ، ولابنتيه
الثلثين (وهو يساوي 400 درهمًا) ، ولأمه سدس المبلغ ، وهو يساوي 100 درهمًا ، ويتبقى
(25 درهمًا) توزع على اخوته الاثنى عشر وعلى أخته ، ويأخذ الرجل ضعف ما تأخذه
المرأة ، فكل أخ درهمان ، ويتبقى للأخت - التي هي أنت - درهم واحد .

OUTPUT IMAGE

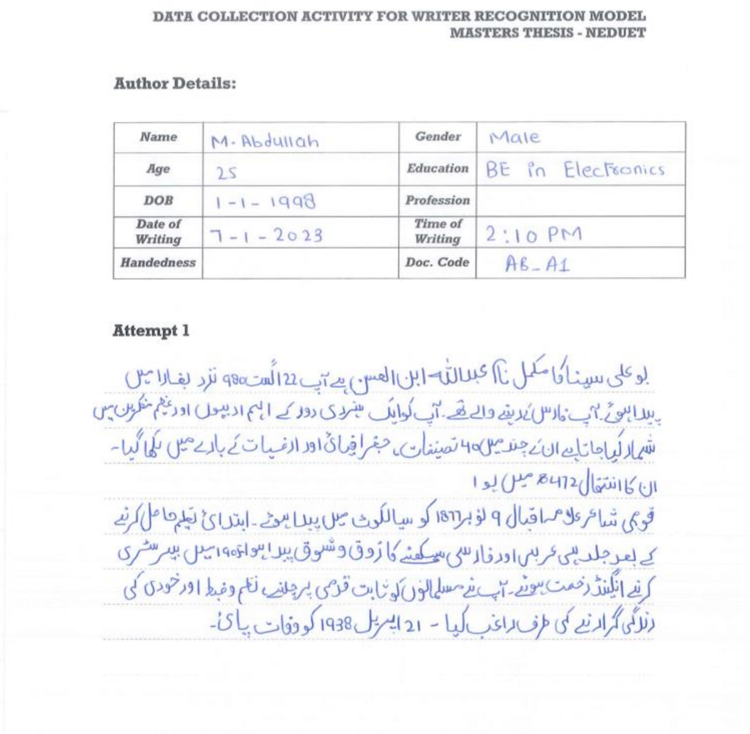


بسم الله الرحمن الرحيم

يحكى أن امرأة جاءت إلى احد الفقهاء ، فقالت له : لقد مات أخي وترك ستمائة درهم ،
ولما قسموا المال لم يعطوني إلا درهمًا واحدًا ! ففكر الفقيه لحظات ، ثم قال : ربما كان لأخيك
زوجة وأم وابنتان واثنا عشر أخًا ، فتعجبت المرأة ، وهنا قالت نعم ، هو كذلك . فقال : إن
الدرهم هذا حقك ، وهم لم يظلموك ، فلزوجته الثمن وهو يساوي (75 درهمًا) ، ولابنتيه
الثلثين (وهو يساوي 400 درهمًا) ، ولأمه سدس المبلغ ، وهو يساوي 100 درهمًا ، ويتبقى
(25 درهمًا) توزع على اخوته الاثنى عشر وعلى أخته ، ويأخذ الرجل ضعف ما تأخذه
المرأة ، فكل أخ درهمان ، ويتبقى للأخت - التي هي أنت - درهم واحد .

TASK 3: Manual Text Line Segmentation

Seven folders of different writers were given to us as data samples. Each folder had 6 different images. These images had handwritten Urdu paragraph. 5 out of these 6 images had the same paragraph while one of them had free text. We were given the task to manually split every image into line segments and label them accordingly.



DATA COLLECTION ACTIVITY FOR WRITER RECOGNITION MODEL
MASTERS THESIS - NEDUET

Author Details:

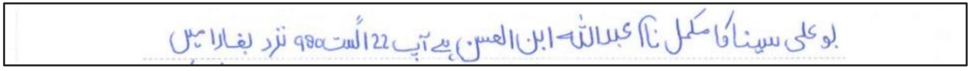
Name	M. Abdullah	Gender	Male
Age	25	Education	BE in Electronics
DOB	1-1-1998	Profession	
Date of Writing	7-1-2023	Time of Writing	2:10 PM
Handedness		Doc. Code	AB_A1

Attempt 1

بو علی سینا کا مکمل نام عبداللہ ابن الحسن ہے آپ 22 اگست ۹۸۰ء نزد بخارا میں
پیدا ہوئے۔ آپ فارس کے رہنے والے تھے۔ آپ کو ایک شہری دور کے اہم ادیبوں اور عظیم مفکرین میں
شمار کیا جاتا ہے ان کے چند میں ۴۰ تصنیفات، جغرافیائی اور ارضیات کے بارے میں لکھا گیا۔
ان کا انتقال 472ء میں ہوا
قومی شاعر علامہ اقبال ۹ نومبر ۱۸۷۷ کو سیالکوٹ میں پیدا ہوئے۔ ابتدائی تعلیم حاصل کرنے
کے بعد جلد ہی عربی اور فارسی سیکھنے کا ذوق و شوق پیدا ہوا ۱۹۰۵ء میں بیرسٹری
کرنے انگلینڈ رخصت ہوئے۔ آپ نے مسلمانوں کو ثابت قدمی پر چلنے نظم و ضبط اور خودی کی
زندگی گزارنے کی طرف راغب کیا - 21 اپریل 1938 کو وفات پائی۔

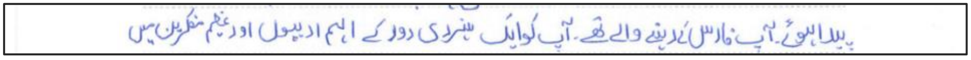
AB_A1

OUTPUT



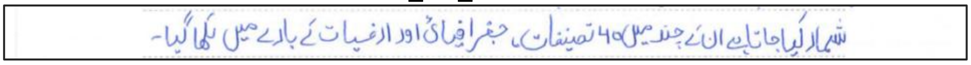
بو علی سینا کا مکمل نام عبداللہ ابن الحسن ہے آپ 22 اگست ۹۸۰ء نزد بخارا میں

AB_A1_L1



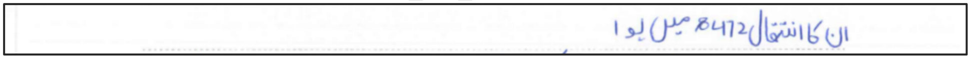
پیدا ہوئے۔ آپ فارس کے رہنے والے تھے۔ آپ کو ایک شہری دور کے اہم ادیبوں اور عظیم مفکرین میں

AB_A1_L2



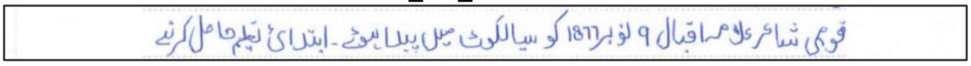
شمار کیا جاتا ہے ان کے چند میں ۴۰ تصنیفات، جغرافیائی اور ارضیات کے بارے میں لکھا گیا۔

AB_A1_L3



ان کا انتقال 472ء میں ہوا

AB_A1_L4



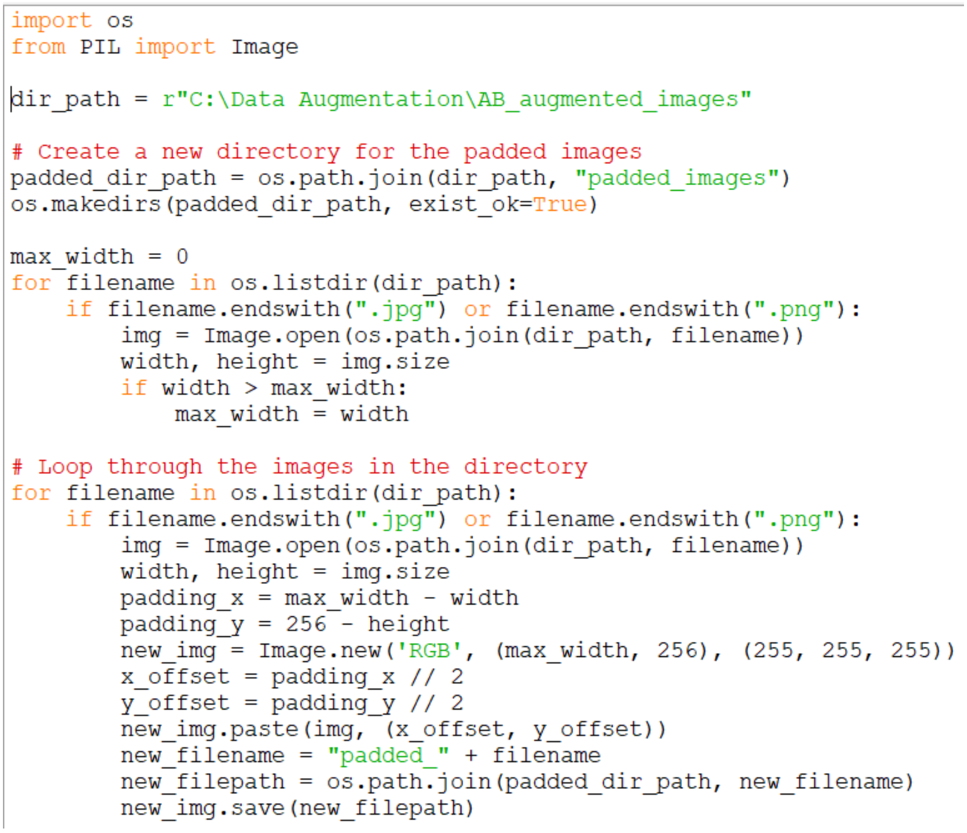
قومی شاعر علامہ اقبال ۹ نومبر ۱۸۷۷ کو سیالکوٹ میں پیدا ہوئے۔ ابتدائی تعلیم حاصل کرنے

AB_A1_L5

Task 4: White Pixel Padding

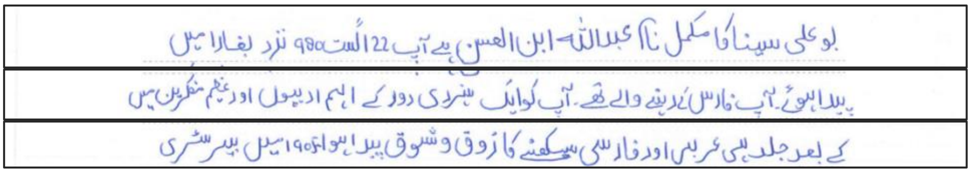
White pixel padding refers to the technique of adding white pixels around the edges of an image to increase its size without distorting its content. After line segmentation, images had different dimensions. We added white pixels to make sure that all the images have the same dimensions, that is 256 height and maximum width.

CODE



```python
import os
from PIL import Image

dir_path = r"C:\Data Augmentation\AB_augmented_images"

# Create a new directory for the padded images
padded_dir_path = os.path.join(dir_path, "padded_images")
os.makedirs(padded_dir_path, exist_ok=True)

max_width = 0
for filename in os.listdir(dir_path):
    if filename.endswith(".jpg") or filename.endswith(".png"):
        img = Image.open(os.path.join(dir_path, filename))
        width, height = img.size
        if width > max_width:
            max_width = width

# Loop through the images in the directory
for filename in os.listdir(dir_path):
    if filename.endswith(".jpg") or filename.endswith(".png"):
        img = Image.open(os.path.join(dir_path, filename))
        width, height = img.size
        padding_x = max_width - width
        padding_y = 256 - height
        new_img = Image.new('RGB', (max_width, 256), (255, 255, 255))
        x_offset = padding_x // 2
        y_offset = padding_y // 2
        new_img.paste(img, (x_offset, y_offset))
        new_filename = "padded_" + filename
        new_filepath = os.path.join(padded_dir_path, new_filename)
        new_img.save(new_filepath)
```

INPUT IMAGES

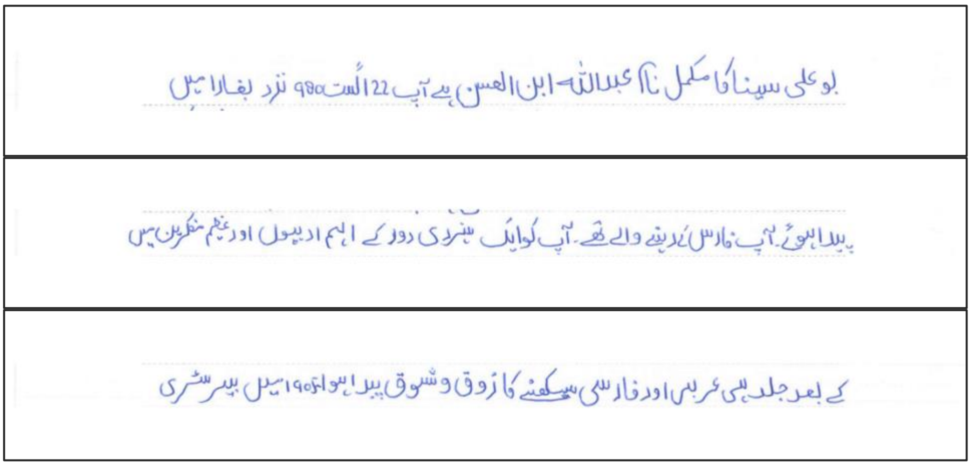


بوعلی سینا کا مکمل نام عبداللہ ابن الحسن ہے آپ 22 اگست ۹۸۰ء نزد بخارا میں

پیدا ہوئے۔ آپ فارس کے رہنے والے تھے۔ آپ کو ایک سنہری دور کے اہم ادیبوں اور عظیم مفکرین میں

کے بعد جلد ہی عربی اور فارسی سیکھنے کا ذوق و شوق پیدا ہوا ۱۹۰۵ء میں بیرسٹری

OUTPUT IMAGES



بو علی سینا کا مکمل نام عبداللہ ابن الحسن ہے آپ 22 اگست ۹۸۰ نزد بخارا میں

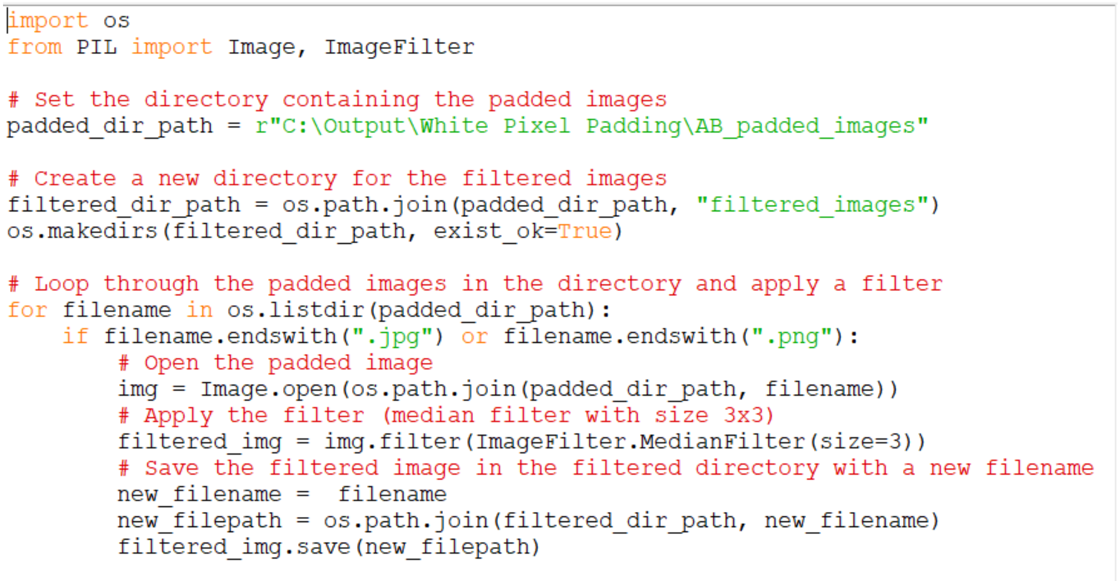
پیدا ہوئے۔ آپ فارس کے رہنے والے تھے۔ آپ کو ایک سنہری دور کے اہم ادیبوں اور عظیم مفکرین میں

کے بعد جلد ہی عربی اور فارسی سیکھنے کا ذوق و شوق پیدا ہوا ۱۹۰۵ میں بیرسٹری

TASK 5: Filtering

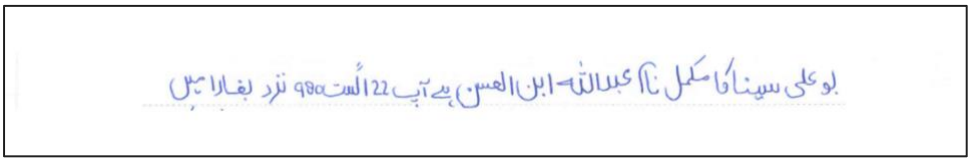
Filters are used to remove noise, enhance certain features of an image, or to blur or sharpen an image. Here median filter is used to remove the noise from the images.

CODE



```
import os
from PIL import Image, ImageFilter

# Set the directory containing the padded images
padded_dir_path = r"C:\Output\White Pixel Padding\AB_padded_images"

# Create a new directory for the filtered images
filtered_dir_path = os.path.join(padded_dir_path, "filtered_images")
os.makedirs(filtered_dir_path, exist_ok=True)

# Loop through the padded images in the directory and apply a filter
for filename in os.listdir(padded_dir_path):
    if filename.endswith(".jpg") or filename.endswith(".png"):
        # Open the padded image
        img = Image.open(os.path.join(padded_dir_path, filename))
        # Apply the filter (median filter with size 3x3)
        filtered_img = img.filter(ImageFilter.MedianFilter(size=3))
        # Save the filtered image in the filtered directory with a new filename
        new_filename =  filename
        new_filepath = os.path.join(filtered_dir_path, new_filename)
        filtered_img.save(new_filepath)
```

OUTPUT IMAGE



بوعلی سینا کا مکمل نام عبداللہ ابن الحسن ہے آپ 22 اگست ۹۸۰ نزد بخارا میں

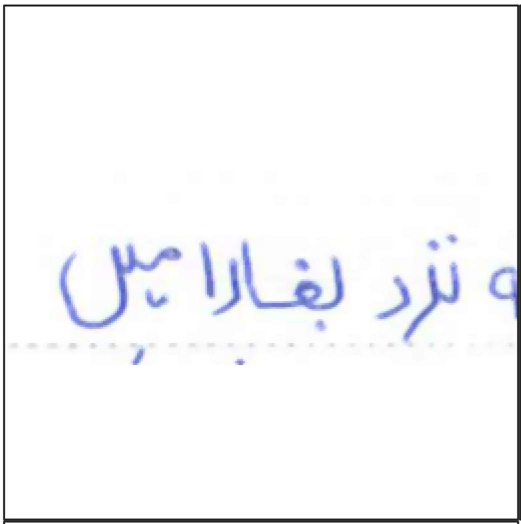
TASK 6: Patch Scanning

Patch scanning is a technique to extract small patches from an image for further processing. We did patch scanning on filtered images to divide them into small fragments of 265*256.

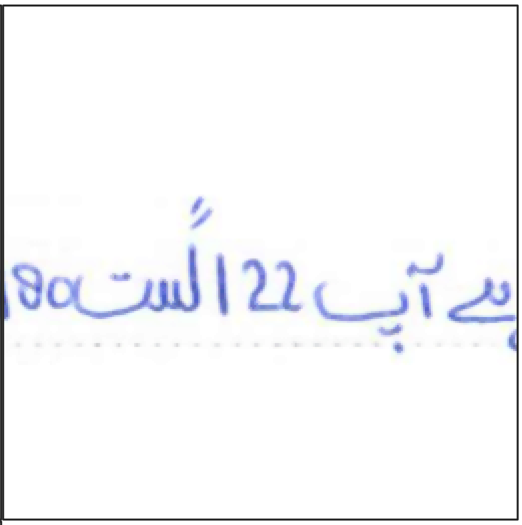
CODE

```
import os
from PIL import Image

# Define the input and output directories
input_dir = r"C:\Output\Filtering\AB_filtered_images"
output_dir = r"C:\Output\Patch Scanning\AB_scanned_images"

# Create the output directory if it doesn't exist
if not os.path.exists(output_dir):
    os.mkdir(output_dir)

# Loop through all the images in the input directory
for filename in os.listdir(input_dir):
    if filename.endswith(".png") or filename.endswith(".jpg"):
        input_image = Image.open(os.path.join(input_dir, filename))
        width, height = input_image.size

        # Loop through the image in 256*256 fragments and save each fragment as a separate image
        for i in range(0, width, 256):
            for j in range(0, height, 256):
                box = (i, j, i+256, j+256)
                output_image = input_image.crop(box)
                output_filename = os.path.splitext(filename)[0] + "_{}_{}.png".format(i, j)
                output_path = os.path.join(output_dir, output_filename)
                output_image.save(output_path)
```

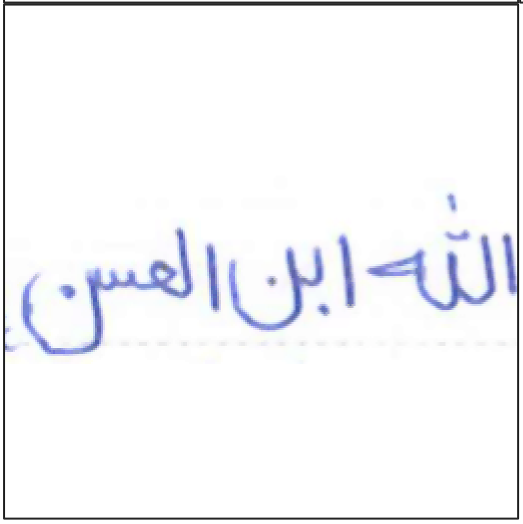
OUTPUT IMAGE



۹ نزد لغارا میں



پے آپ 122 اُلست ۸۵

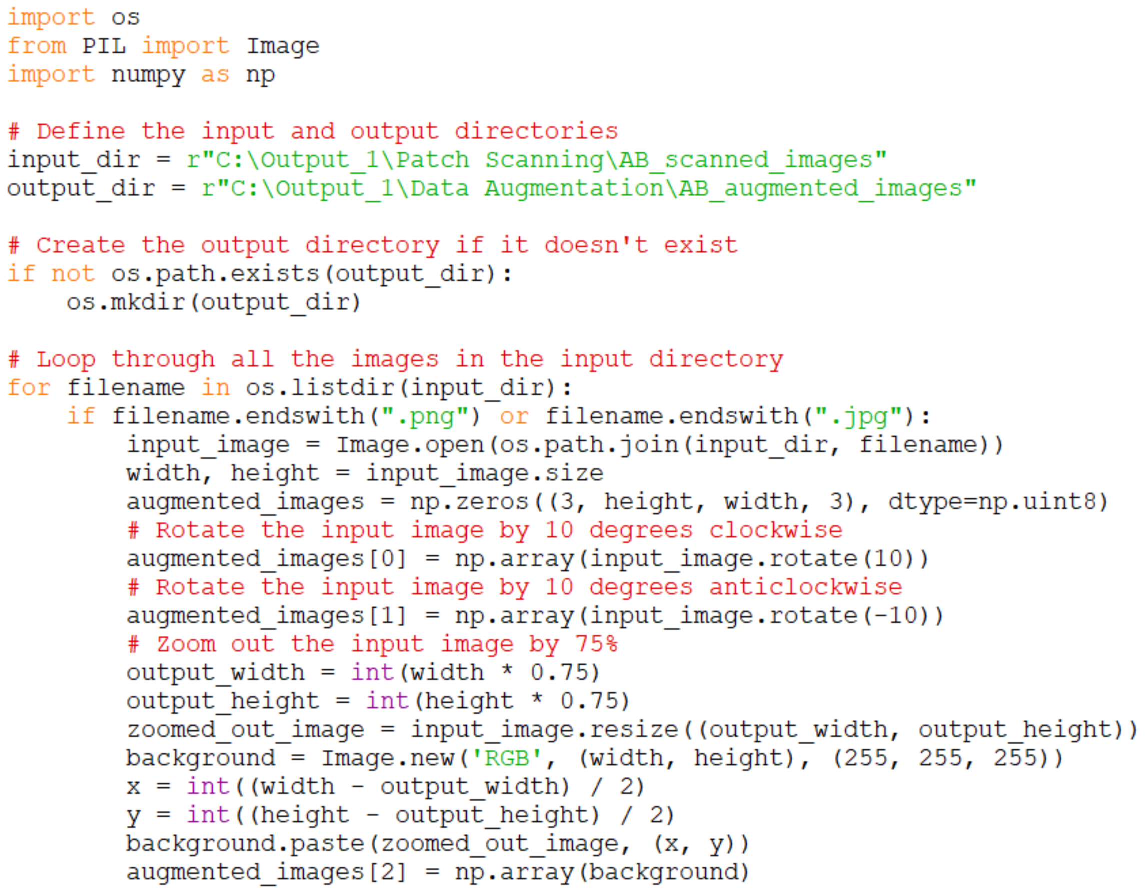


اللہ ابن الحسن

TASK 7: Data Augmentation

Image augmentation expands the size and diversity of a training dataset. We have applied data augmentation to have modified versions of existing patches. Here we have rotated the images 10 degree clockwise and anticlockwise and zoomed out by 75%.

CODE



```
import os
from PIL import Image
import numpy as np

# Define the input and output directories
input_dir = r"C:\Output_1\Patch Scanning\AB_scanned_images"
output_dir = r"C:\Output_1\Data Augmentation\AB_augmented_images"

# Create the output directory if it doesn't exist
if not os.path.exists(output_dir):
    os.mkdir(output_dir)

# Loop through all the images in the input directory
for filename in os.listdir(input_dir):
    if filename.endswith(".png") or filename.endswith(".jpg"):
        input_image = Image.open(os.path.join(input_dir, filename))
        width, height = input_image.size
        augmented_images = np.zeros((3, height, width, 3), dtype=np.uint8)
        # Rotate the input image by 10 degrees clockwise
        augmented_images[0] = np.array(input_image.rotate(10))
        # Rotate the input image by 10 degrees anticlockwise
        augmented_images[1] = np.array(input_image.rotate(-10))
        # Zoom out the input image by 75%
        output_width = int(width * 0.75)
        output_height = int(height * 0.75)
        zoomed_out_image = input_image.resize((output_width, output_height))
        background = Image.new('RGB', (width, height), (255, 255, 255))
        x = int((width - output_width) / 2)
        y = int((height - output_height) / 2)
        background.paste(zoomed_out_image, (x, y))
        augmented_images[2] = np.array(background)
```

OUTPUT IMAGES

Zoomed out

10 deg clockwise

10 deg anticlockwise

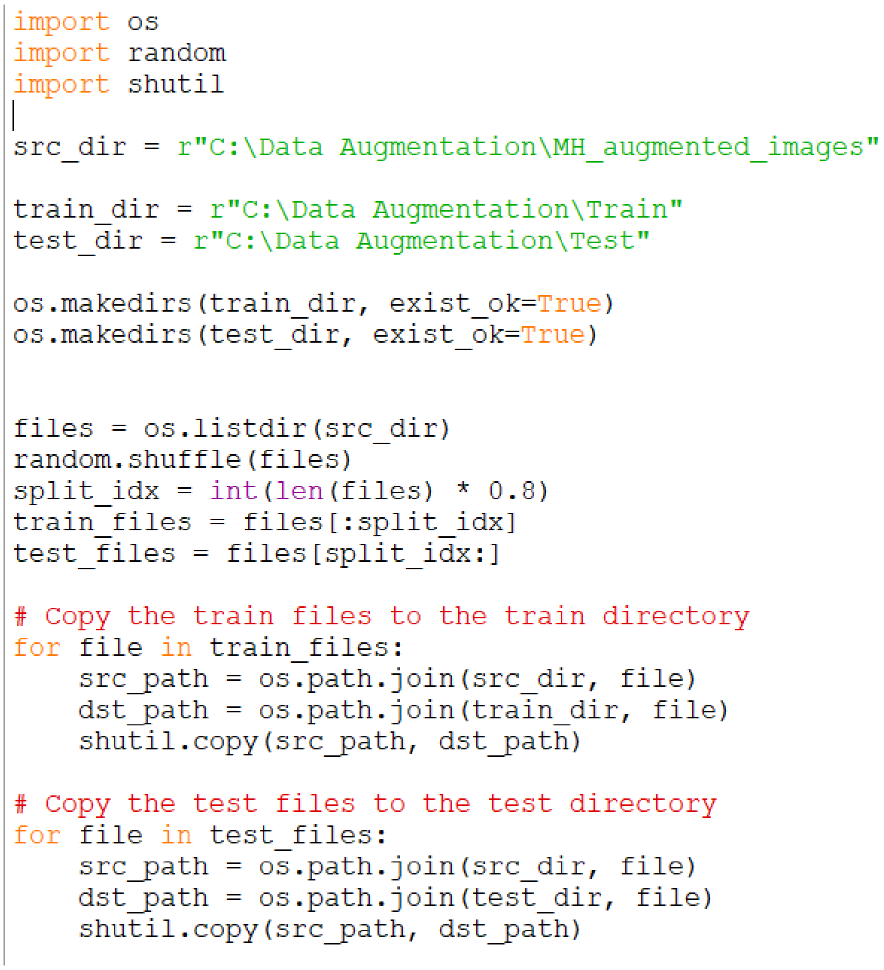
TASK 8: Image Cleaning

After applying all the above image process techniques, we removed all the images with null value or the images that contained single characters. This will ensure more accuracy at the time of feature extraction.

TASK 9: Train/Test Split

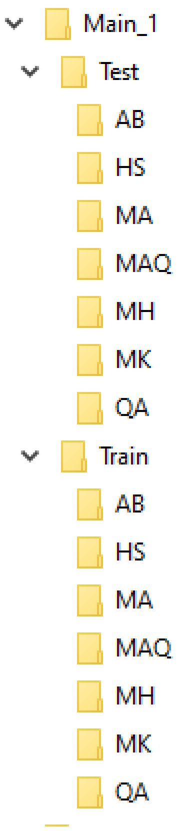
After cleaning the data, we divided the final augmented images into two folders. 80% of the images were included in the train folder while 20% of the images were in the test folder.

CODE



```
import os
import random
import shutil

src_dir = r"C:\Data Augmentation\MH_augmented_images"

train_dir = r"C:\Data Augmentation\Train"
test_dir = r"C:\Data Augmentation\Test"

os.makedirs(train_dir, exist_ok=True)
os.makedirs(test_dir, exist_ok=True)


files = os.listdir(src_dir)
random.shuffle(files)
split_idx = int(len(files) * 0.8)
train_files = files[:split_idx]
test_files = files[split_idx:]

# Copy the train files to the train directory
for file in train_files:
    src_path = os.path.join(src_dir, file)
    dst_path = os.path.join(train_dir, file)
    shutil.copy(src_path, dst_path)

# Copy the test files to the test directory
for file in test_files:
    src_path = os.path.join(src_dir, file)
    dst_path = os.path.join(test_dir, file)
    shutil.copy(src_path, dst_path)
```

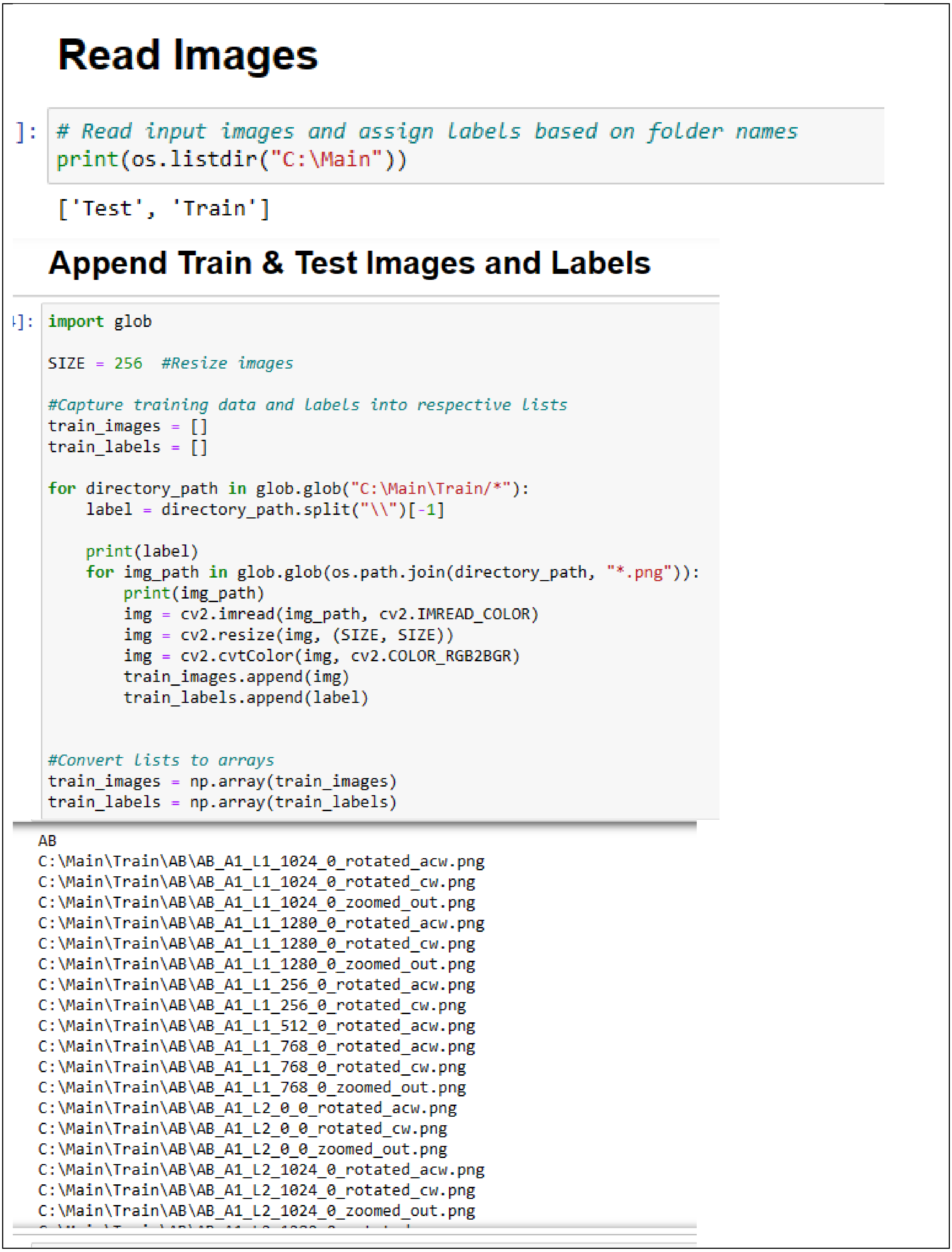
OUTPUT DIRECTORY



```
Main_1
  Test
    AB
    HS
    MA
    MAQ
    MH
    MK
    QA
  Train
    AB
    HS
    MA
    MAQ
    MH
    MK
    QA
```

TASK 10: Feature Extraction/ Classification/ Results

In the final step, we gave our directory path to the code which was given to us.



Read Images

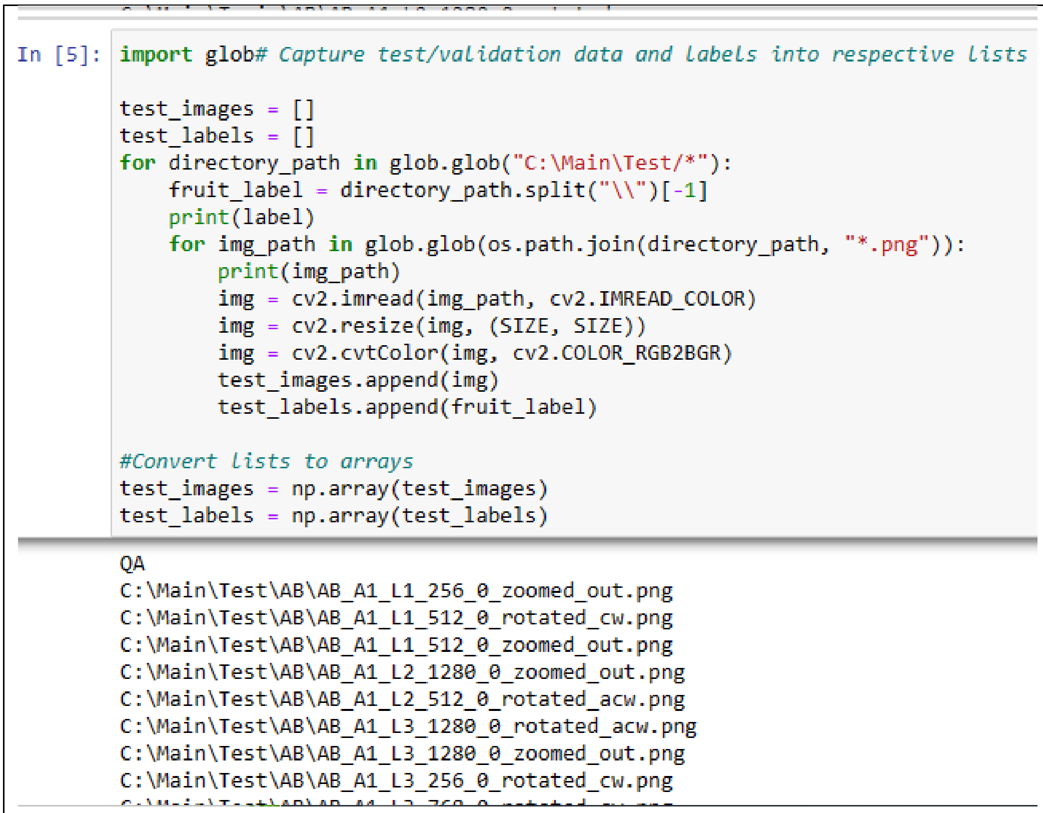
```
# Read input images and assign labels based on folder names
print(os.listdir("C:\Main"))
```

```
['Test', 'Train']
```

Append Train & Test Images and Labels

```
import glob

SIZE = 256  #Resize images

#Capture training data and labels into respective lists
train_images = []
train_labels = []

for directory_path in glob.glob("C:\Main\Train/*"):
    label = directory_path.split("\\")[-1]

    print(label)
    for img_path in glob.glob(os.path.join(directory_path, "*.png")):
        print(img_path)
        img = cv2.imread(img_path, cv2.IMREAD_COLOR)
        img = cv2.resize(img, (SIZE, SIZE))
        img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
        train_images.append(img)
        train_labels.append(label)


#Convert lists to arrays
train_images = np.array(train_images)
train_labels = np.array(train_labels)
```

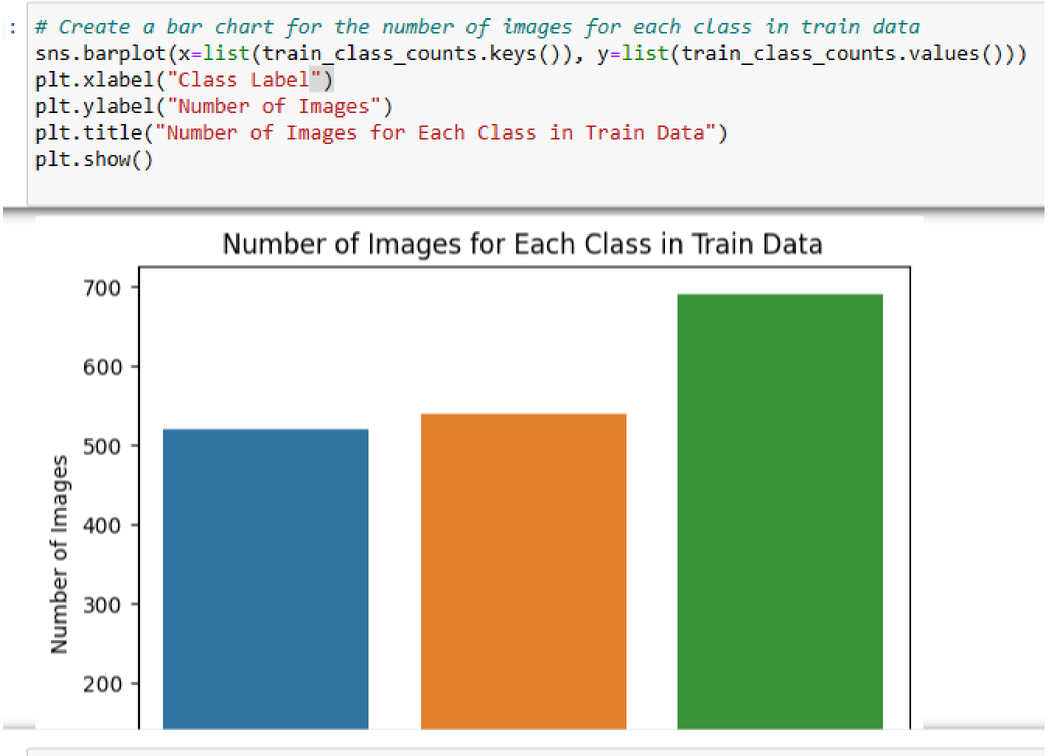
```
AB
C:\Main\Train\AB\AB_A1_L1_1024_0_rotated_acw.png
C:\Main\Train\AB\AB_A1_L1_1024_0_rotated_cw.png
C:\Main\Train\AB\AB_A1_L1_1024_0_zoomed_out.png
C:\Main\Train\AB\AB_A1_L1_1280_0_rotated_acw.png
C:\Main\Train\AB\AB_A1_L1_1280_0_rotated_cw.png
C:\Main\Train\AB\AB_A1_L1_1280_0_zoomed_out.png
C:\Main\Train\AB\AB_A1_L1_256_0_rotated_acw.png
C:\Main\Train\AB\AB_A1_L1_256_0_rotated_cw.png
C:\Main\Train\AB\AB_A1_L1_512_0_rotated_acw.png
C:\Main\Train\AB\AB_A1_L1_768_0_rotated_acw.png
C:\Main\Train\AB\AB_A1_L1_768_0_rotated_cw.png
C:\Main\Train\AB\AB_A1_L1_768_0_zoomed_out.png
C:\Main\Train\AB\AB_A1_L2_0_0_rotated_acw.png
C:\Main\Train\AB\AB_A1_L2_0_0_rotated_cw.png
C:\Main\Train\AB\AB_A1_L2_0_0_zoomed_out.png
C:\Main\Train\AB\AB_A1_L2_1024_0_rotated_acw.png
C:\Main\Train\AB\AB_A1_L2_1024_0_rotated_cw.png
C:\Main\Train\AB\AB_A1_L2_1024_0_zoomed_out.png
```



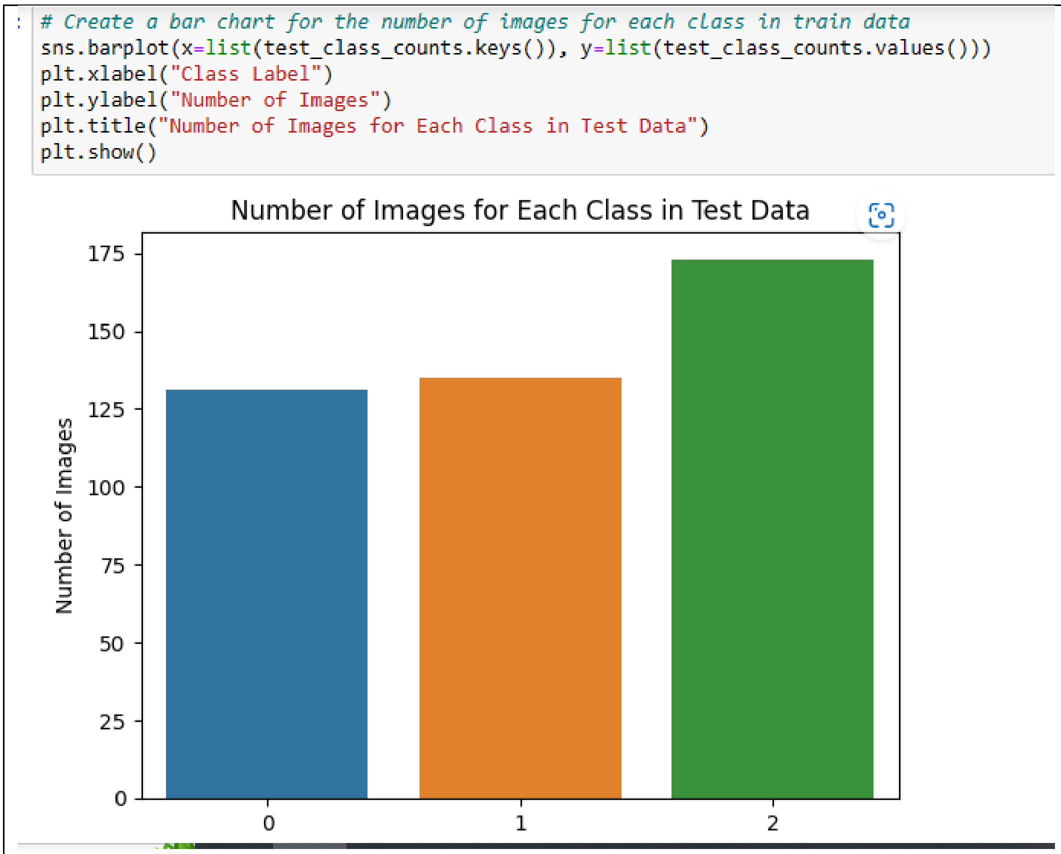
```
In [5]: import glob# Capture test/validation data and labels into respective lists

        test_images = []
        test_labels = []
        for directory_path in glob.glob("C:\Main\Test/*"):
            fruit_label = directory_path.split("\\")[-1]
            print(label)
            for img_path in glob.glob(os.path.join(directory_path, "*.png")):
                print(img_path)
                img = cv2.imread(img_path, cv2.IMREAD_COLOR)
                img = cv2.resize(img, (SIZE, SIZE))
                img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
                test_images.append(img)
                test_labels.append(fruit_label)

        #Convert lists to arrays
        test_images = np.array(test_images)
        test_labels = np.array(test_labels)
```

```
QA
C:\Main\Test\AB\AB_A1_L1_256_0_zoomed_out.png
C:\Main\Test\AB\AB_A1_L1_512_0_rotated_cw.png
C:\Main\Test\AB\AB_A1_L1_512_0_zoomed_out.png
C:\Main\Test\AB\AB_A1_L2_1280_0_zoomed_out.png
C:\Main\Test\AB\AB_A1_L2_512_0_rotated_acw.png
C:\Main\Test\AB\AB_A1_L3_1280_0_rotated_acw.png
C:\Main\Test\AB\AB_A1_L3_1280_0_zoomed_out.png
C:\Main\Test\AB\AB_A1_L3_256_0_rotated_cw.png
C:\Main\Test\AB\AB_A1_L3_768_0_rotated_cw.png
```

Bar chart showing the number of images in train and test folder.

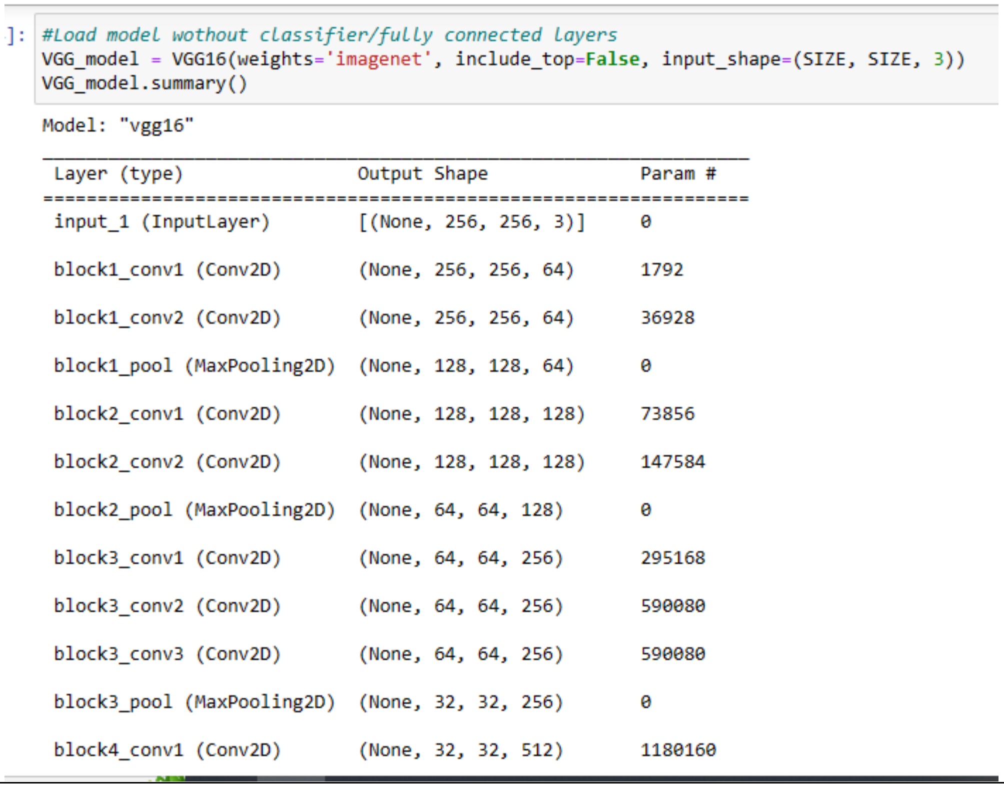


```
: # Create a bar chart for the number of images for each class in train data
  sns.barplot(x=list(train_class_counts.keys()), y=list(train_class_counts.values()))
  plt.xlabel("Class Label")
  plt.ylabel("Number of Images")
  plt.title("Number of Images for Each Class in Train Data")
  plt.show()
```



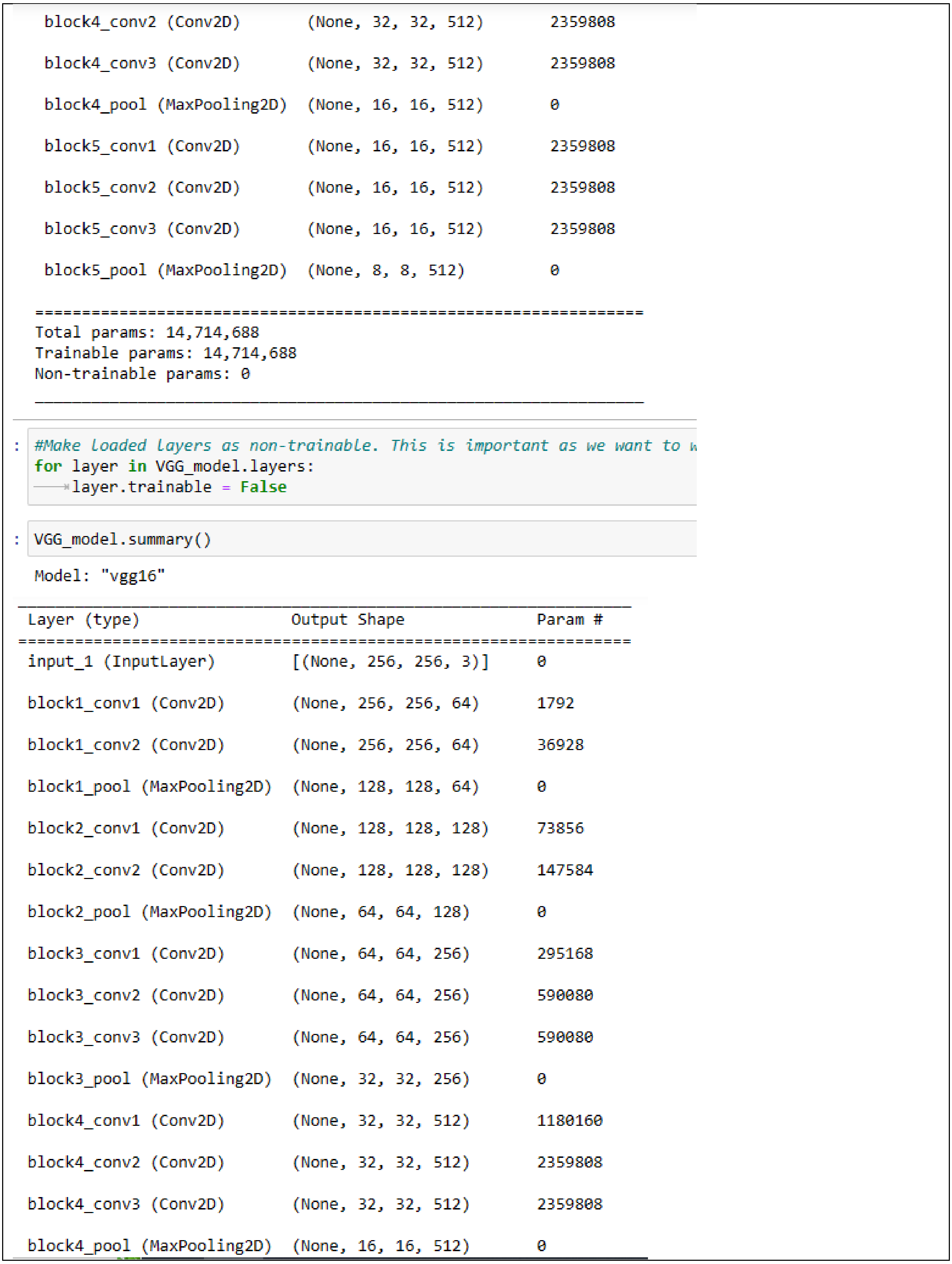
```
# Create a bar chart for the number of images for each class in train data
sns.barplot(x=list(test_class_counts.keys()), y=list(test_class_counts.values()))
plt.xlabel("Class Label")
plt.ylabel("Number of Images")
plt.title("Number of Images for Each Class in Test Data")
plt.show()
```

Here we used VGG for feature extraction.



```
#Load model wothout classifier/fully connected layers
VGG_model = VGG16(weights='imagenet', include_top=False, input_shape=(SIZE, SIZE, 3))
VGG_model.summary()
```

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_1 (InputLayer)        [(None, 256, 256, 3)]     0         
                                                                 
 block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792      
                                                                 
 block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928     
                                                                 
 block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0         
                                                                 
 block2_conv1 (Conv2D)       (None, 128, 128, 128)     73856     
                                                                 
 block2_conv2 (Conv2D)       (None, 128, 128, 128)     147584    
                                                                 
 block2_pool (MaxPooling2D)  (None, 64, 64, 128)       0         
                                                                 
 block3_conv1 (Conv2D)       (None, 64, 64, 256)       295168    
                                                                 
 block3_conv2 (Conv2D)       (None, 64, 64, 256)       590080    
                                                                 
 block3_conv3 (Conv2D)       (None, 64, 64, 256)       590080    
                                                                 
 block3_pool (MaxPooling2D)  (None, 32, 32, 256)       0         
                                                                 
 block4_conv1 (Conv2D)       (None, 32, 32, 512)       1180160   
```

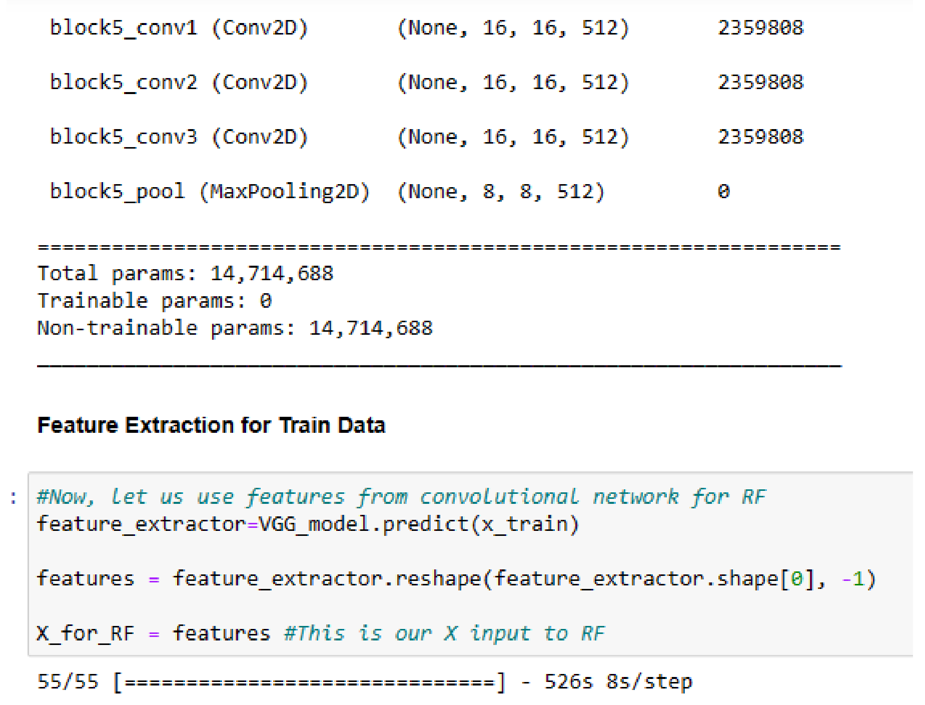


```
 block4_conv2 (Conv2D)       (None, 32, 32, 512)       2359808

 block4_conv3 (Conv2D)       (None, 32, 32, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 16, 16, 512)       0

 block5_conv1 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_conv2 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_conv3 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 8, 8, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

```
: #Make loaded layers as non-trainable. This is important as we want to w
  for layer in VGG_model.layers:
      layer.trainable = False
```

```
: VGG_model.summary()
```

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 256, 256, 3)]     0

 block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792

 block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928

 block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0

 block2_conv1 (Conv2D)       (None, 128, 128, 128)     73856

 block2_conv2 (Conv2D)       (None, 128, 128, 128)     147584

 block2_pool (MaxPooling2D)  (None, 64, 64, 128)       0

 block3_conv1 (Conv2D)       (None, 64, 64, 256)       295168

 block3_conv2 (Conv2D)       (None, 64, 64, 256)       590080

 block3_conv3 (Conv2D)       (None, 64, 64, 256)       590080

 block3_pool (MaxPooling2D)  (None, 32, 32, 256)       0

 block4_conv1 (Conv2D)       (None, 32, 32, 512)       1180160

 block4_conv2 (Conv2D)       (None, 32, 32, 512)       2359808

 block4_conv3 (Conv2D)       (None, 32, 32, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 16, 16, 512)       0
```



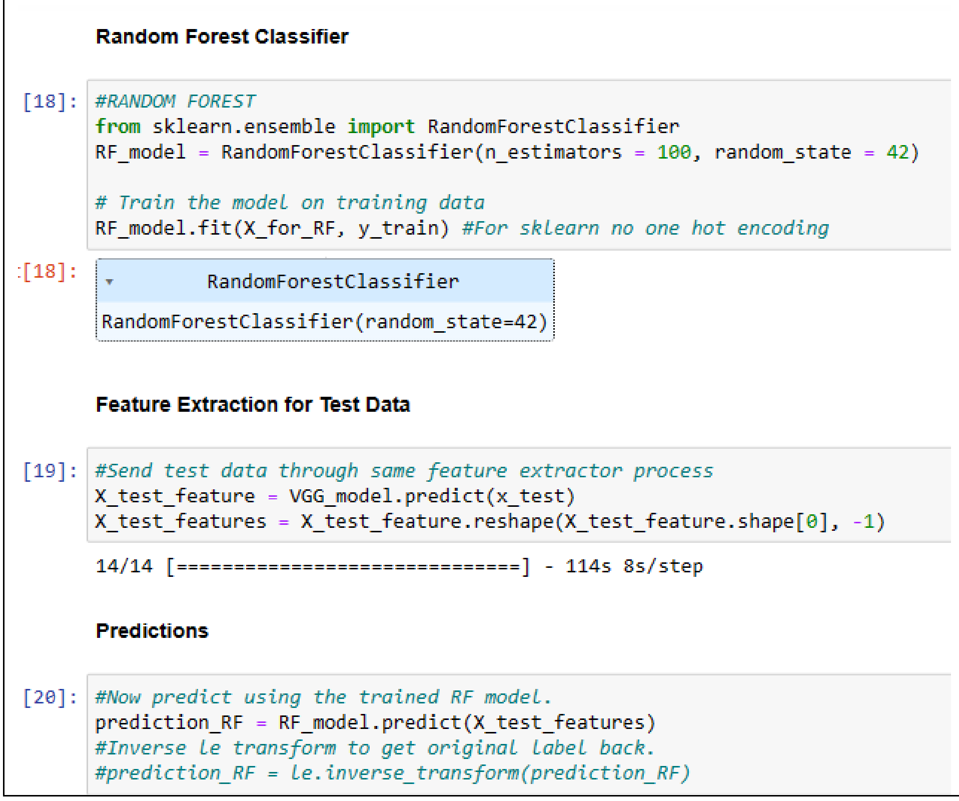
```
 block5_conv1 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_conv2 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_conv3 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 8, 8, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
_________________________________________________________________
```

Feature Extraction for Train Data

```
: #Now, let us use features from convolutional network for RF
  feature_extractor=VGG_model.predict(x_train)

  features = feature_extractor.reshape(feature_extractor.shape[0], -1)

  X_for_RF = features #This is our X input to RF
```

```
55/55 [==============================] - 526s 8s/step
```

Random Forest and SVM were used as classifiers to train the model and predict the test data.



Random Forest Classifier

```
[18]: #RANDOM FOREST
      from sklearn.ensemble import RandomForestClassifier
      RF_model = RandomForestClassifier(n_estimators = 100, random_state = 42)

      # Train the model on training data
      RF_model.fit(X_for_RF, y_train) #For sklearn no one hot encoding
```

```
:[18]: ▾         RandomForestClassifier
       RandomForestClassifier(random_state=42)
```

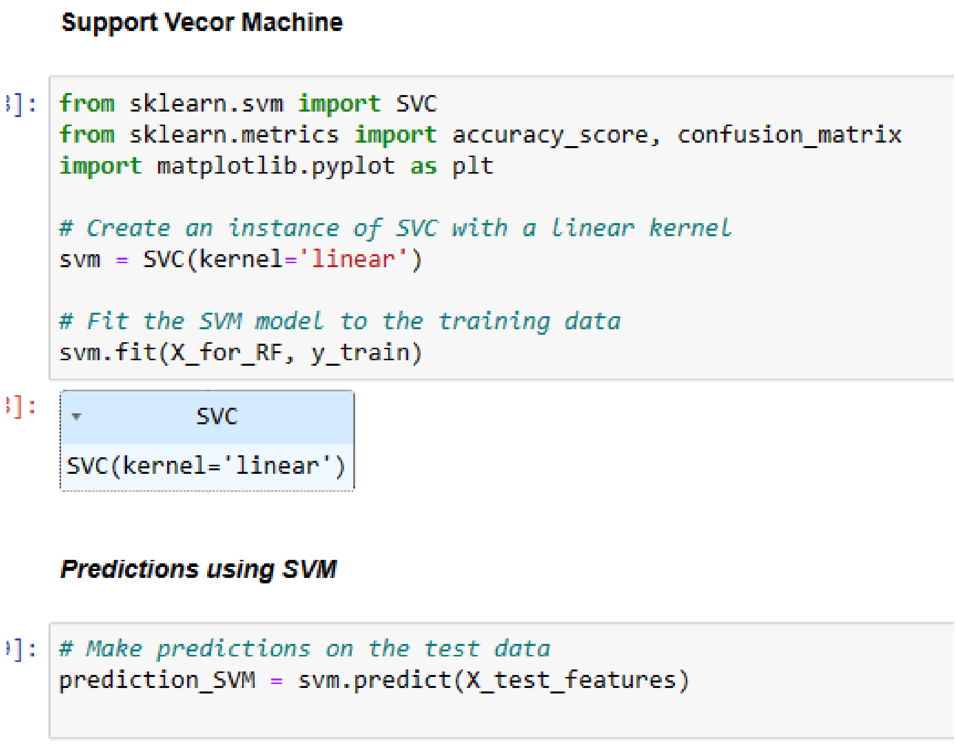
Feature Extraction for Test Data

```
[19]: #Send test data through same feature extractor process
      X_test_feature = VGG_model.predict(x_test)
      X_test_features = X_test_feature.reshape(X_test_feature.shape[0], -1)
```

```
14/14 [==============================] - 114s 8s/step
```

Predictions

```
[20]: #Now predict using the trained RF model.
      prediction_RF = RF_model.predict(X_test_features)
      #Inverse le transform to get original label back.
      #prediction_RF = le.inverse_transform(prediction_RF)
```



Support Vecor Machine

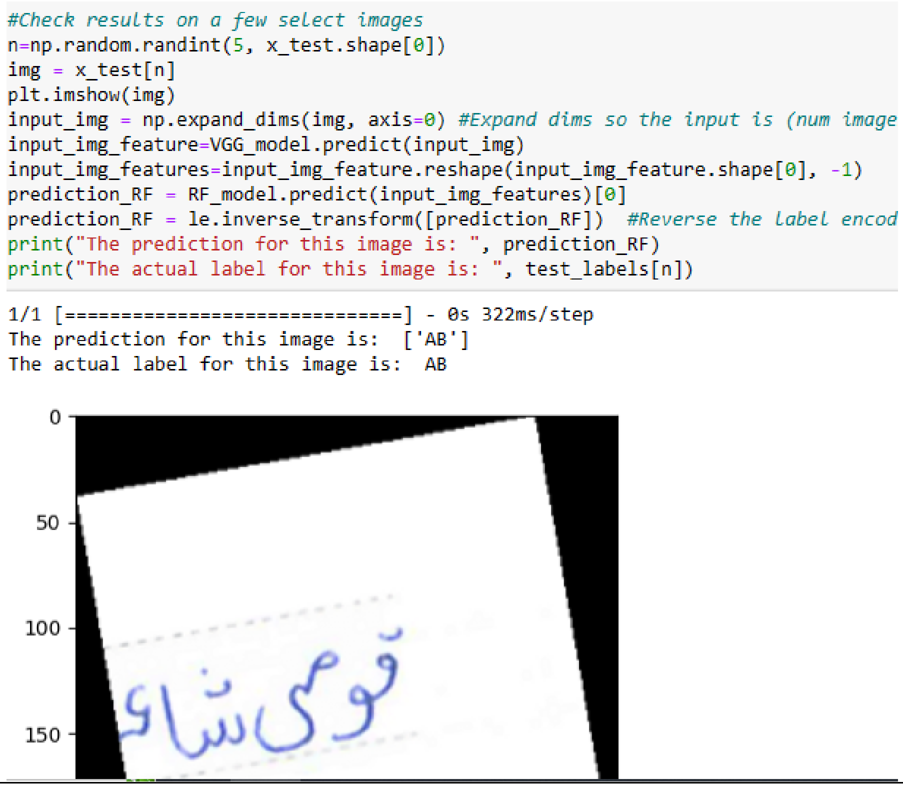
```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt

# Create an instance of SVC with a linear kernel
svm = SVC(kernel='linear')

# Fit the SVM model to the training data
svm.fit(X_for_RF, y_train)
```

```
SVC
SVC(kernel='linear')
```

Predictions using SVM

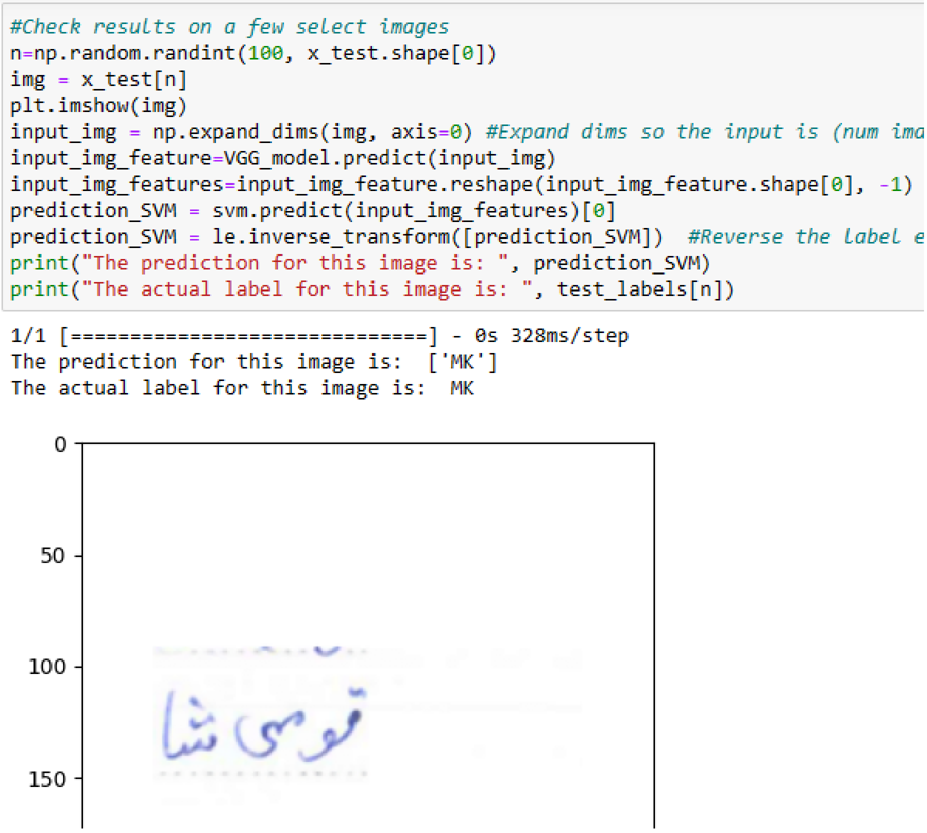
```
# Make predictions on the test data
prediction_SVM = svm.predict(X_test_features)
```

Predictions using random forest model and SVM respectively



```
#Check results on a few select images
n=np.random.randint(5, x_test.shape[0])
img = x_test[n]
plt.imshow(img)
input_img = np.expand_dims(img, axis=0) #Expand dims so the input is (num image
input_img_feature=VGG_model.predict(input_img)
input_img_features=input_img_feature.reshape(input_img_feature.shape[0], -1)
prediction_RF = RF_model.predict(input_img_features)[0]
prediction_RF = le.inverse_transform([prediction_RF])  #Reverse the label encod
print("The prediction for this image is: ", prediction_RF)
print("The actual label for this image is: ", test_labels[n])
```

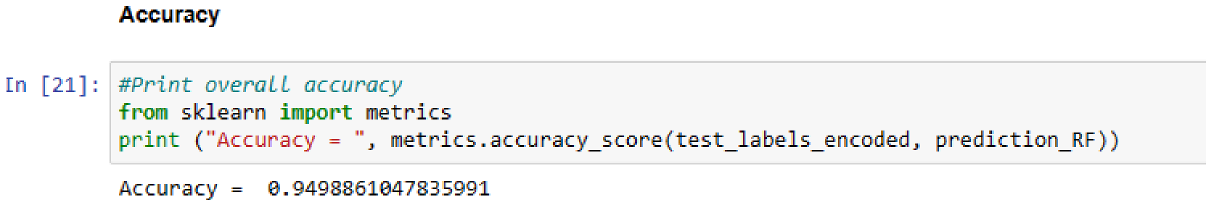
```
1/1 [==============================] - 0s 322ms/step
The prediction for this image is:  ['AB']
The actual label for this image is:  AB
```



```
#Check results on a few select images
n=np.random.randint(100, x_test.shape[0])
img = x_test[n]
plt.imshow(img)
input_img = np.expand_dims(img, axis=0) #Expand dims so the input is (num ima
input_img_feature=VGG_model.predict(input_img)
input_img_features=input_img_feature.reshape(input_img_feature.shape[0], -1)
prediction_SVM = svm.predict(input_img_features)[0]
prediction_SVM = le.inverse_transform([prediction_SVM])  #Reverse the label e
print("The prediction for this image is: ", prediction_SVM)
print("The actual label for this image is: ", test_labels[n])
```

```
1/1 [==============================] - 0s 328ms/step
The prediction for this image is:  ['MK']
The actual label for this image is:  MK
```

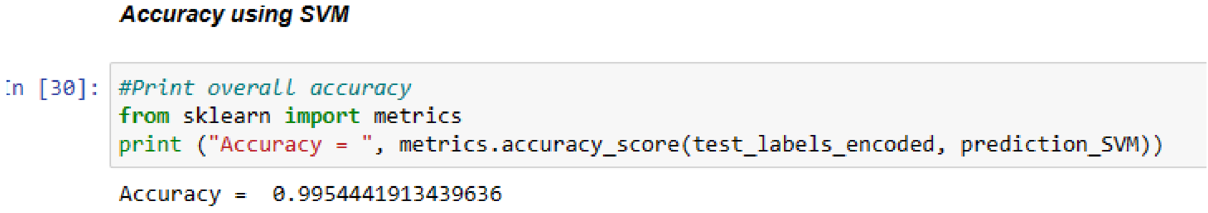
Accuracy using random forest model was almost 94%



Accuracy

```
In [21]: #Print overall accuracy
         from sklearn import metrics
         print ("Accuracy = ", metrics.accuracy_score(test_labels_encoded, prediction_RF))

         Accuracy =  0.9498861047835991
```

Accuracy using SVM was almost 99%



Accuracy using SVM

```
:n [30]: #Print overall accuracy
         from sklearn import metrics
         print ("Accuracy = ", metrics.accuracy_score(test_labels_encoded, prediction_SVM))

         Accuracy =  0.9954441913439636
```

INTERNSHIP EXPERIENCE FEEDBACK

Please give us your valuable feedback

Questions	**Worst**	**Average**	**Good**	**Very good**	**Excellent**
Overall, how would you rate your internship experience?				✓	
How would you rate the quality of supervision and mentorship you received during your internship?					✓
In terms of the tasks and projects you were given during the internship, how would you rate the level of challenge and opportunity for growth?			✓		
How would you rate the level of communication and collaboration among team members during the internship?				✓	
In terms of networking opportunities, how would you rate the ability to connect with professionals in your field during the internship?					✓
How would you rate the overall organization and management of the internship program				✓	
How would you rate the level of feedback and support provided for your personal and professional development during the internship?				✓	
How would you rate the level of diversity and inclusion within the internship program and organization?			✓		
Please provide suggestions. It would be highly appreciated for program improvements in the future.					

Checklist:

Duration completed	

Presentation completed	
Presentation submitted	
Documents submitted	
DIL form filled	

Interns Signature:

Mentor Signature:

Rafay Mustafa

PI, ESCV and Co-PI, NCL Signature: