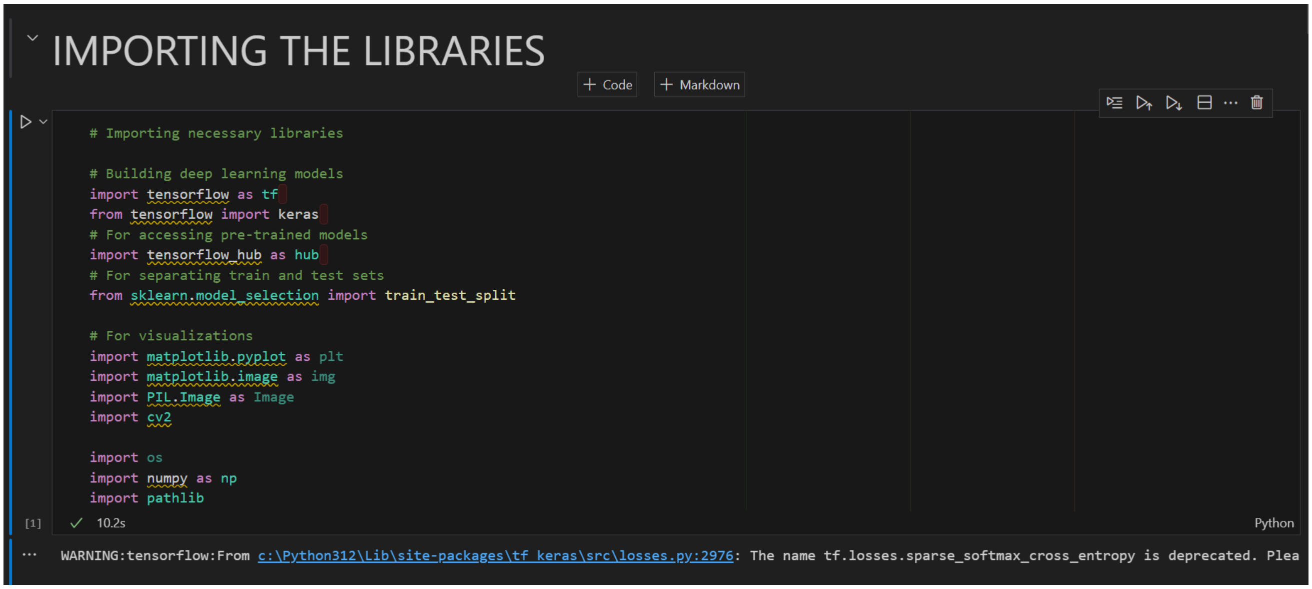
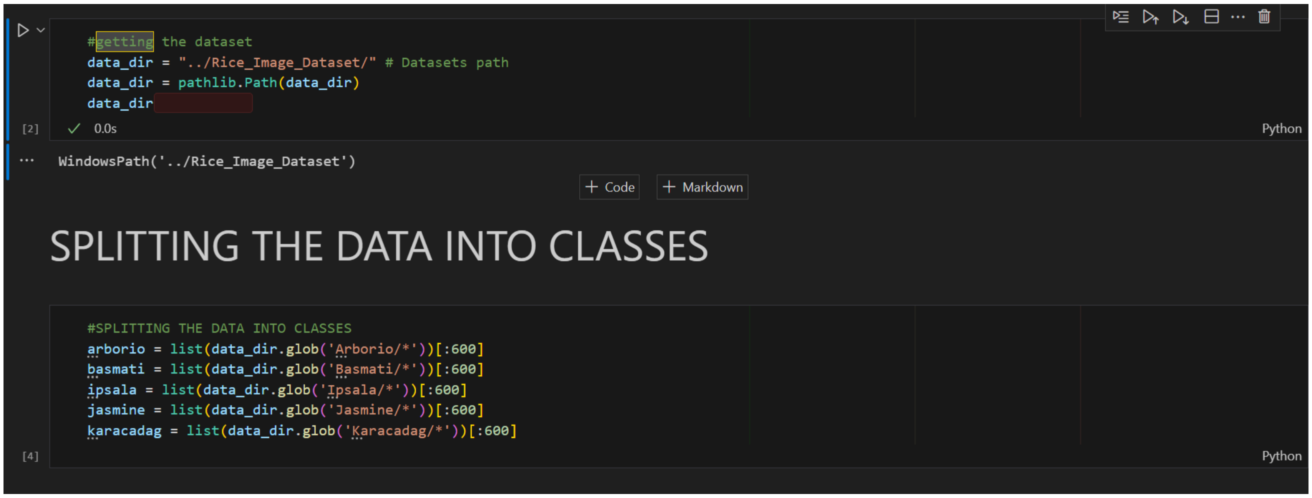
Rice_classification.ipynb



IMPORTING THE LIBRARIES

```
# Importing necessary libraries

# Building deep learning models
import tensorflow as tf
from tensorflow import keras
# For accessing pre-trained models
import tensorflow_hub as hub
# For separating train and test sets
from sklearn.model_selection import train_test_split

# For visualizations
import matplotlib.pyplot as plt
import matplotlib.image as img
import PIL.Image as Image
import cv2

import os
import numpy as np
import pathlib
```

```
WARNING:tensorflow:From c:\Python312\Lib\site-packages\tf_keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Plea
```

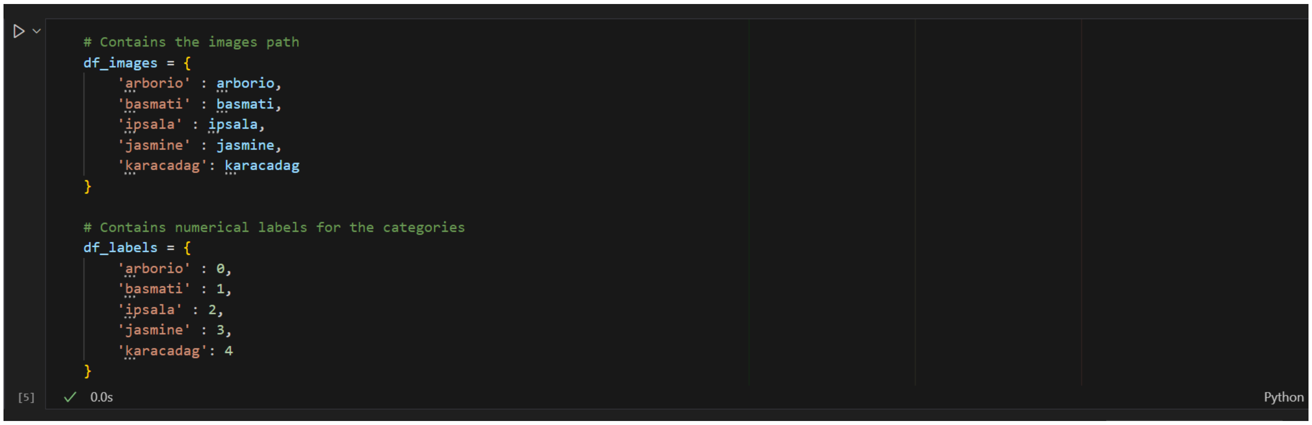


```
#getting the dataset
data_dir = "../Rice_Image_Dataset/" # Datasets path
data_dir = pathlib.Path(data_dir)
data_dir
```

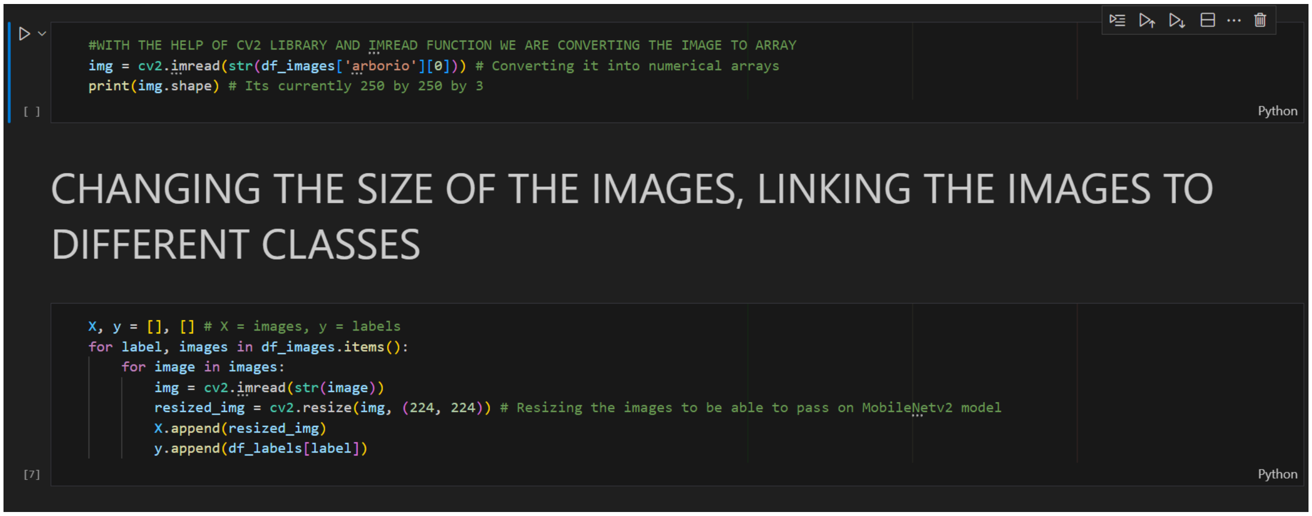
```
WindowsPath('../Rice_Image_Dataset')
```

SPLITTING THE DATA INTO CLASSES

```
#SPLITTING THE DATA INTO CLASSES
arborio = list(data_dir.glob('Arborio/*'))[:600]
basmati = list(data_dir.glob('Basmati/*'))[:600]
ipsala = list(data_dir.glob('Ipsala/*'))[:600]
jasmine = list(data_dir.glob('Jasmine/*'))[:600]
karacadag = list(data_dir.glob('Karacadag/*'))[:600]
```



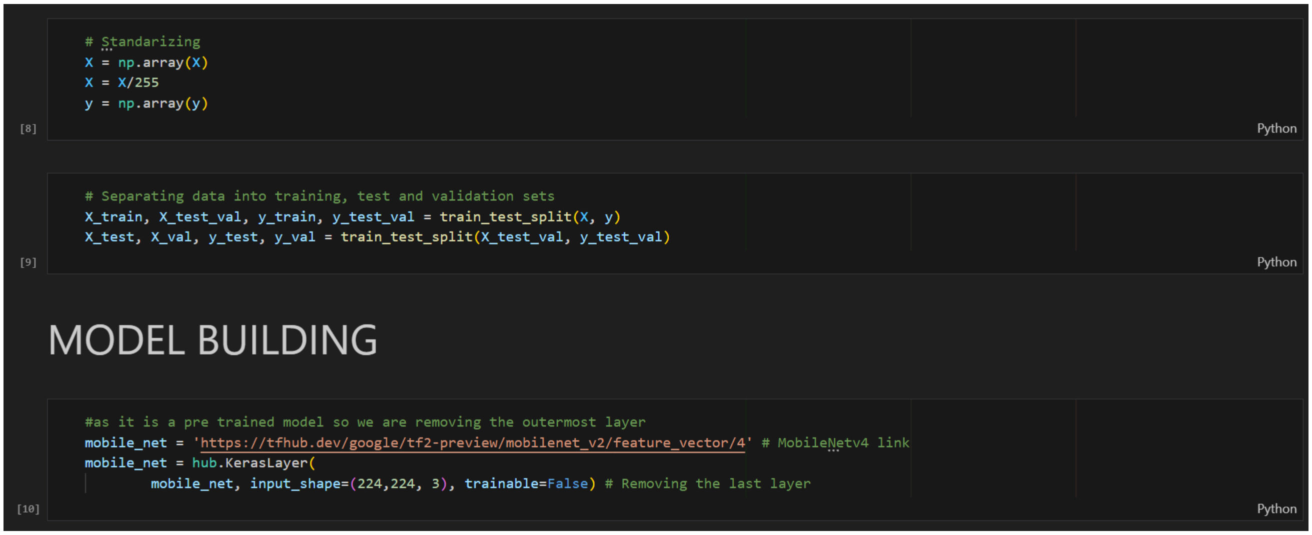
```
# Contains the images path
df_images = {
    'arborio' : arborio,
    'basmati' : basmati,
    'ipsala' : ipsala,
    'jasmine' : jasmine,
    'karacadag': karacadag
}

# Contains numerical labels for the categories
df_labels = {
    'arborio' : 0,
    'basmati' : 1,
    'ipsala' : 2,
    'jasmine' : 3,
    'karacadag': 4
}
```



```python
#WITH THE HELP OF CV2 LIBRARY AND IMREAD FUNCTION WE ARE CONVERTING THE IMAGE TO ARRAY
img = cv2.imread(str(df_images['arborio'][0])) # Converting it into numerical arrays
print(img.shape) # Its currently 250 by 250 by 3
```

CHANGING THE SIZE OF THE IMAGES, LINKING THE IMAGES TO DIFFERENT CLASSES

```python
X, y = [], [] # X = images, y = labels
for label, images in df_images.items():
    for image in images:
        img = cv2.imread(str(image))
        resized_img = cv2.resize(img, (224, 224)) # Resizing the images to be able to pass on MobileNetv2 model
        X.append(resized_img)
        y.append(df_labels[label])
```

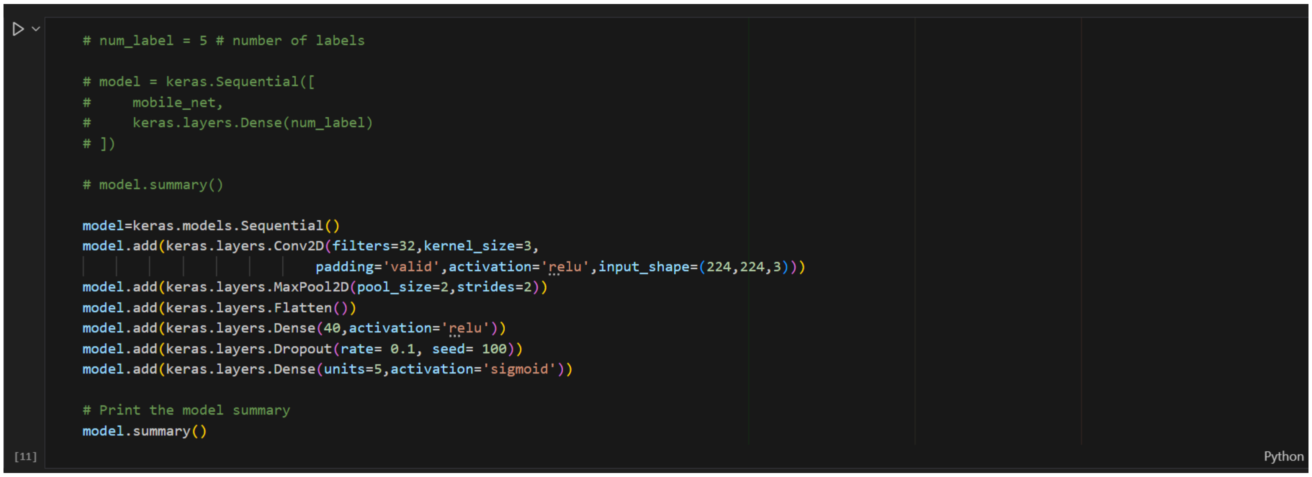


```python
# Standarizing
X = np.array(X)
X = X/255
y = np.array(y)
```

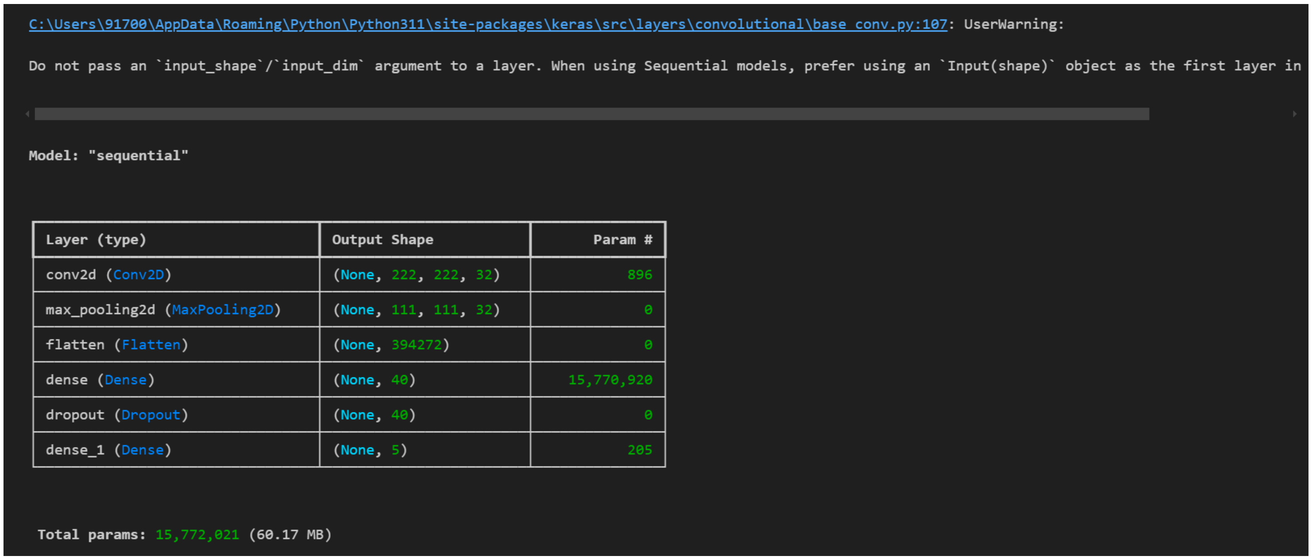
```python
# Separating data into training, test and validation sets
X_train, X_test_val, y_train, y_test_val = train_test_split(X, y)
X_test, X_val, y_test, y_val = train_test_split(X_test_val, y_test_val)
```

MODEL BUILDING

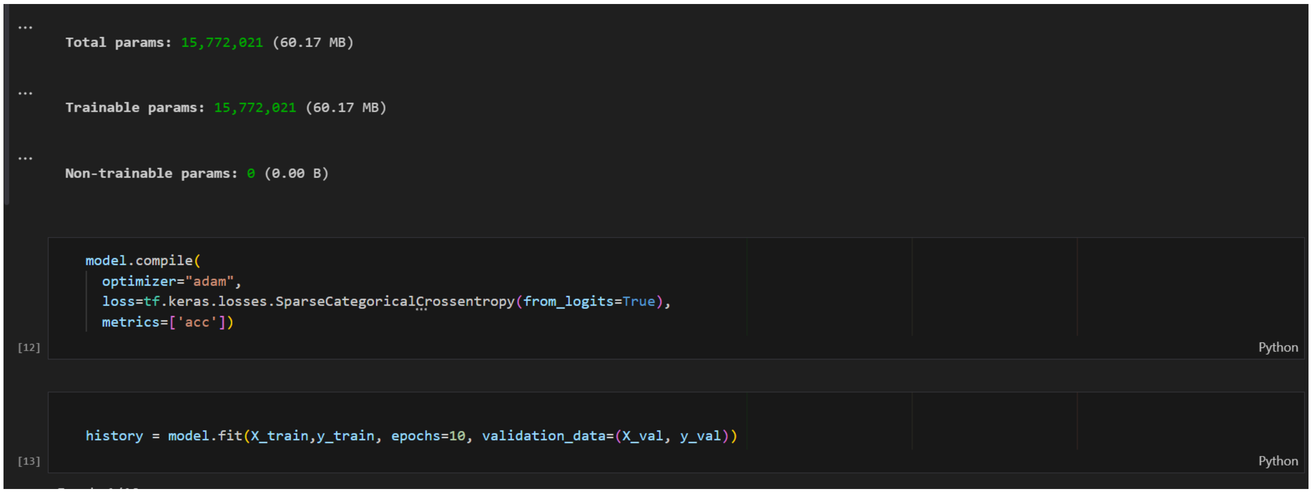
```python
#as it is a pre trained model so we are removing the outermost layer
mobile_net = 'https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4' # MobileNetv4 link
mobile_net = hub.KerasLayer(
        mobile_net, input_shape=(224,224, 3), trainable=False) # Removing the last layer
```



```python
# num_label = 5 # number of labels

# model = keras.Sequential([
#     mobile_net,
#     keras.layers.Dense(num_label)
# ])

# model.summary()

model=keras.models.Sequential()
model.add(keras.layers.Conv2D(filters=32,kernel_size=3,
                            padding='valid',activation='relu',input_shape=(224,224,3)))
model.add(keras.layers.MaxPool2D(pool_size=2,strides=2))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(40,activation='relu'))
model.add(keras.layers.Dropout(rate= 0.1, seed= 100))
model.add(keras.layers.Dense(units=5,activation='sigmoid'))

# Print the model summary
model.summary()
```



```
C:\Users\91700\AppData\Roaming\Python\Python311\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:

Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in

Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                     ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                  │ (None, 222, 222, 32)   │           896 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d (MaxPooling2D)     │ (None, 111, 111, 32)   │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)                │ (None, 394272)         │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                    │ (None, 40)             │    15,770,920 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)                │ (None, 40)             │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                  │ (None, 5)              │           205 │
└──────────────────────────────────┴────────────────────────┴───────────────┘

 Total params: 15,772,021 (60.17 MB)
```



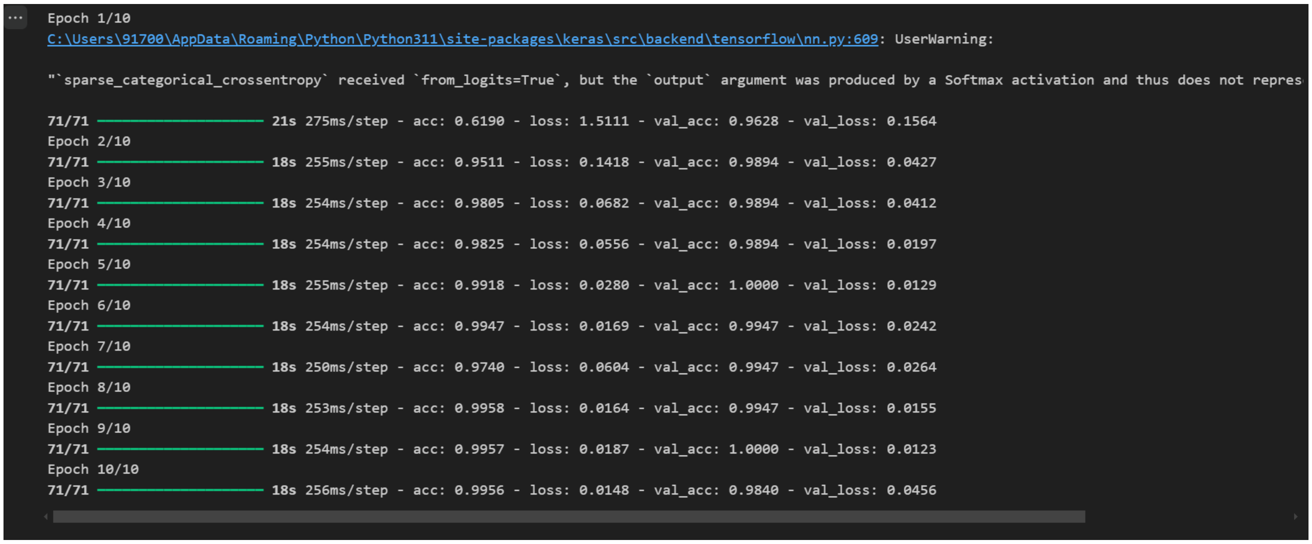
```
 Total params: 15,772,021 (60.17 MB)

 Trainable params: 15,772,021 (60.17 MB)

 Non-trainable params: 0 (0.00 B)
```

```
[12]
model.compile(
    optimizer="adam",
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['acc'])
```

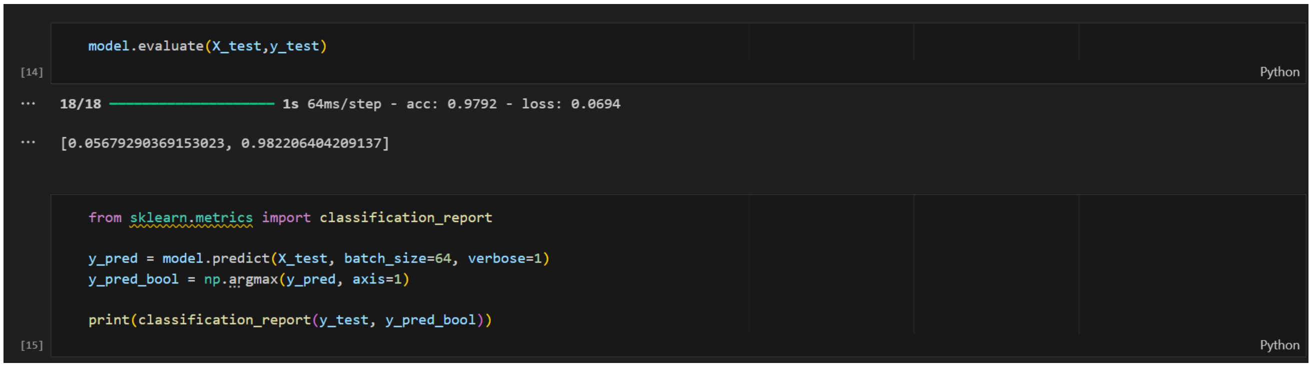
Python

```
[13]
history = model.fit(X_train,y_train, epochs=10, validation_data=(X_val, y_val))
```

Python



```
Epoch 1/10
C:\Users\91700\AppData\Roaming\Python\Python311\site-packages\keras\src\backend\tensorflow\nn.py:609: UserWarning:

"`sparse_categorical_crossentropy` received `from_logits=True`, but the `output` argument was produced by a Softmax activation and thus does not repres

71/71 ━━━━━━━━━━━━━━━━━━━━ 21s 275ms/step - acc: 0.6190 - loss: 1.5111 - val_acc: 0.9628 - val_loss: 0.1564
Epoch 2/10
71/71 ━━━━━━━━━━━━━━━━━━━━ 18s 255ms/step - acc: 0.9511 - loss: 0.1418 - val_acc: 0.9894 - val_loss: 0.0427
Epoch 3/10
71/71 ━━━━━━━━━━━━━━━━━━━━ 18s 254ms/step - acc: 0.9805 - loss: 0.0682 - val_acc: 0.9894 - val_loss: 0.0412
Epoch 4/10
71/71 ━━━━━━━━━━━━━━━━━━━━ 18s 254ms/step - acc: 0.9825 - loss: 0.0556 - val_acc: 0.9894 - val_loss: 0.0197
Epoch 5/10
71/71 ━━━━━━━━━━━━━━━━━━━━ 18s 255ms/step - acc: 0.9918 - loss: 0.0280 - val_acc: 1.0000 - val_loss: 0.0129
Epoch 6/10
71/71 ━━━━━━━━━━━━━━━━━━━━ 18s 254ms/step - acc: 0.9947 - loss: 0.0169 - val_acc: 0.9947 - val_loss: 0.0242
Epoch 7/10
71/71 ━━━━━━━━━━━━━━━━━━━━ 18s 250ms/step - acc: 0.9740 - loss: 0.0604 - val_acc: 0.9947 - val_loss: 0.0264
Epoch 8/10
71/71 ━━━━━━━━━━━━━━━━━━━━ 18s 253ms/step - acc: 0.9958 - loss: 0.0164 - val_acc: 0.9947 - val_loss: 0.0155
Epoch 9/10
71/71 ━━━━━━━━━━━━━━━━━━━━ 18s 254ms/step - acc: 0.9957 - loss: 0.0187 - val_acc: 1.0000 - val_loss: 0.0123
Epoch 10/10
71/71 ━━━━━━━━━━━━━━━━━━━━ 18s 256ms/step - acc: 0.9956 - loss: 0.0148 - val_acc: 0.9840 - val_loss: 0.0456
```

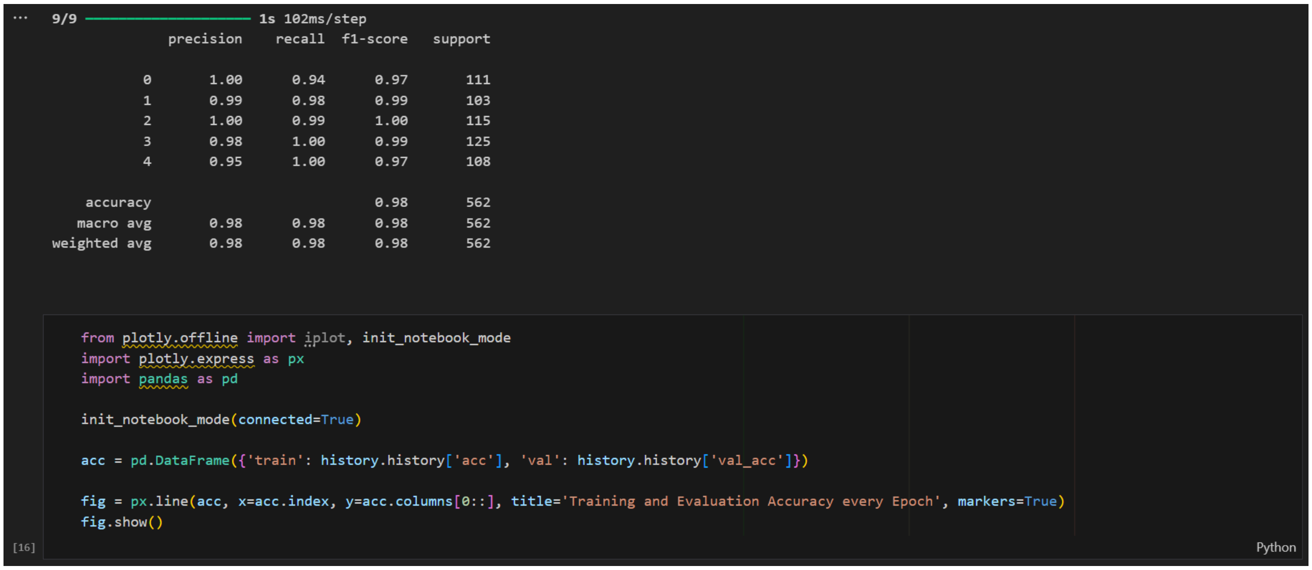


```
model.evaluate(X_test,y_test)
```
[14] Python

```
18/18 ━━━━━━━━━━━━━━━━━━━━ 1s 64ms/step - acc: 0.9792 - loss: 0.0694
```

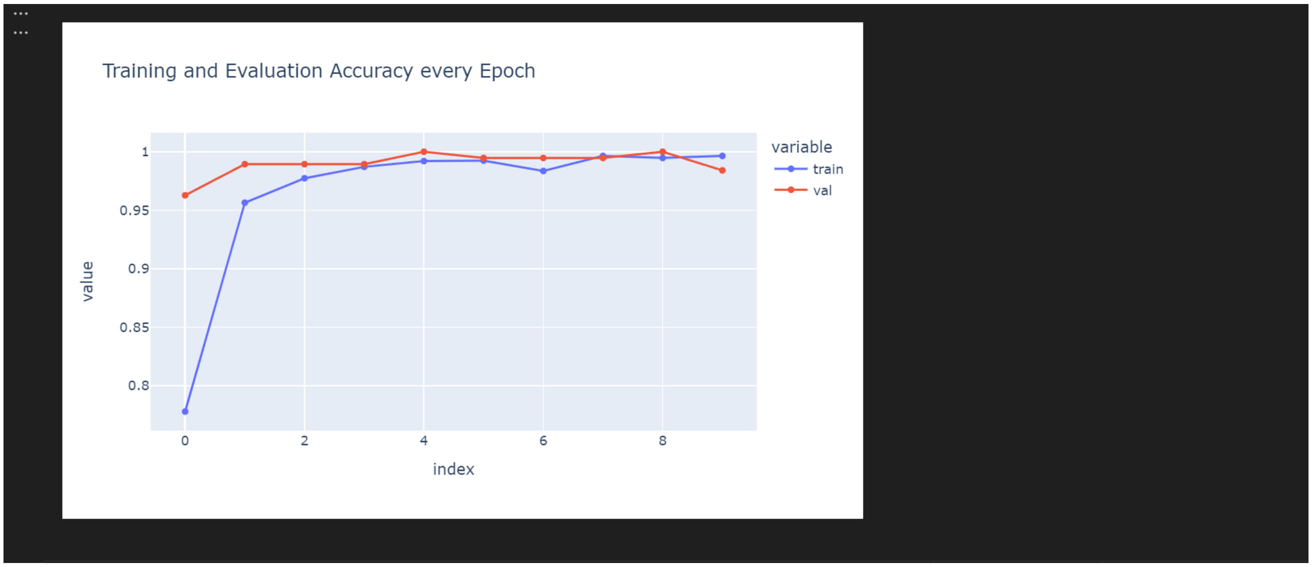
```
[0.05679290369153023, 0.982206404209137]
```

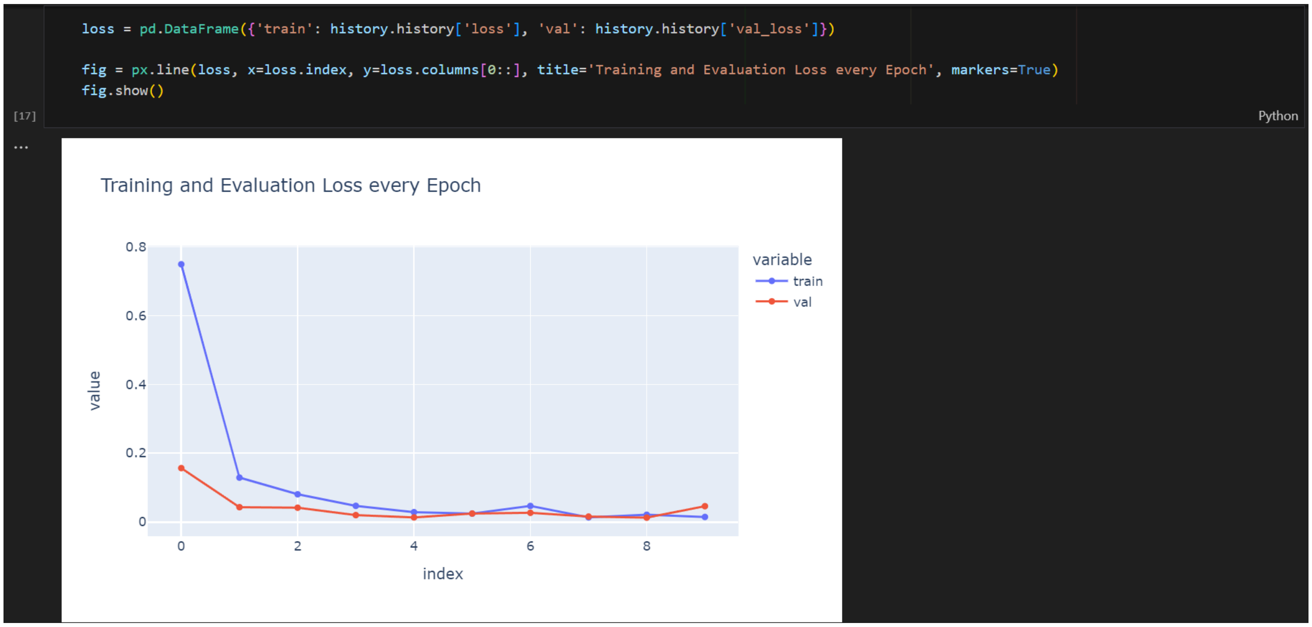
```
from sklearn.metrics import classification_report

y_pred = model.predict(X_test, batch_size=64, verbose=1)
y_pred_bool = np.argmax(y_pred, axis=1)

print(classification_report(y_test, y_pred_bool))
```
[15] Python



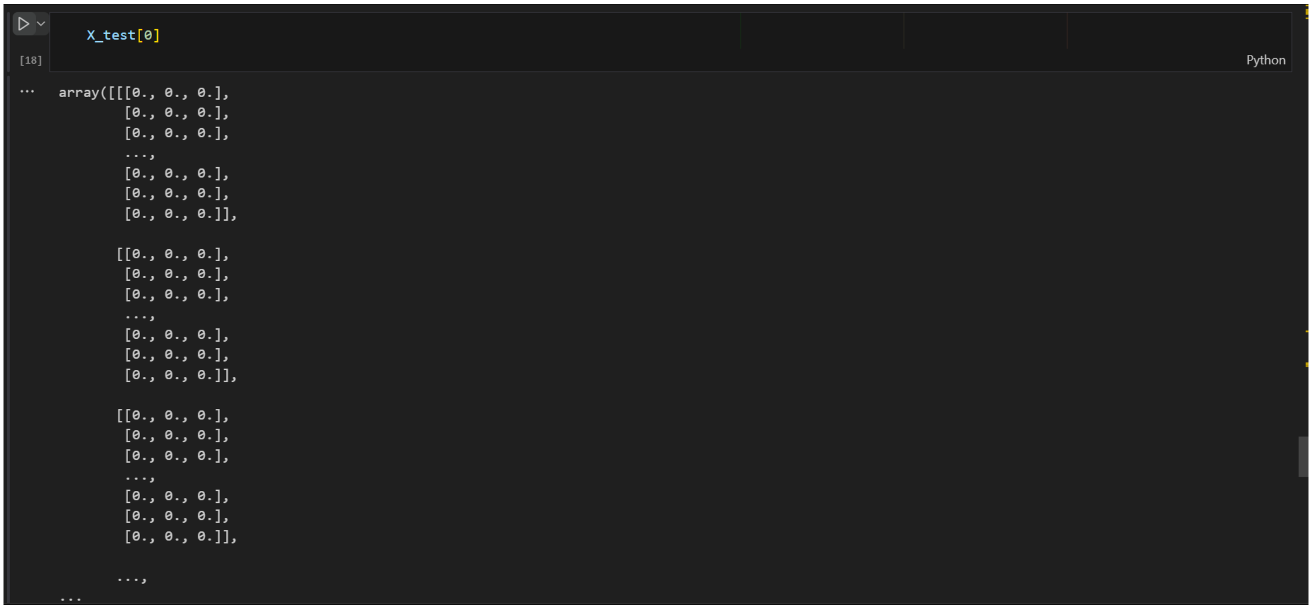
```
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 102ms/step
              precision    recall  f1-score   support

           0       1.00      0.94      0.97       111
           1       0.99      0.98      0.99       103
           2       1.00      0.99      1.00       115
           3       0.98      1.00      0.99       125
           4       0.95      1.00      0.97       108

    accuracy                           0.98       562
   macro avg       0.98      0.98      0.98       562
weighted avg       0.98      0.98      0.98       562
```

```
from plotly.offline import iplot, init_notebook_mode
import plotly.express as px
import pandas as pd

init_notebook_mode(connected=True)

acc = pd.DataFrame({'train': history.history['acc'], 'val': history.history['val_acc']})

fig = px.line(acc, x=acc.index, y=acc.columns[0::], title='Training and Evaluation Accuracy every Epoch', markers=True)
fig.show()
```
[16] Python



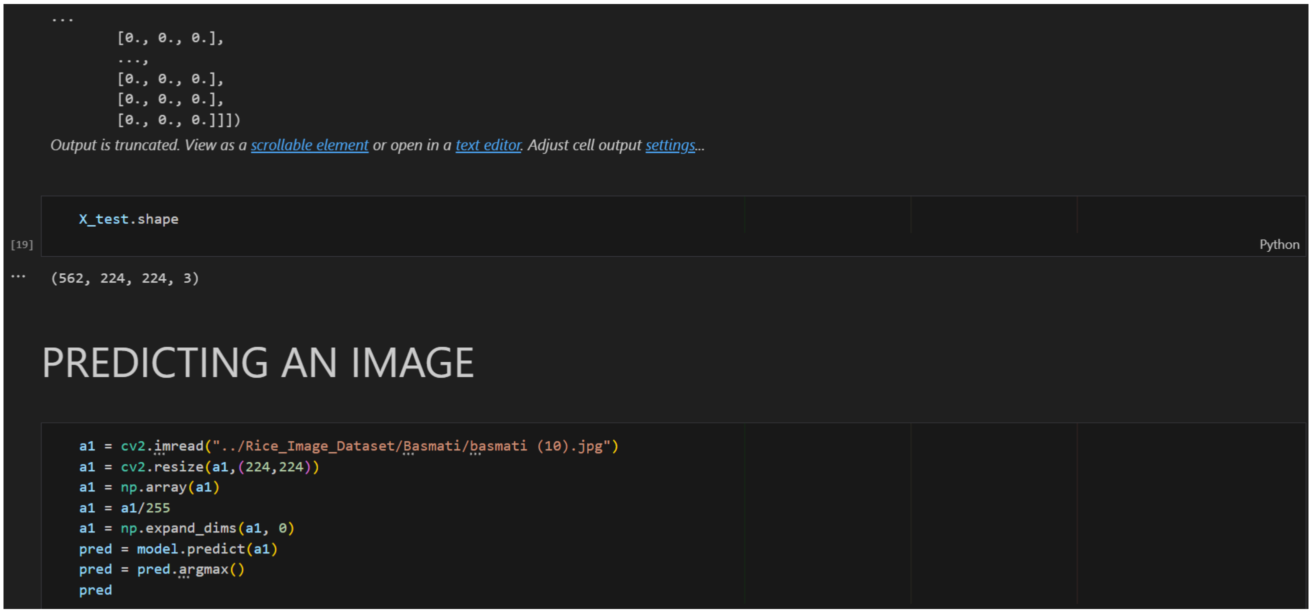


```
loss = pd.DataFrame({'train': history.history['loss'], 'val': history.history['val_loss']})

fig = px.line(loss, x=loss.index, y=loss.columns[0::], title='Training and Evaluation Loss every Epoch', markers=True)
fig.show()
```



```
X_test[0]
```

```
array([[[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.],
        ...,
        [0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]],

       [[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.],
        ...,
        [0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]],

       [[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.],
        ...,
        [0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]],

       ...,
...
```



```
...
        [0., 0., 0.],
        ...,
        [0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]]])
```

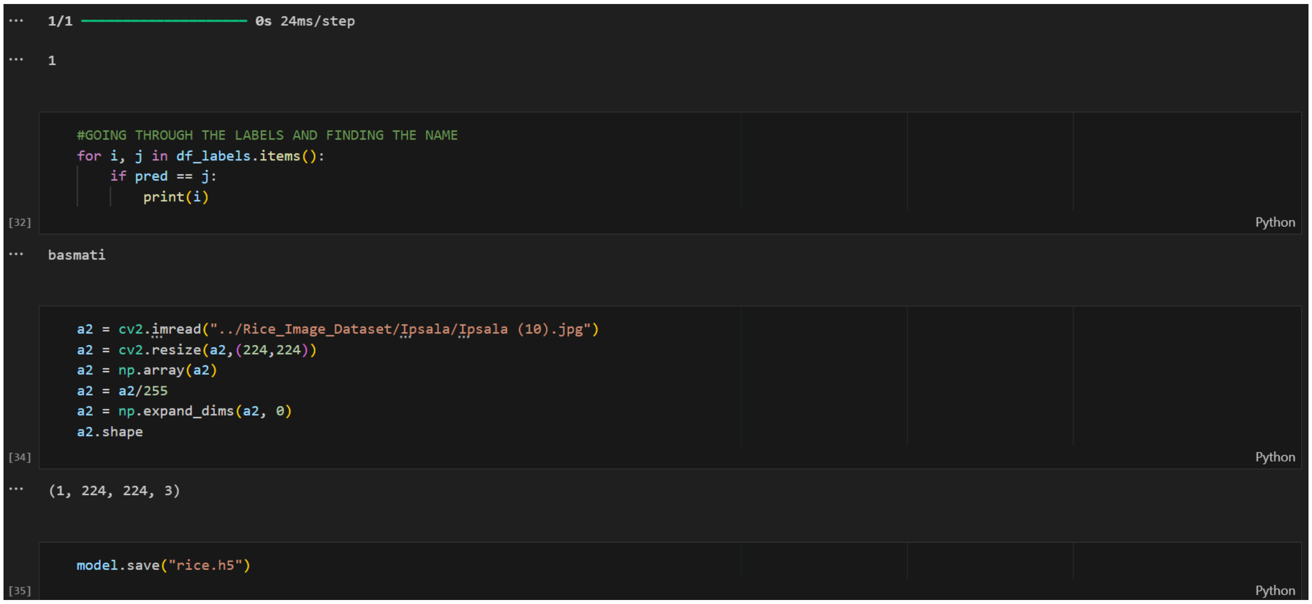
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
X_test.shape
```

```
(562, 224, 224, 3)
```

PREDICTING AN IMAGE

```
a1 = cv2.imread("../Rice_Image_Dataset/Basmati/basmati (10).jpg")
a1 = cv2.resize(a1,(224,224))
a1 = np.array(a1)
a1 = a1/255
a1 = np.expand_dims(a1, 0)
pred = model.predict(a1)
pred = pred.argmax()
pred
```



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
```

```
1
```

```python
#GOING THROUGH THE LABELS AND FINDING THE NAME
for i, j in df_labels.items():
    if pred == j:
        print(i)
```

[32] Python

```
basmati
```

```python
a2 = cv2.imread("../Rice_Image_Dataset/Ipsala/Ipsala (10).jpg")
a2 = cv2.resize(a2,(224,224))
a2 = np.array(a2)
a2 = a2/255
a2 = np.expand_dims(a2, 0)
a2.shape
```

[34] Python

```
(1, 224, 224, 3)
```

```python
model.save("rice.h5")
```

[35] Python