IMAGE CLASSIFICATION ON RICE IMAGE DATASET

IMPORTING PYTHON LIBRARIES

In [17]:
```
import tensorflow as tf
import keras
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Activation,Dropout
from keras.models import Model, Sequential

from keras.metrics import categorical_crossentropy
from sklearn.metrics import confusion_matrix,classification_report
import imageio
import matplotlib.image as img

import os
import pathlib

from tensorflow.keras.applications import imagenet_utils

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
sns.set_style('darkgrid')
import itertools
```

In [18]:
```
path = pathlib.Path("/Users/siddivinayakayandakuditi/Desktop/Rice_Image_Dataset")
```

In [19]:
```
arborio = list(path.glob('Arborio/*'))[:1000]
basmati = list(path.glob('Basmati/*'))[:1000]
ipsala = list(path.glob('Ipsala/*'))[:1000]
jasmine = list(path.glob('Jasmine/*'))[:1000]
karacadag = list(path.glob('Karacadag/*'))[:1000]
```

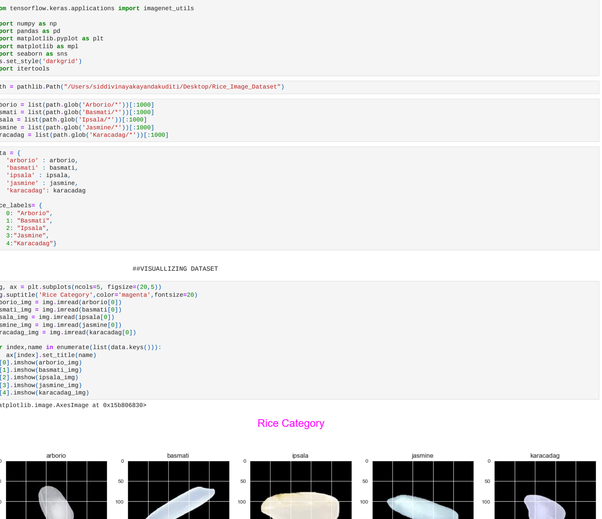
In [20]:
```
data = {
    'arborio' : arborio,
    'basmati' : basmati,
    'ipsala' : ipsala,
    'jasmine' : jasmine,
    'karacadag': karacadag
}
rice_labels= {
    0: "Arborio",
    1: "Basmati",
    2: "Ipsala",
    3:"Jasmine",
    4:"Karacadag"}
```

##VISUALLIZING DATASET

In [21]:
```
fig, ax = plt.subplots(ncols=5, figsize=(20,5))
fig.suptitle('Rice Category',color='magenta',fontsize=20)
arborio_img = img.imread(arborio[0])
basmati_img = img.imread(basmati[0])
ipsala_img = img.imread(ipsala[0])
jasmine_img = img.imread(jasmine[0])
karacadag_img = img.imread(karacadag[0])

for index,name in enumerate(list(data.keys())):
    ax[index].set_title(name)
ax[0].imshow(arborio_img)
ax[1].imshow(basmati_img)
ax[2].imshow(ipsala_img)
ax[3].imshow(jasmine_img)
ax[4].imshow(karacadag_img)
```

Out[21]:
```
<matplotlib.image.AxesImage at 0x15b806830>
```



##TRAINING AND TESTING DATASET

In [22]:
```
train_gen=ImageDataGenerator(rescale=1./255,validation_split=0.2)
train_data=train_gen.flow_from_directory("/Users/siddivinayakayandakuditi/Desktop/Rice_Image_Dataset",target_size=(224,224),batch_size=32,class_mode='categorical',shuff
test_data=train_gen.flow_from_directory("/Users/siddivinayakayandakuditi/Desktop/Rice_Image_Dataset",target_size=(224,224),batch_size=1,shuffle=False,subset='validation
```

```
Found 60000 images belonging to 5 classes.
Found 15000 images belonging to 5 classes.
```

In [23]:
```
cnn=keras.models.Sequential()
cnn.add(keras.layers.Conv2D(filters=32,kernel_size=3,
                            padding='valid',activation='relu',input_shape=(224,224,3)))
cnn.add(keras.layers.MaxPool2D(pool_size=2,strides=2))
cnn.add(keras.layers.Flatten())
cnn.add(keras.layers.Dense(40,activation='relu'))
cnn.add(keras.layers.Dropout(rate= 0.1, seed= 100))
cnn.add(keras.layers.Dense(units=5,activation='sigmoid'))
cnn.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 222, 222, 32)      896

 max_pooling2d_1 (MaxPooling  (None, 111, 111, 32)     0
 2D)

 flatten_1 (Flatten)         (None, 394272)            0

 dense_2 (Dense)             (None, 40)                15770920

 dropout_1 (Dropout)         (None, 40)                0

 dense_3 (Dense)             (None, 5)                 205

=================================================================
Total params: 15,772,021
Trainable params: 15,772,021
Non-trainable params: 0
_________________________________________________________________
```

In [24]:
```
cnn.compile(optimizer='adam',metrics=['accuracy'],loss='categorical_crossentropy')
```

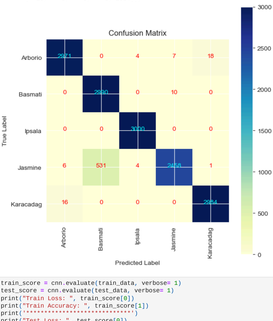
In [25]:
```
cnn.fit(train_data,epochs=2,validation_data=test_data,shuffle=True)
```

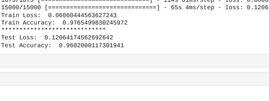
```
Epoch 1/2
1875/1875 [==============================] - 414s 221ms/step - loss: 0.1585 - accuracy: 0.9455 - val_loss: 0.0707 - val_accuracy: 0.9758
Epoch 2/2
1875/1875 [==============================] - 419s 224ms/step - loss: 0.0629 - accuracy: 0.9788 - val_loss: 0.1206 - val_accuracy: 0.9602
```

Out[25]:
```
<keras.callbacks.History at 0x2b4197e50>
```

In [42]:
```
def plot(c):
    plt.plot(c.history.history['accuracy'],marker='o',color='black',markersize=10)
    plt.plot(c.history.history['val_accuracy'],marker='*',color='red',markersize=10)
    plt.title('accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'])
    plt.show()
    plt.plot(c.history.history['loss'],marker='o',color='black',markersize=10)
    plt.plot(c.history.history['val_loss'],marker='*',color='red',markersize=10)
    plt.title('loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'])
    plt.show()
```

In [43]:
```
plot(cnn)
```





In [44]:
```
import warnings
warnings.filterwarnings('ignore')    #IGNORING WARNINGS


test_steps_per_epoch = np.math.ceil(test_data.samples / test_data.batch_size)

predictions = cnn.predict_generator(test_data, steps=test_steps_per_epoch)
# Get most likely class
predicted_classes = np.argmax(predictions, axis=1)
```

In [45]:
```
true_classes = test_data.classes
class_labels = list(test_data.class_indices.keys())
report = classification_report(true_classes, predicted_classes, target_names=class_labels)
print(report)
```

```
              precision    recall  f1-score   support

      Arborio       0.99      0.99      0.99      3000
      Basmati       0.85      1.00      0.92      3000
       Ipsala       1.00      1.00      1.00      3000
      Jasmine       0.99      0.82      0.90      3000
   Karacadag       0.99      0.99      0.99      3000

    accuracy                           0.96     15000
   macro avg       0.97      0.96      0.96     15000
weighted avg       0.97      0.96      0.96     15000
```

##CREATING CONFUSION MATRIX

In [47]:
```
cm = confusion_matrix(test_data.classes, predicted_classes)
di=test_data.class_indices
classes = list(di.keys())
cmap= plt.cm.YlGnBu
plt.figure(figsize= (6, 6))
plt.imshow(cm, interpolation= 'nearest', cmap= cmap)
plt.title('Confusion Matrix')
plt.colorbar(shrink=True)
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation= 90)
plt.yticks(tick_marks, classes)
thresh = cm.max() / 2.
for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, cm[i, j], horizontalalignment= 'center', color= 'aqua' if cm[i, j] > thresh else 'red')
plt.tight_layout()
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
cm
```

Out[47]:
```
array([[2971,    0,    4,    7,   18],
       [   0, 2998,    0,    0,    0],
       [   0,    0, 3000,    0,    0],
       [   6,  531,    4, 2458,    1],
       [  16,    0,    0,    0, 2984]])
```


In [48]:
```
train_score = cnn.evaluate(train_data, verbose= 1)
test_score = cnn.evaluate(test_data, verbose= 1)
print("Train Loss: ", train_score[0])
print("Train Accuracy: ", train_score[1])
print('*'*40)
print("Test Loss: ", test_score[0])
print("Test Accuracy: ", test_score[1])
```

```
1875/1875 [==============================] - 114s 61ms/step - loss: 0.0606 - accuracy: 0.9765
15000/15000 [==============================] - 65s 4ms/step - loss: 0.1206 - accuracy: 0.9602
Train Loss:  0.06060444563627243
Train Accuracy:  0.9765499830245972
****************************************
Test Loss:  0.12064174562692642
Test Accuracy:  0.9602000117301941
```

In [ ]:

In [ ]: