```
# 1. IMPORT LIBRARIES
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.preprocessing import image_dataset_from_directory
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import GlobalAveragePooling2D, Dropout, Dense
from tensorflow.keras.models import Sequential
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import os
import pathlib
# 2. LOAD & PREPROCESS DATA
dataset_path = "/content/rice_dataset"  # Update to your path
batch_size = 32
img_size = (224, 224)
train_ds = image_dataset_from_directory(
    dataset_path,
    validation_split=0.2,
    subset="training",
    seed=123,
    image_size=img_size,
    batch_size=batch_size
)
val_ds = image_dataset_from_directory(
    dataset_path,
    validation_split=0.2,
    subset="validation",
    seed=123,
    image_size=img_size,
```

```
  batch_size=batch_size
)
class_names = train_ds.class_names
AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.prefetch(buffer_size=AUTOTUNE)
# 3. BUILD TRANSFER LEARNING MODEL (MobileNetV2)
base_model = MobileNetV2(input_shape=(224, 224, 3),
                         include_top=False,
                         weights='imagenet')
base_model.trainable = False
model = Sequential([
   base_model,
   GlobalAveragePooling2D(),
   Dropout(0.3),
   Dense(128, activation='relu'),
   Dropout(0.3),
   Dense(len(class_names), activation='softmax')
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.summary()
# 4. TRAIN MODEL
epochs = 10
history = model.fit(train_ds, validation_data=val_ds, epochs=epochs)
# 5. VISUALIZE PERFORMANCE
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
```

```
val_loss = history.history['val_loss']
epochs_range = range(epochs)
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')
plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
# 6. EVALUATE MODEL
y_true = []
y_pred = []
for images, labels in val_ds:
    preds = model.predict(images)
    y_true.extend(labels.numpy())
    y_pred.extend(np.argmax(preds, axis=1))
# Confusion Matrix
cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

```
# Classification Report
print("Classification Report:\n", classification_report(y_true, y_pred,
target_names=class_names))
# 7. PREDICT SAMPLE IMAGE
from tensorflow.keras.preprocessing import image
img_path = '/content/sample_rice.jpg'  # Replace with your own image
img = image.load_img(img_path, target_size=(224, 224))
img_array = image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)  # Create batch axis
predictions = model.predict(img_array)
predicted_class = class_names[np.argmax(predictions[0])]
confidence = 100 * np.max(predictions[0])
print(f"Predicted: {predicted_class} ({confidence:.2f}% confidence)")
```