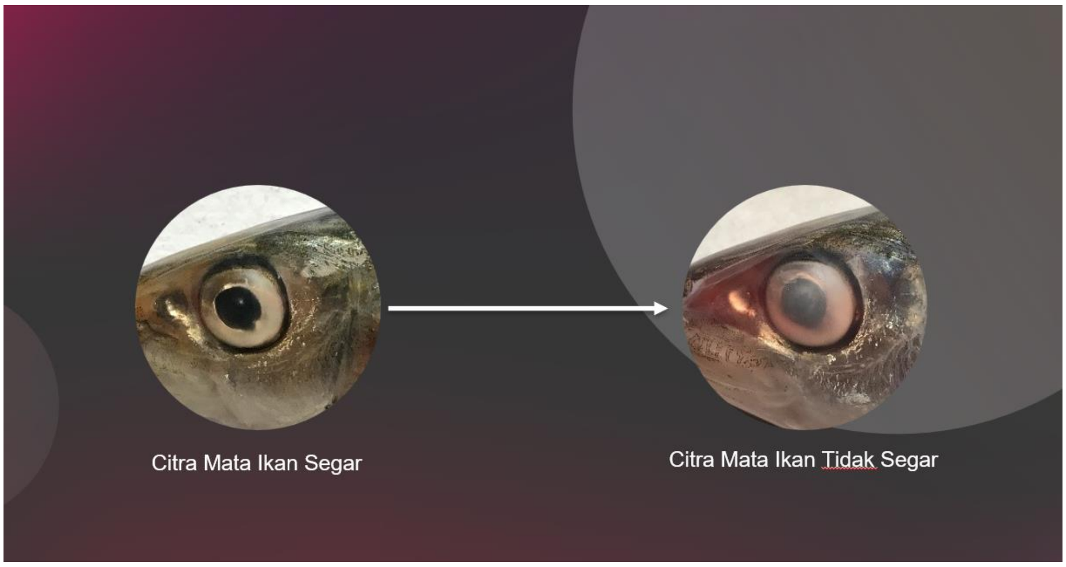
Lampiran

Lampiran 1 : Dataset



Lampiran 2 : Source code

```
##Import Google Drive
    from google.colab import drive
    drive.mount('/content/drive')
    data_dir =
    '/content/drive/MyDrive/DATASET_MATA_IKAN_2/MATA_IKAN_1.1/'

    ##Import Library
    import numpy as np
    import os
    from PIL import Image
    import tensorflow as tf
    from tensorflow import keras
    from tensorflow.keras import layers
    from tensorflow.keras.models import Model
    from sklearn.model_selection import KFold, StratifiedKFold
    import pandas as pd
    import shutil
    from tensorflow.keras.preprocessing.image import
    ImageDataGenerator
    from tensorflow.python.keras import Input
    from tensorflow.keras.layers import  MaxPooling2D, Dense,
    Dropout, Conv2D, MaxPool2D, Flatten, Reshape, BatchNormalization,
    Add, GlobalAveragePooling2D, ZeroPadding2D, AveragePooling2D,
    Activation
    from tensorflow.keras.activations import relu, softmax, swish,
    sigmoid
    from tensorflow.keras.losses import
    SparseCategoricalCrossentropy, CategoricalCrossentropy
    from tensorflow.keras.optimizers import Adam, SGD,RMSprop
    from tensorflow.keras.regularizers import l2
    from tensorflow.keras.metrics import
    Precision,SensitivityAtSpecificity, SpecificityAtSensitivity,
    Metric, TruePositives,FalseNegatives, Recall
    from tensorflow.keras.models import load_model
    from tensorflow.keras.applications import ResNet50V2, ResNet101
    from keras.models import Sequential
    from keras.utils.vis_utils import plot_model
    from tensorflow.python.keras.utils import metrics_utils
    from keras.preprocessing import image
    from keras.preprocessing.image import img_to_array
    from google.colab import files
    from keras.applications.vgg16 import preprocess_input
    from sklearn.utils import shuffle, resample
    from pathlib import Path
    import seaborn as sns
    import matplotlib.pyplot as plt
    from tensorflow.keras.callbacks import TensorBoard,EarlyStopping,
    ModelCheckpoint,CSVLogger, ReduceLROnPlateau,
    LearningRateScheduler
    import datetime
    from sklearn.metrics import accuracy_score, classification_report
    from keras import backend as K
    import random
    from sklearn.model_selection import
    train_test_split,  StratifiedShuffleSplit
    !pip install pycm==3.1
    %load_ext tensorboard
    from pycm import *

    ##Create DataFrame
    df = None
    for label in os.listdir(data_dir):
      data = []
      for img_filename in os.listdir(data_dir + '/' + label):
        filename, ext = os.path.splitext(img_filename)
        label_folder = os.path.join(data_dir, label)
        src_img_filepath = os.path.join(label_folder, img_filename)
        data.append(src_img_filepath)
      new_df = pd.DataFrame(data, columns=['path',])
      new_df['label'] = label
      df = pd.concat([df, new_df])
    df

    ##Split Data
    df_train_val, df_test = train_test_split(df, random_state=666,
    test_size=0.1 )
    df_train_val.to_csv('/content/drive/MyDrive/
    DATASET_MATA_IKAN_2/train_label.csv', index=False)
    df_test.to_csv('/content/drive/MyDrive/
    DATASET_MATA_IKAN_2/test_label.csv', index=False)
    image_size=(224, 224)

    df_test
    df_train_val


    # VGG-16 With (input 224 Filter 32 )

    model = keras.Sequential()


    # Creating first block- (2 Convolution + 1 Max pool)
    model.add(Conv2D(filters= 32, kernel_size= (3,3), strides= (1,1),
     padding='same', input_shape= (224, 224, 3), activation= 'relu'))
    # model.add(BatchNormalization())
    model.add(Conv2D(filters= 32, kernel_size= (3,3), strides= (1,1),
     padding='same', activation= 'relu'))
    # model.add(BatchNormalization())
    model.add(MaxPool2D(pool_size= (2,2), strides=(2,2)))
```

```
# Creating second block- (2 Convolution + 1 Max pool)
model.add(Conv2D(filters= 64, kernel_size= (3,3), strides= (1,1),
 padding='same', activation= 'relu'))
# model.add(BatchNormalization())
model.add(Conv2D(filters= 64, kernel_size= (3,3), strides= (1,1),
 padding='same', activation= 'relu'))
# model.add(BatchNormalization())
model.add(MaxPool2D(pool_size= (2,2), strides=(2,2)))

# Creating third block- (3 Convolution + 1 Max pool)
model.add(Conv2D(filters= 128, kernel_size= (3,3), strides= (1,1)
, padding='same', activation= 'relu'))
# model.add(BatchNormalization())
model.add(Conv2D(filters= 128, kernel_size= (3,3), strides= (1,1)
, padding='same', activation= 'relu'))
# model.add(BatchNormalization())
model.add(Conv2D(filters= 128, kernel_size= (3,3), strides= (1,1)
, padding='same', activation= 'relu'))
# model.add(BatchNormalization())
model.add(MaxPool2D(pool_size= (2,2), strides=(2,2)))

# Creating fourth block- (3 Convolution + 1 Max pool)
model.add(Conv2D(filters= 256, kernel_size= (3,3), strides= (1,1)
, padding='same', activation= 'relu'))
# model.add(BatchNormalization())
model.add(Conv2D(filters= 256, kernel_size= (3,3), strides= (1,1)
, padding='same', activation= 'relu'))
# model.add(BatchNormalization())
model.add(Conv2D(filters= 256, kernel_size= (3,3), strides= (1,1)
, padding='same', activation= 'relu'))
# model.add(BatchNormalization())
model.add(MaxPool2D(pool_size= (2,2), strides=(2,2)))

# Creating fifth block- (3 Convolution + 1 Max pool)
model.add(Conv2D(filters= 512, kernel_size= (3,3), strides= (1,1)
, padding='same', activation= 'relu'))
# model.add(BatchNormalization())
model.add(Conv2D(filters= 512, kernel_size= (3,3), strides= (1,1)
, padding='same', activation= 'relu'))
# model.add(BatchNormalization())
model.add(Conv2D(filters= 512, kernel_size= (3,3), strides= (1,1)
, padding='same', activation= 'relu'))
# model.add(BatchNormalization())
model.add(MaxPool2D(pool_size= (2,2), strides=(2,2)))

# Flattening the pooled image pixels
model.add(Flatten())
```

```
 # Creating 2 Dense Layers
 model.add(Dense(units= 4096, activation='relu'))
 model.add(Dropout(rate=0.5))
 model.add(Dense(units= 4096, activation='relu'))
 model.add(Dropout(rate=0.5))
 # Creating an output layer
 model.add(Dense(units= 2, activation='softmax'))

model.summary()

datagen = ImageDataGenerator(rescale=1./255)

train_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val,
        x_col="path",
        y_col="label",
        shuffle=True,
        target_size=(224,224),
        class_mode="categorical",)

class_names = train_generator.class_indices
print(class_names)
for image_batch, labels_batch in train_generator:
  print(image_batch.shape)
  print(labels_batch.shape)
  # print(image_batch[0].shape)
  # zz.fit(image_batch)
  Break

###CROSS VALIDATION EVALUASI 1
def cross_validation_model_evaluate(model,  valData, i):
  val_datagen = ImageDataGenerator(rescale=1./255,)
  valData = df_train_val.iloc[val_index]
  val_generator = val_datagen.flow_from_dataframe(
        dataframe=valData,
        directory=data_dir,
        x_col="path",
        y_col="label",
        shuffle=False,
        target_size=image_size,
        class_mode="categorical", )
  Y_pred = model.predict(val_generator,)
  y_pred = np.argmax(Y_pred, axis=1)
  y_test = val_generator.classes
  cm = ConfusionMatrix(actual_vector=y_test, predict_vector=y_pred
)
  print(cm)
  eval_csv = '/content/drive/MyDrive/DATASET_MATA_IKAN_2/' +"Model
/" + '_fold_' + str(i) + '_model.log'
  eval_csv = pd.read_csv(eval_csv, )
  plt.plot(eval_csv['accuracy'])
  plt.plot(eval_csv['val_accuracy'])
  plt.title('model performance')
  plt.ylabel('accuracy')
  plt.xlabel('epoch')
  plt.legend(['train_accuracy','val_accuracy'], loc='upper left')
  plt.show()
  ##
  plt.plot(eval_csv['loss'])
  plt.plot(eval_csv['val_loss'])
  plt.title('model performance loss')
  plt.ylabel('loss')
  plt.xlabel('epoch')
  plt.legend(['train_loss', 'val_loss'], loc='upper left')
  plt.show()
  eval_csv = eval_csv[(eval_csv.val_accuracy == eval_csv.val_accur
acy.max())]
  eval_csv = eval_csv[(eval_csv.val_loss== eval_csv.val_loss.min()
)]
  print(eval_csv)
```

```
##Train K-Fold Cross Validation K=10
k = 10
kf = StratifiedKFold(n_splits=k, shuffle=True, random_state=666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'],
df_train_val['label']):
    if i == 1:
        print('Train shape:',
df_train_val.iloc[train_index].shape)
        print('val shaoe:', df_train_val.iloc[val_index].shape)
        datagen = ImageDataGenerator(rescale=1. / 255, )
        csv_filepath = CSVLogger(
            '/content/drive/MyDrive/DATASET_MATA_IKAN_2/' +
"Model/" + '_fold_' + str(i) + '_model.log')
        model_checkpoint_callback = ModelCheckpoint(
            filepath='/content/drive/MyDrive/ DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5',
            save_weights_only=False,
            monitor='val_accuracy',
            mode='max',
            verbose=1,
            save_best_only=True)
        csvlogger = csv_filepath
        callbacks = []
        callbacks.append(csvlogger)
        callbacks.append(model_checkpoint_callback)
    train_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[train_index],
        x_col="path",
        y_col="label",
        shuffle=True,
        target_size=(224, 224),
        class_mode="categorical", )
    val_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[val_index],
        x_col="path",
        y_col="label",
        shuffle=False,
        target_size=(224, 224),
        class_mode="categorical", )
    optimizer = Adam(learning_rate=0.0001)
    model.compile(optimizer=optimizer,
loss='categorical_crossentropy', metrics=['accuracy'])
    result = model.fit(x=train_generator,
validation_data=val_generator, epochs=100, callbacks=callbacks)
    break
i += 1

n_splits = 10
kf =  StratifiedKFold(n_splits = n_splits, shuffle = True,
random_state = 666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'],
df_train_val['label']):
  if i == 1:
    model_filepath = '/content/drive/MyDrive/DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5'
    model = load_model(model_filepath)
    cross_validation_model_evaluate(
      model,  df_train_val.iloc[val_index], i,
       )
    break
  i += 1

k = 10
kf = StratifiedKFold(n_splits=k, shuffle=True, random_state=666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'],
df_train_val['label']):
    if i == 2:
        print('Train shape:',
df_train_val.iloc[train_index].shape)
        print('val shaoe:', df_train_val.iloc[val_index].shape)
        datagen = ImageDataGenerator(rescale=1. / 255, )
        csv_filepath = CSVLogger(
            '/content/drive/MyDrive/DATASET_MATA_IKAN_2/' +
"Model/" + '_fold_' + str(i) + '_model.log')
        model_checkpoint_callback = ModelCheckpoint(
            filepath='/content/drive/MyDrive/ DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5',
            save_weights_only=False,
            monitor='val_accuracy',
            mode='max',
            verbose=1,
            save_best_only=True)
        csvlogger = csv_filepath
        callbacks = []
        callbacks.append(csvlogger)
        callbacks.append(model_checkpoint_callback)
    train_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[train_index],
        x_col="path",
        y_col="label",
        shuffle=True,
        target_size=(224, 224),
        class_mode="categorical", )
    val_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[val_index],
        x_col="path",
        y_col="label",
        shuffle=False,
        target_size=(224, 224),
        class_mode="categorical", )
    optimizer = Adam(learning_rate=0.0001)
    model.compile(optimizer=optimizer,
loss='categorical_crossentropy', metrics=['accuracy'])
    result = model.fit(x=train_generator,
validation_data=val_generator, epochs=100, callbacks=callbacks)
    break
i += 1

n_splits = 10
kf =  StratifiedKFold(n_splits = n_splits, shuffle = True,
random_state = 666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'],
df_train_val['label']):
  if i == 2:
    model_filepath = '/content/drive/MyDrive/DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5'
    model = load_model(model_filepath)
    cross_validation_model_evaluate(
      model,  df_train_val.iloc[val_index], i,
       )
    break
  i += 1

k = 10
kf = StratifiedKFold(n_splits=k, shuffle=True, random_state=666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'],
df_train_val['label']):
    if i == 3:
        print('Train shape:',
df_train_val.iloc[train_index].shape)
        print('val shaoe:', df_train_val.iloc[val_index].shape)
        datagen = ImageDataGenerator(rescale=1. / 255, )
        csv_filepath = CSVLogger(
            '/content/drive/MyDrive/DATASET_MATA_IKAN_2/' +
"Model/" + '_fold_' + str(i) + '_model.log')
        model_checkpoint_callback = ModelCheckpoint(
            filepath='/content/drive/MyDrive/ DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5',
            monitor='val_accuracy',
            mode='max',
            verbose=1,
            save_best_only=True)
        csvlogger = csv_filepath
        callbacks = []
        callbacks.append(csvlogger)
        callbacks.append(model_checkpoint_callback)
    train_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[train_index],
        x_col="path",
        y_col="label",
        shuffle=True,
        target_size=(224, 224),
        class_mode="categorical", )
    val_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[val_index],
        x_col="path",
        y_col="label",
        shuffle=False,
        target_size=(224, 224),
        class_mode="categorical", )
    optimizer = Adam(learning_rate=0.0001)
    model.compile(optimizer=optimizer,
loss='categorical_crossentropy', metrics=['accuracy'])
    result = model.fit(x=train_generator,
validation_data=val_generator, epochs=100, callbacks=callbacks)
    break
i += 1

n_splits = 10
kf =  StratifiedKFold(n_splits = n_splits, shuffle = True,
random_state = 666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'],
df_train_val['label']):
  if i == 3:
    model_filepath = '/content/drive/MyDrive/DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5'
    model = load_model(model_filepath)
    cross_validation_model_evaluate(
      model,  df_train_val.iloc[val_index], i,
       )
    break
  i += 1

k = 10
kf = StratifiedKFold(n_splits=k, shuffle=True, random_state=666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'],
df_train_val['label']):
    if i == 4:
        print('Train shape:',
df_train_val.iloc[train_index].shape)
        print('val shaoe:', df_train_val.iloc[val_index].shape)
        datagen = ImageDataGenerator(rescale=1. / 255, )
        csv_filepath = CSVLogger(
            '/content/drive/MyDrive/DATASET_MATA_IKAN_2/' +
"Model/" + '_fold_' + str(i) + '_model.log')
        model_checkpoint_callback = ModelCheckpoint(
            filepath='/content/drive/MyDrive/ DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5',
            save_weights_only=False,
            monitor='val_accuracy',
            mode='max',
            verbose=1,
            save_best_only=True)
        csvlogger = csv_filepath
        callbacks = []
        callbacks.append(csvlogger)
        callbacks.append(model_checkpoint_callback)
    train_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[train_index],
        x_col="path",
        y_col="label",
        shuffle=True,
        target_size=(224, 224),
        class_mode="categorical", )
    val_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[val_index],
        x_col="path",
        y_col="label",
        shuffle=False,
        target_size=(224, 224),
        class_mode="categorical", )
    optimizer = Adam(learning_rate=0.0001)
    model.compile(optimizer=optimizer,
loss='categorical_crossentropy', metrics=['accuracy'])
    result = model.fit(x=train_generator,
validation_data=val_generator, epochs=100, callbacks=callbacks)
    break
i += 1

n_splits = 10
kf =  StratifiedKFold(n_splits = n_splits, shuffle = True,
random_state = 666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'],
df_train_val['label']):
  if i == 4:
    model_filepath = '/content/drive/MyDrive/DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5'
    model = load_model(model_filepath)
    cross_validation_model_evaluate(
      model,  df_train_val.iloc[val_index], i,
       )
    break
  i += 1

k = 10
kf = StratifiedKFold(n_splits=k, shuffle=True, random_state=666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'],
df_train_val['label']):
    if i == 5:
        print('Train shape:',
df_train_val.iloc[train_index].shape)
        print('val shaoe:', df_train_val.iloc[val_index].shape)
        datagen = ImageDataGenerator(rescale=1. / 255, )
        csv_filepath = CSVLogger(
            '/content/drive/MyDrive/DATASET_MATA_IKAN_2/' +
"Model/" + '_fold_' + str(i) + '_model.log')
        model_checkpoint_callback = ModelCheckpoint(
            filepath='/content/drive/MyDrive/ DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5',
            save_weights_only=False,
            monitor='val_accuracy',
            mode='max',
            verbose=1,
            save_best_only=True)
        csvlogger = csv_filepath
        callbacks = []
        callbacks.append(csvlogger)
        callbacks.append(model_checkpoint_callback)
    train_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[train_index],
        x_col="path",
        y_col="label",
        shuffle=True,
        target_size=(224, 224),
        class_mode="categorical", )
    val_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[val_index],
        x_col="path",
        y_col="label",
        shuffle=False,
        target_size=(224, 224),
        class_mode="categorical", )
    optimizer = Adam(learning_rate=0.0001)
    model.compile(optimizer=optimizer,
loss='categorical_crossentropy', metrics=['accuracy'])
    result = model.fit(x=train_generator,
validation_data=val_generator, epochs=100, callbacks=callbacks)
    break
i += 1

n_splits = 10
kf =  StratifiedKFold(n_splits = n_splits, shuffle = True,
random_state = 666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'],
df_train_val['label']):
  if i == 5:
    model_filepath = '/content/drive/MyDrive/DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5'
    model = load_model(model_filepath)
    cross_validation_model_evaluate(
      model,  df_train_val.iloc[val_index], i,
       )
    break
  i += 1

k = 10
kf = StratifiedKFold(n_splits=k, shuffle=True, random_state=666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'],
df_train_val['label']):
    if i == 6:
        print('Train shape:',
df_train_val.iloc[train_index].shape)
        print('val shaoe:', df_train_val.iloc[val_index].shape)
        datagen = ImageDataGenerator(rescale=1. / 255, )
        csv_filepath = CSVLogger(
            '/content/drive/MyDrive/DATASET_MATA_IKAN_2/' +
"Model/" + '_fold_' + str(i) + '_model.log')
        model_checkpoint_callback = ModelCheckpoint(
            filepath='/content/drive/MyDrive/ DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5',
            save_weights_only=False,
            monitor='val_accuracy',
            mode='max',
            verbose=1,
            save_best_only=True)
        csvlogger = csv_filepath
        callbacks = []
        callbacks.append(csvlogger)
        callbacks.append(model_checkpoint_callback)
    train_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[train_index],
        x_col="path",
        y_col="label",
        shuffle=True,
        target_size=(224, 224),
        class_mode="categorical", )
    val_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[val_index],
        x_col="path",
        y_col="label",
        shuffle=False,
        target_size=(224, 224),
        class_mode="categorical", )
    optimizer = Adam(learning_rate=0.0001)
    model.compile(optimizer=optimizer,
loss='categorical_crossentropy', metrics=['accuracy'])
    result = model.fit(x=train_generator, 
validation_data=val_generator, epochs=100, callbacks=callbacks)
    break
i += 1

n_splits = 10
kf =  StratifiedKFold(n_splits = n_splits, shuffle = True, 
random_state = 666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'], 
df_train_val['label']):
  if i == 6:
    model_filepath = '/content/drive/MyDrive/DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5'
    model = load_model(model_filepath)
    cross_validation_model_evaluate(
      model,  df_train_val.iloc[val_index], i,
       )
    break
  i += 1

k = 10
kf = StratifiedKFold(n_splits=k, shuffle=True, random_state=666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'], 
df_train_val['label']):
    if i == 7:
        print('Train shape:', 
df_train_val.iloc[train_index].shape)
        print('val shaoe:', df_train_val.iloc[val_index].shape)
        datagen = ImageDataGenerator(rescale=1. / 255, )
        csv_filepath = CSVLogger(
            '/content/drive/MyDrive/DATASET_MATA_IKAN_2/' + 
"Model/" + '_fold_' + str(i) + '_model.log')
        model_checkpoint_callback = ModelCheckpoint(
            filepath='/content/drive/MyDrive/ DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5',
            save_weights_only=False,
            monitor='val_accuracy',
            mode='max',
            verbose=1,
            save_best_only=True)
        csvlogger = csv_filepath
        callbacks = []
        callbacks.append(csvlogger)
        callbacks.append(model_checkpoint_callback)
    train_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[train_index],
        x_col="path",
        y_col="label",
        shuffle=True,
        target_size=(224, 224),
        class_mode="categorical", )
    val_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[val_index],
        x_col="path",
        y_col="label",
        shuffle=False,
        target_size=(224, 224),
        class_mode="categorical", )
    optimizer = Adam(learning_rate=0.0001)
    model.compile(optimizer=optimizer, 
loss='categorical_crossentropy', metrics=['accuracy'])
    result = model.fit(x=train_generator,
validation_data=val_generator, epochs=100, callbacks=callbacks)
    break
i += 1

n_splits = 10
kf =  StratifiedKFold(n_splits = n_splits, shuffle = True,
random_state = 666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'],
df_train_val['label']):
  if i == 7:
    model_filepath = '/content/drive/MyDrive/DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5'
    model = load_model(model_filepath)
    cross_validation_model_evaluate(
      model,  df_train_val.iloc[val_index], i,
       )
    break
  i += 1

k = 10
kf = StratifiedKFold(n_splits=k, shuffle=True, random_state=666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'],
df_train_val['label']):
    if i == 8:
        print('Train shape:',
df_train_val.iloc[train_index].shape)
        print('val shaoe:', df_train_val.iloc[val_index].shape)
        datagen = ImageDataGenerator(rescale=1. / 255, )
        csv_filepath = CSVLogger(
            '/content/drive/MyDrive/DATASET_MATA_IKAN_2/' +
"Model/" + '_fold_' + str(i) + '_model.log')
        model_checkpoint_callback = ModelCheckpoint(
            filepath='/content/drive/MyDrive/ DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5',
            save_weights_only=False,
            monitor='val_accuracy',
            mode='max',
            verbose=1,
            save_best_only=True)
        csvlogger = csv_filepath
        callbacks = []
        callbacks.append(csvlogger)
        callbacks.append(model_checkpoint_callback)
    train_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[train_index],
        x_col="path",
        y_col="label",
        shuffle=True,
        target_size=(224, 224),
        class_mode="categorical", )
    val_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[val_index],
        x_col="path",
        y_col="label",
        shuffle=False,
        target_size=(224, 224),
        class_mode="categorical", )
    optimizer = Adam(learning_rate=0.0001)
    model.compile(optimizer=optimizer,
loss='categorical_crossentropy', metrics=['accuracy'])
    result = model.fit(x=train_generator,
validation_data=val_generator, epochs=100, callbacks=callbacks)
    break
```

```
i += 1

n_splits = 10
kf =  StratifiedKFold(n_splits = n_splits, shuffle = True,
random_state = 666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'],
df_train_val['label']):
  if i == 8:
    model_filepath = '/content/drive/MyDrive/DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5'
    model = load_model(model_filepath)
    cross_validation_model_evaluate(
      model,  df_train_val.iloc[val_index], i,
       )
    break
  i += 1

k = 10
kf = StratifiedKFold(n_splits=k, shuffle=True, random_state=666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'],
df_train_val['label']):
    if i == 9:
        print('Train shape:',
df_train_val.iloc[train_index].shape)
        print('val shaoe:', df_train_val.iloc[val_index].shape)
        datagen = ImageDataGenerator(rescale=1. / 255, )
        csv_filepath = CSVLogger(
            '/content/drive/MyDrive/DATASET_MATA_IKAN_2/' +
"Model/" + '_fold_' + str(i) + '_model.log')
        model_checkpoint_callback = ModelCheckpoint(
            filepath='/content/drive/MyDrive/ DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5',
            save_weights_only=False,
            monitor='val_accuracy',
            mode='max',
            verbose=1,
            save_best_only=True)
        csvlogger = csv_filepath
        callbacks = []
        callbacks.append(csvlogger)
        callbacks.append(model_checkpoint_callback)
    train_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[train_index],
        x_col="path",
        y_col="label",
        shuffle=True,
        target_size=(224, 224),
        class_mode="categorical", )
    val_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[val_index],
        x_col="path",
        y_col="label",
        shuffle=False,
        target_size=(224, 224),
        class_mode="categorical", )
    optimizer = Adam(learning_rate=0.0001)
    model.compile(optimizer=optimizer,
loss='categorical_crossentropy', metrics=['accuracy'])
    result = model.fit(x=train_generator,
validation_data=val_generator, epochs=100, callbacks=callbacks)
    break
i += 1
```

```
n_splits = 10
kf =  StratifiedKFold(n_splits = n_splits, shuffle = True,
random_state = 666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'],
df_train_val['label']):
  if i == 9:
    model_filepath = '/content/drive/MyDrive/DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5'
    model = load_model(model_filepath)
    cross_validation_model_evaluate(
      model,  df_train_val.iloc[val_index], i,
       )
    break
  i += 1

k = 10
kf = StratifiedKFold(n_splits=k, shuffle=True, random_state=666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'],
df_train_val['label']):
    if i == 10:
        print('Train shape:',
df_train_val.iloc[train_index].shape)
        print('val shaoe:', df_train_val.iloc[val_index].shape)
        datagen = ImageDataGenerator(rescale=1. / 255, )
        csv_filepath = CSVLogger(
            '/content/drive/MyDrive/DATASET_MATA_IKAN_2/' +
"Model/" + '_fold_' + str(i) + '_model.log')
        model_checkpoint_callback = ModelCheckpoint(
            filepath='/content/drive/MyDrive/ DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5',
            save_weights_only=False,
            monitor='val_accuracy',
            mode='max',
            verbose=1,
            save_best_only=True)
        csvlogger = csv_filepath
        callbacks = []
        callbacks.append(csvlogger)
        callbacks.append(model_checkpoint_callback)
    train_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[train_index],
        x_col="path",
        y_col="label",
        shuffle=True,
        target_size=(224, 224),
        class_mode="categorical", )
    val_generator = datagen.flow_from_dataframe(
        dataframe=df_train_val.iloc[val_index],
        x_col="path",
        y_col="label",
        shuffle=False,
        target_size=(224, 224),
        class_mode="categorical", )
    optimizer = Adam(learning_rate=0.0001)
    model.compile(optimizer=optimizer,
loss='categorical_crossentropy', metrics=['accuracy'])
    result = model.fit(x=train_generator,
validation_data=val_generator, epochs=100, callbacks=callbacks)
    break
i += 1

n_splits = 10
kf =  StratifiedKFold(n_splits = n_splits, shuffle = True,
random_state = 666)
i = 1
for train_index, val_index in kf.split(df_train_val['path'],
df_train_val['label']):
  if i == 10:
    model_filepath = '/content/drive/MyDrive/DATASET_MATA_IKAN_2
/' + "Model/" + '_fold_' + str(i) + '_model.h5'
    model = load_model(model_filepath)
    cross_validation_model_evaluate(
      model,  df_train_val.iloc[val_index], i,
       )
    break
  i += 1

##Create Model Evaluasi
def cross_validation_model_evaluate(model,  valData, i):
  val_datagen = ImageDataGenerator(rescale=1./255,)
  valData = df_train_val.iloc[val_index]
  val_generator = val_datagen.flow_from_dataframe(
        dataframe=valData,
        directory=data_dir,
        x_col="path",
        y_col="label",
        shuffle=False,
        target_size=image_size,
        class_mode="categorical", )
  Y_pred = model.predict(val_generator,)
  y_pred = np.argmax(Y_pred, axis=1)
  y_test = val_generator.classes
  cm = ConfusionMatrix(actual_vector=y_test,
predict_vector=y_pred)
  print(cm)
  eval_csv = '/content/drive/MyDrive/DATASET_MATA_IKAN_2/'
+"Model/" + '_fold_' + str(i) + '_model.log'
  eval_csv = pd.read_csv(eval_csv, )
  plt.plot(eval_csv['accuracy'])
  plt.plot(eval_csv['val_accuracy'])
  plt.title('model performance')
  plt.ylabel('accuracy')
  plt.xlabel('epoch')
  plt.legend(['train_accuracy','val_accuracy'], loc='upper left')
  plt.show()

  plt.plot(eval_csv['loss'])
  plt.plot(eval_csv['val_loss'])
  plt.title('model performance loss')
  plt.ylabel('loss')
  plt.xlabel('epoch')
  plt.legend(['train_loss', 'val_loss'], loc='upper left')
  plt.show()
  eval_csv = eval_csv[(eval_csv.val_accuracy ==
eval_csv.val_accuracy.max())]
  eval_csv = eval_csv[(eval_csv.val_loss==
eval_csv.val_loss.min())]
  print(eval_csv)

  ##Evaluasi Keseluruhan
  df_result_experiment = pd.DataFrame()
  for i in range(1, 11):
      df_result = pd.read_csv('/content/drive/MyDrive/
DATASET_MATA_IKAN_2/' + "ModeL/" + '_fold_' + str(i) +
'_model.log')
      df_result = df_result[(df_result.val_accuracy ==
df_result.val_accuracy.max())]
      df_result = df_result[(df_result.val_loss ==
df_result.val_loss.min())]
      df_result_experiment = pd.concat([df_result_experiment,
df_result])
  df_result_experiment =
df_result_experiment.reset_index(drop=True)
  df_result_experiment.index += 1
  pd.set_option('display.max_rows', df_result_experiment.shape[0]
+ 1)
  df_average = df_result_experiment.iloc[:].mean(axis=0)
  df_result_experiment.loc[11] = df_average
  df_result_experiment =
df_result_experiment.style.background_gradient(cmap='Accent',

      subset=df_result_experiment.index[-1])
  df_result_experiment

##Uji Coba Keefektivitas Model
  val_datagen = ImageDataGenerator(rescale=1. / 255, )
  test_generator = val_datagen.flow_from_dataframe(
      dataframe=df_test_2,
      directory=data_dir,
      x_col="path",
      y_col="label",
      shuffle=False,
      target_size=image_size,
      class_mode="categorical", )
  model_filepath = '/content/drive/MyDrive/ DATASET_MATA_IKAN_2/'
+ "Model/" + '_fold_' + '2' + '_model.h5'
  model = load_model(model_filepath)
  Y_pred = model.predict(test_generator, )
  y_pred = np.argmax(Y_pred, axis=1)
  y_test = test_generator.classes
  cm = ConfusionMatrix(actual_vector=y_test,
predict_vector=y_pred)
  print(cm)

##Uji Coba Data Testing
model_filepath = '/content/drive/MyDrive/ DATASET_MATA_IKAN_2/' +
"ModeL/" + '_fold_' + '2' + '_model.h5'
model = load_model(model_filepath)
def plot_img_array(path):
    img = image.load_img(path)
    plt.grid(False)
    plt.imshow(img)
def predict_image(model, google_drive=None):
    if google_drive == None:
        uploaded = files.upload()
        for fn in uploaded.keys():
            path = '/content/' + fn
    else:
        path = google_drive
    image = tf.keras.preprocessing.image.load_img(path,
                                              target_size=ima
ge_size, )
    input_arr = tf.keras.preprocessing.image.img_to_array(image)
/ 255
    input_arr = np.array([input_arr])
    classes = model.predict(input_arr)
    rows = 10
    cols = 2
    plt.figure(figsize=(2 * 4 * cols, 4 * rows))
    index = 1
    plt.subplot(rows, cols * 2, index * 2 + 1)
    plot_img_array(path)
    keyList = list(class_names.keys())
    valList = list(class_names.values())
    position = valList.index(np.argmax(classes))
    prediction = 'prediction is ' + str(keyList[position])
    print(prediction)
    if google_drive == None:
        os.remove(path)
def plot_predict_value_array(predictions_array):
    plt.grid(False)
    plt.ylim([0, 1])
predict_image(model, '/content/drive/MyDrive/
DATASET_MATA_IKAN_2/MATA_IKAN_1.1/segar1.1/ _5_1479445.png')
```