UNIVERSITAS ISLAM KALIMANTAN MUHAMMAD ARSYAD AL BANJARI
FAKULTAS TEKNOLOGI INFORMASI
PROGRAM STUDI TEKNIK INFORMATIKA

UJIAN AKHIR SEMESTER (UAS) Ganjil T A 2022/2023

Dosen	: Haldi Budiman, M.Kom	Hari/Tanggal	: Selasa/ 10-1-2023
Mata Kuliah	: Kecerdasan Buatan	Kelas	: 5B
Sifat	: Open book	Waktu	: 09.10 – 10.10 Wita

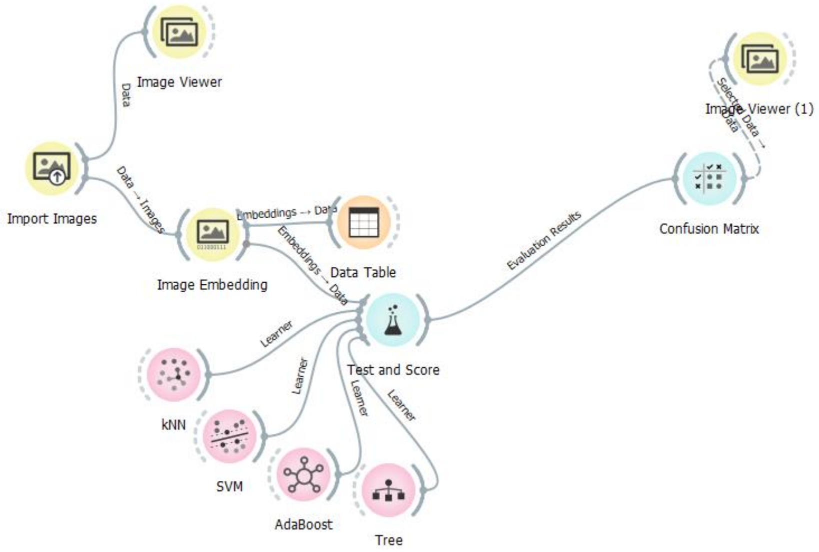
SOAL UJIAN AKHIR SEMESTER KECERDASAN BUATAN

1. Dalam artikel dengan judul Komparasi Algoritma Nonparametrik untuk Klasifikasi Citra Wajah Berdasarkan Suku di Indonesia oleh Seno Hartono, Anggi Perwitasari, Herry Sujaini, 2020 diterbitkan pada : jurnal Edukasi & Penelitian nformatika (JEPIN) dengan index Sinta2. https://jurnal.untan.ac.id/index.php/jepin/article/view/43268
   soal : dengan menggunakan teknik klasifikasi seperti artikel tersebut, dan algoritma k-Nearest Neigbor, Support Vector Machine, Decision Tree, dan AdaBoost, SVM, buat di Orange ML dengan dataset yang berbeda atau dataset yang anda miliki.

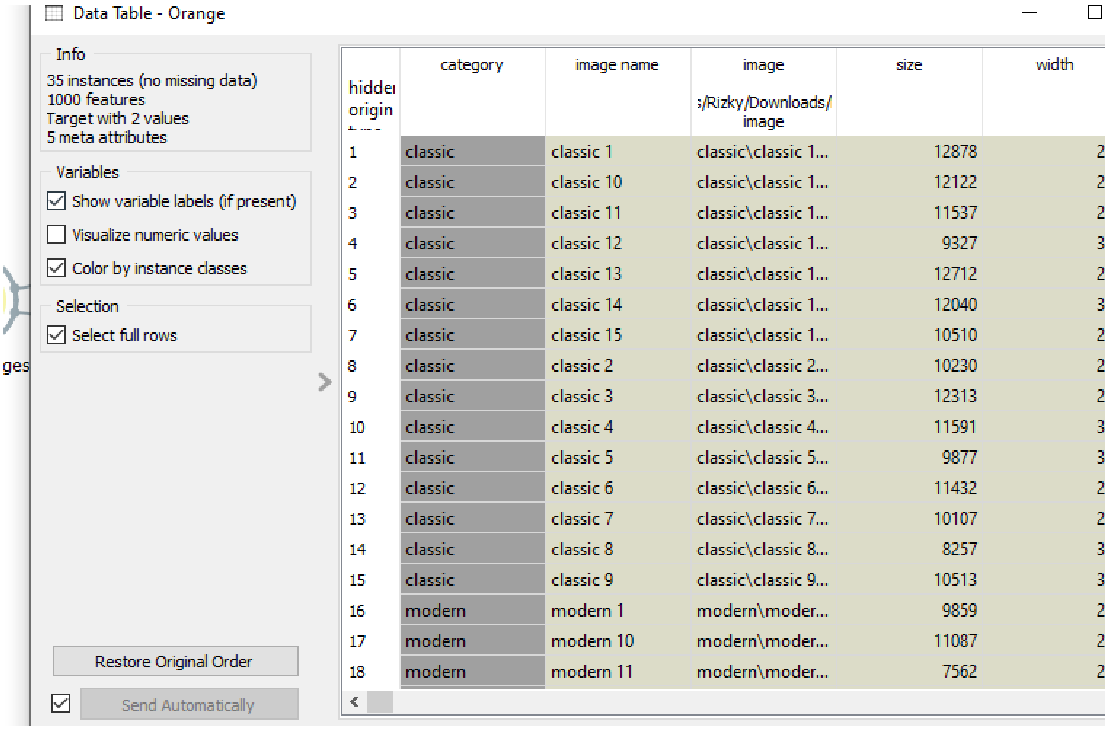
2. Source code di colab berikut adalah klasifikasi mobil dengan dataset di upload dari google drive, dengan teknik yang sama menggunakan dataset berbeda atau dataset yang anda miliki buatlah Klasifikasi dan hasil analisinya di lembar jawaban.
   Link colab :
   https://colab.research.google.com/drive/1MncZS75axMnmlsXhfufmPD6_zqL54gvs?usp=share_link

Jawab :

1. Gambar Penggunaan pada Orange :



Isi Data Table:



Data Table - Orange

Info

35 instances (no missing data)
1000 features
Target with 2 values
5 meta attributes

Variables

☑ Show variable labels (if present)

☐ Visualize numeric values

☑ Color by instance classes

Selection

☑ Select full rows

Restore Original Order

☑ Send Automatically

hiddei origin	category	image name	image ;/Rizky/Downloads/ image	size	width
1	classic	classic 1	classic\classic 1...	12878	2
2	classic	classic 10	classic\classic 1...	12122	2
3	classic	classic 11	classic\classic 1...	11537	2
4	classic	classic 12	classic\classic 1...	9327	3
5	classic	classic 13	classic\classic 1...	12712	2
6	classic	classic 14	classic\classic 1...	12040	3
7	classic	classic 15	classic\classic 1...	10510	2
8	classic	classic 2	classic\classic 2...	10230	2
9	classic	classic 3	classic\classic 3...	12313	2
10	classic	classic 4	classic\classic 4...	11591	3
11	classic	classic 5	classic\classic 5...	9877	3
12	classic	classic 6	classic\classic 6...	11432	2
13	classic	classic 7	classic\classic 7...	10107	2
14	classic	classic 8	classic\classic 8...	8257	3
15	classic	classic 9	classic\classic 9...	10513	3
16	modern	modern 1	modern\moder...	9859	2
17	modern	modern 10	modern\moder...	11087	2
18	modern	modern 11	modern\moder...	7562	2

Isi Image dataset :

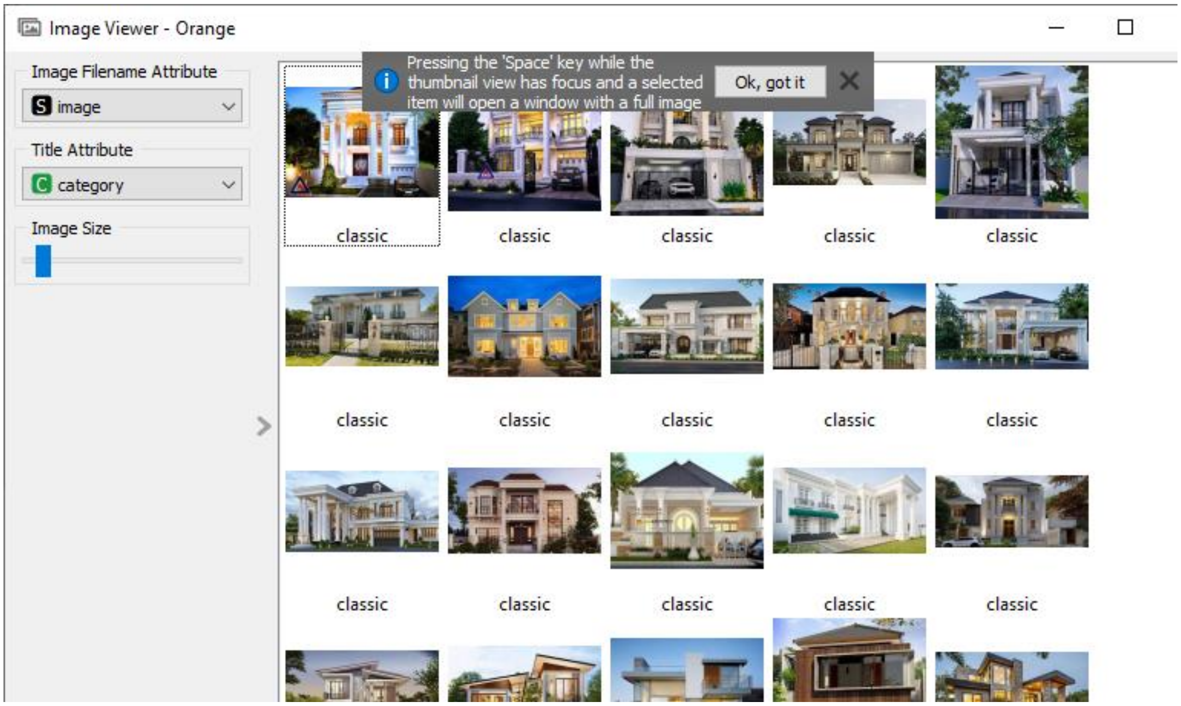


Image Viewer - Orange

Image Filename Attribute: image

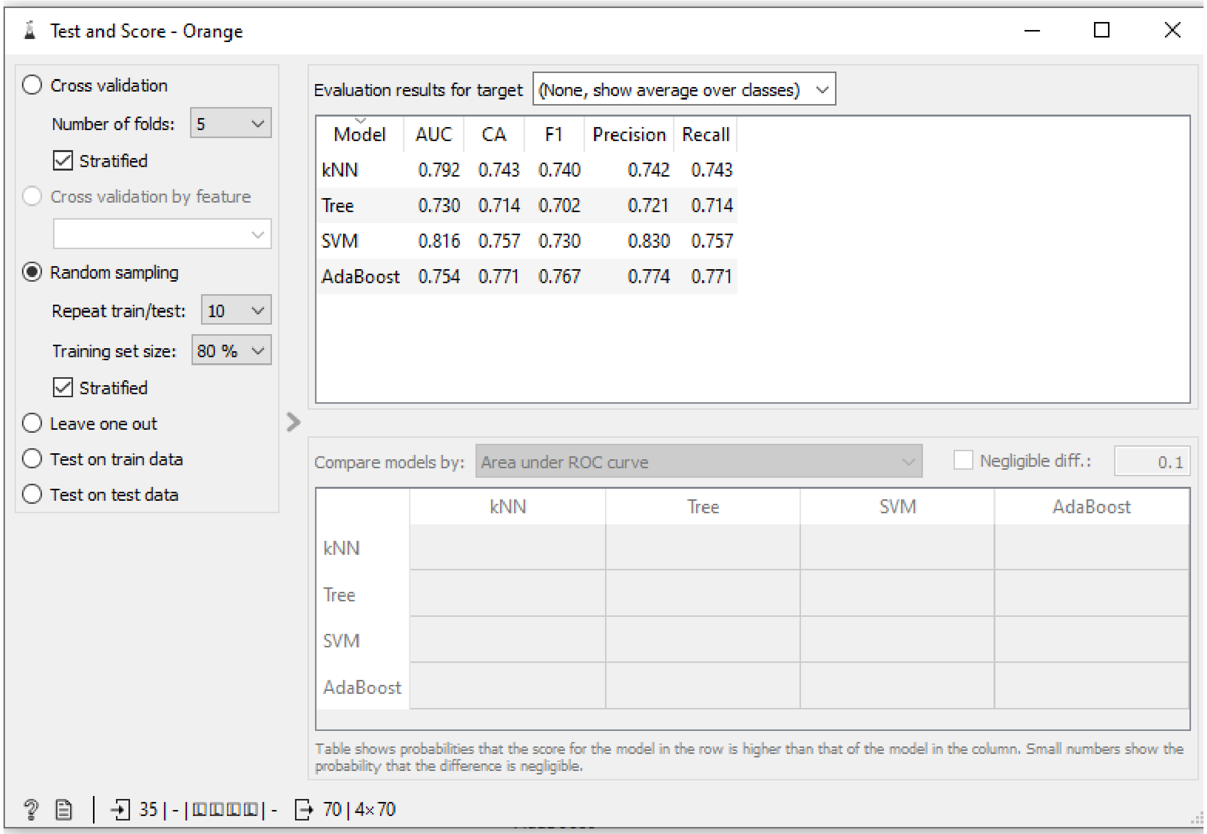
Title Attribute: category

Image Size

Pressing the 'Space' key while the thumbnail view has focus and a selected item will open a window with a full image

Ok, got it

Hasil Test and Score berdasarkan dataset yang ada:



Test and Score - Orange

○ Cross validation

Number of folds: 5

☑ Stratified

○ Cross validation by feature

● Random sampling

Repeat train/test: 10

Training set size: 80 %

☑ Stratified

○ Leave one out

○ Test on train data

○ Test on test data

Evaluation results for target (None, show average over classes)

Model	AUC	CA	F1	Precision	Recall
kNN	0.792	0.743	0.740	0.742	0.743
Tree	0.730	0.714	0.702	0.721	0.714
SVM	0.816	0.757	0.730	0.830	0.757
AdaBoost	0.754	0.771	0.767	0.774	0.771

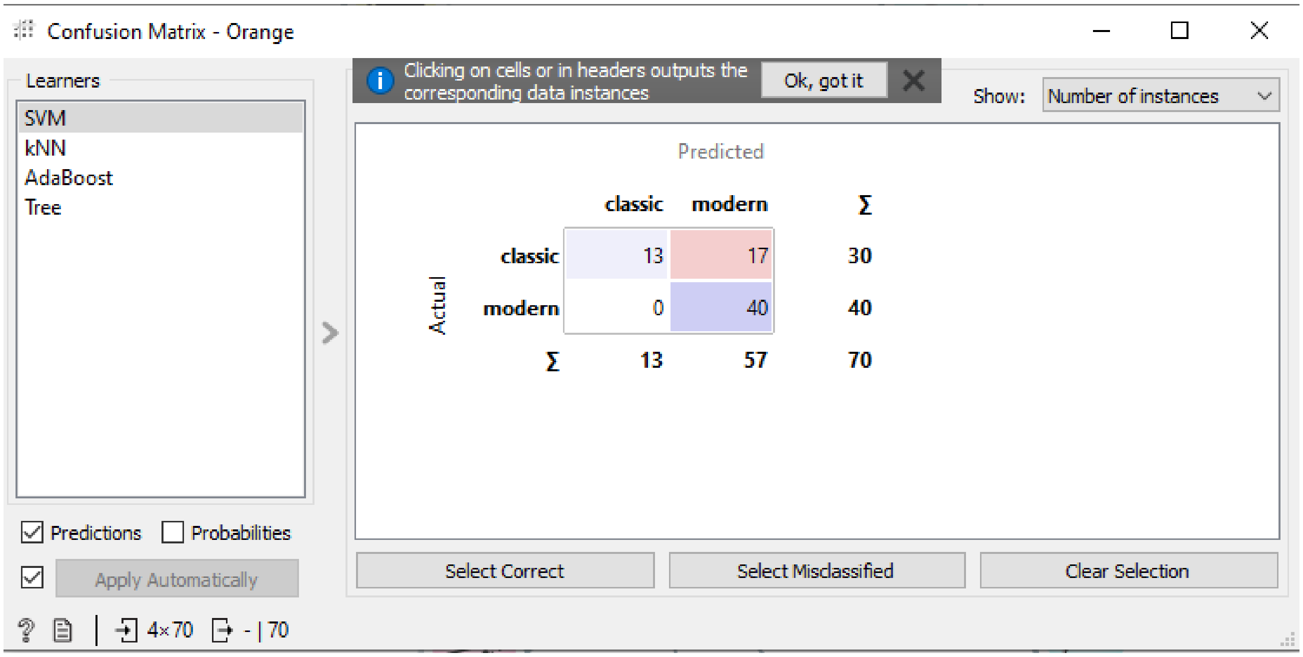
Compare models by: Area under ROC curve □ Negligible diff.: 0.1

	kNN	Tree	SVM	AdaBoost
kNN				
Tree				
SVM				
AdaBoost				

Table shows probabilities that the score for the model in the row is higher than that of the model in the column. Small numbers show the probability that the difference is negligible.

35 | - | | - 70 | 4×70

Hasil Confusion Matrix SVM :



Confusion Matrix - Orange

Learners

SVM
kNN
AdaBoost
Tree

Clicking on cells or in headers outputs the corresponding data instances

Show: Number of instances

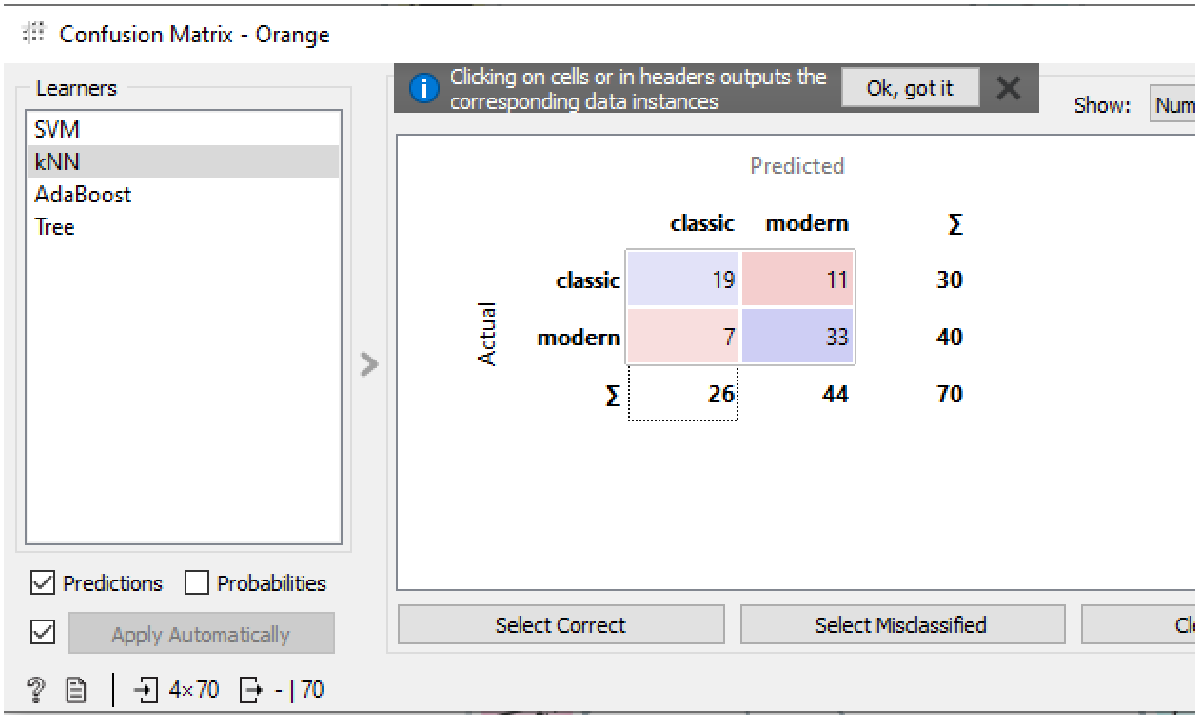
		Predicted		
		classic	modern	∑
Actual	classic	13	17	30
	modern	0	40	40
	∑	13	57	70

☑ Predictions □ Probabilities

☑ Apply Automatically

4×70 - | 70

Hasil Confusion Matrix kNN:



Clicking on cells or in headers outputs the corresponding data instances

Show: Num

Learners: SVM, kNN, AdaBoost, Tree

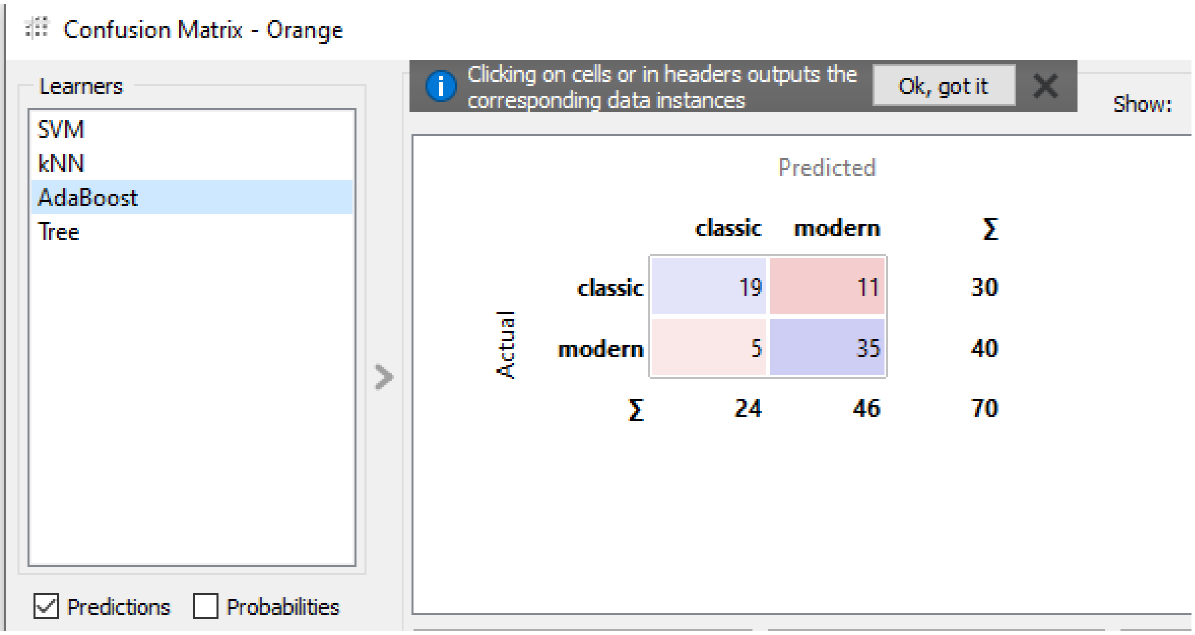
		Predicted		
		classic	modern	Σ
Actual	classic	19	11	30
	modern	7	33	40
	Σ	26	44	70

☑ Predictions □ Probabilities

☑ Apply Automatically

4×70 - | 70

Hasil Confusion Matrix AdaBoost :



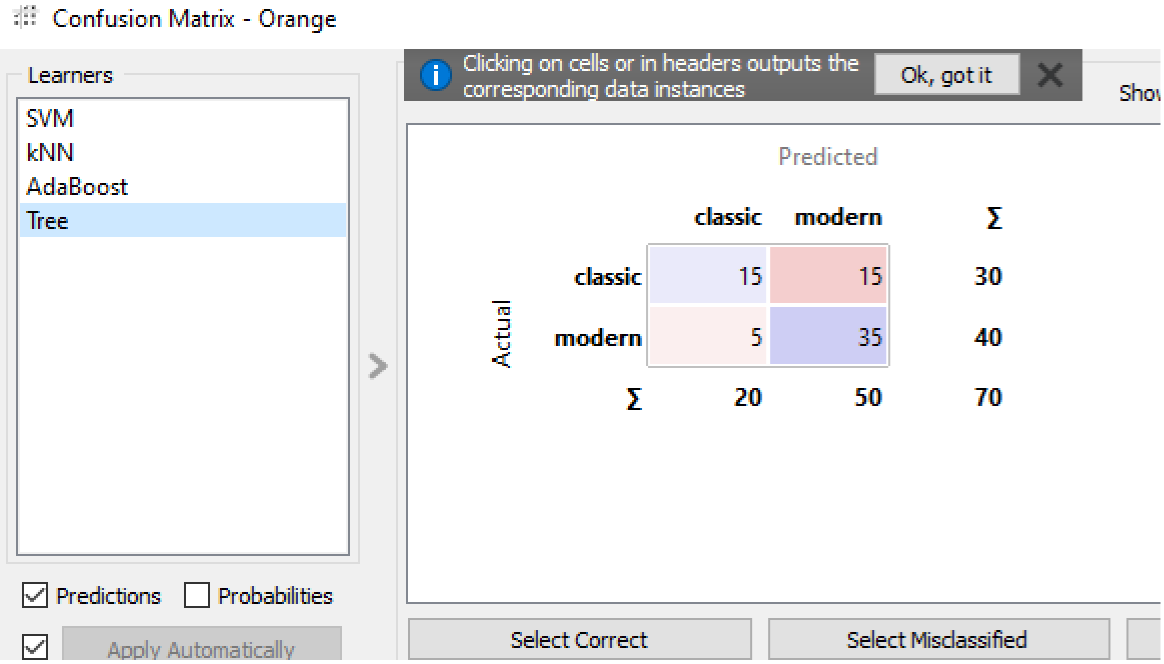
Clicking on cells or in headers outputs the corresponding data instances

Learners: SVM, kNN, AdaBoost, Tree

		Predicted		
		classic	modern	Σ
Actual	classic	19	11	30
	modern	5	35	40
	Σ	24	46	70

☑ Predictions □ Probabilities

Hasil Confusion Matrix Tree:



Confusion Matrix - Orange

Learners

- SVM
- kNN
- AdaBoost
- Tree

Clicking on cells or in headers outputs the corresponding data instances

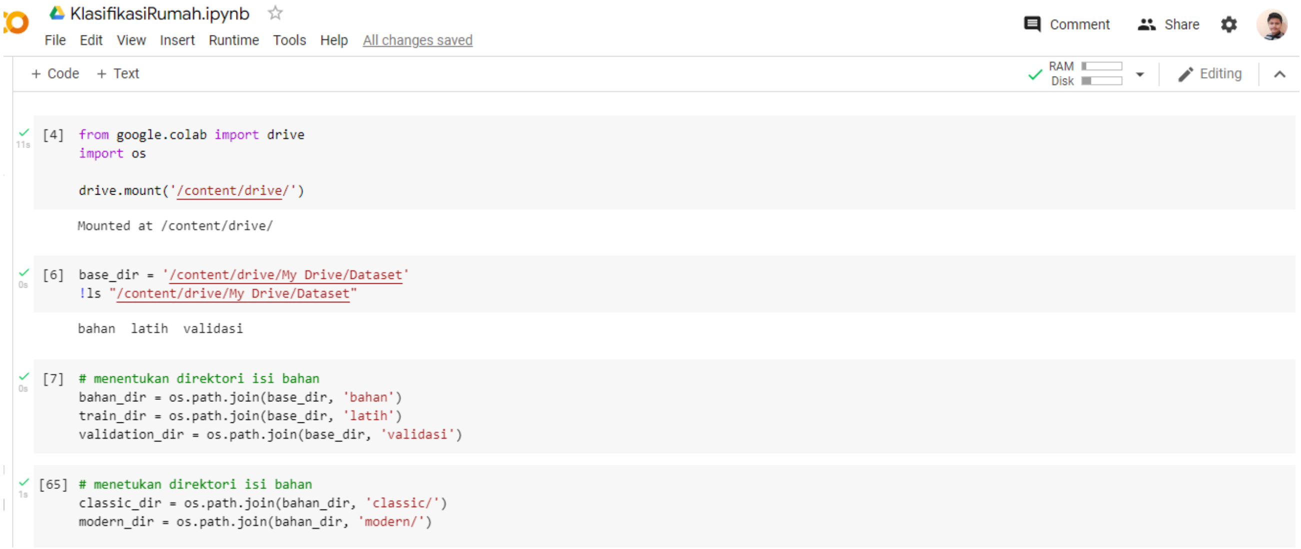
Actual \ Predicted	classic	modern	Σ
classic	15	15	30
modern	5	35	40
Σ	20	50	70

☑ Predictions ☐ Probabilities

☑ Apply Automatically

2. Google Colab:

Dataset yang digunakan Rumah Modern dan Rumah Classic



KlasifikasiRumah.ipynb

```
[4]  from google.colab import drive
     import os

     drive.mount('/content/drive/')

     Mounted at /content/drive/
```

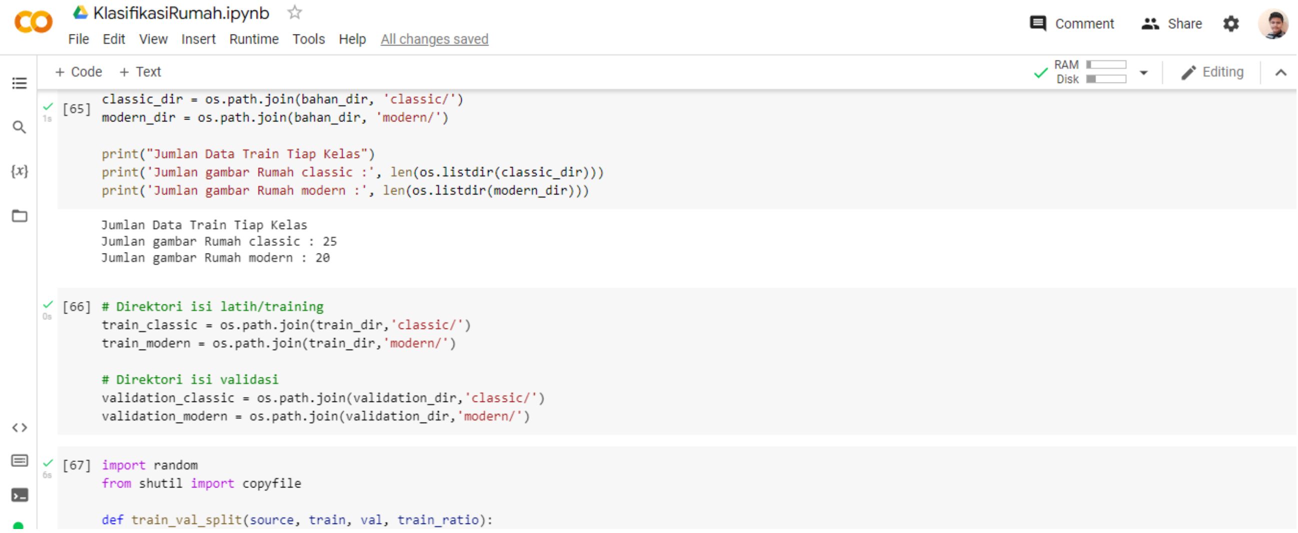
```
[6]  base_dir = '/content/drive/My Drive/Dataset'
     !ls "/content/drive/My Drive/Dataset"

     bahan  latih  validasi
```

```
[7]  # menentukan direktori isi bahan
     bahan_dir = os.path.join(base_dir, 'bahan')
     train_dir = os.path.join(base_dir, 'latih')
     validation_dir = os.path.join(base_dir, 'validasi')
```

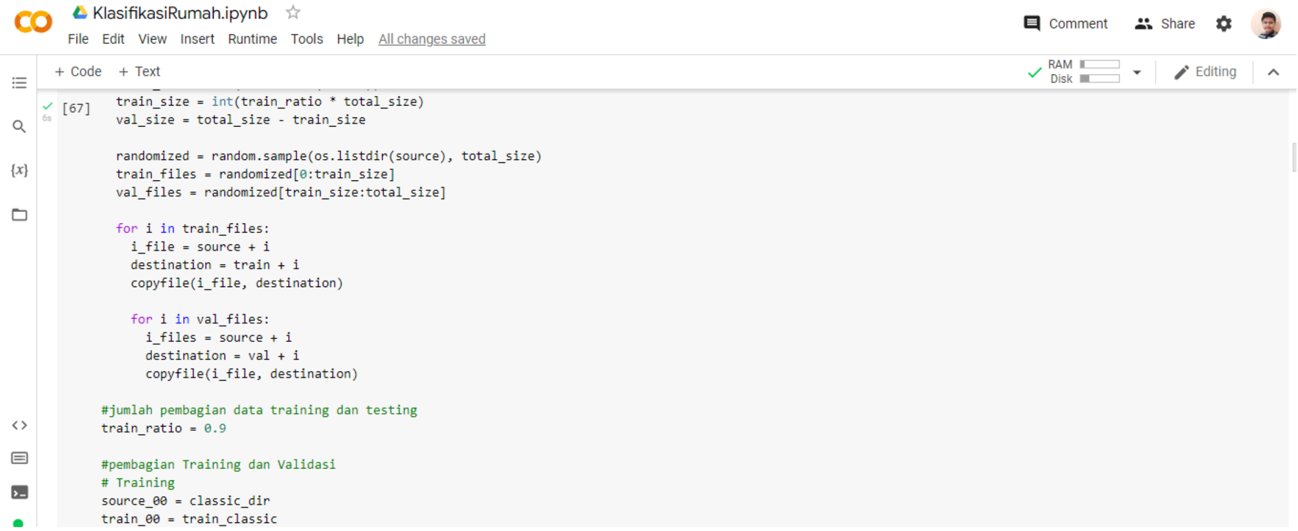
```
[65] # menetukan direktori isi bahan
     classic_dir = os.path.join(bahan_dir, 'classic/')
     modern_dir = os.path.join(bahan_dir, 'modern/')
```

Data yang di deteksi oleh google colab adalah

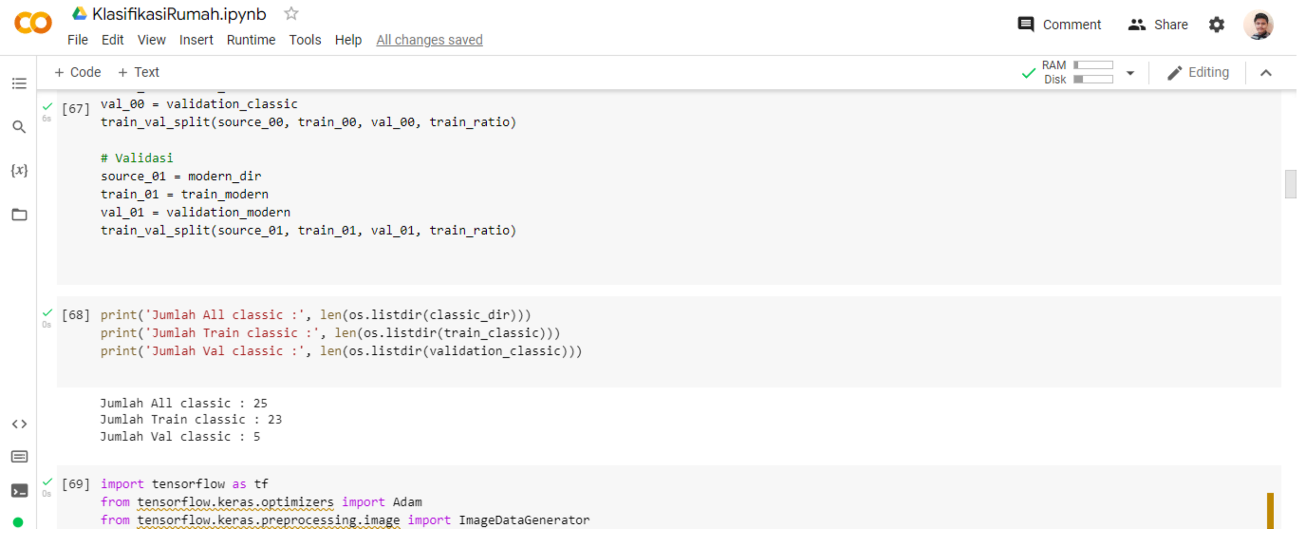
25 Rumah Classic dan 20 Rumah Modern



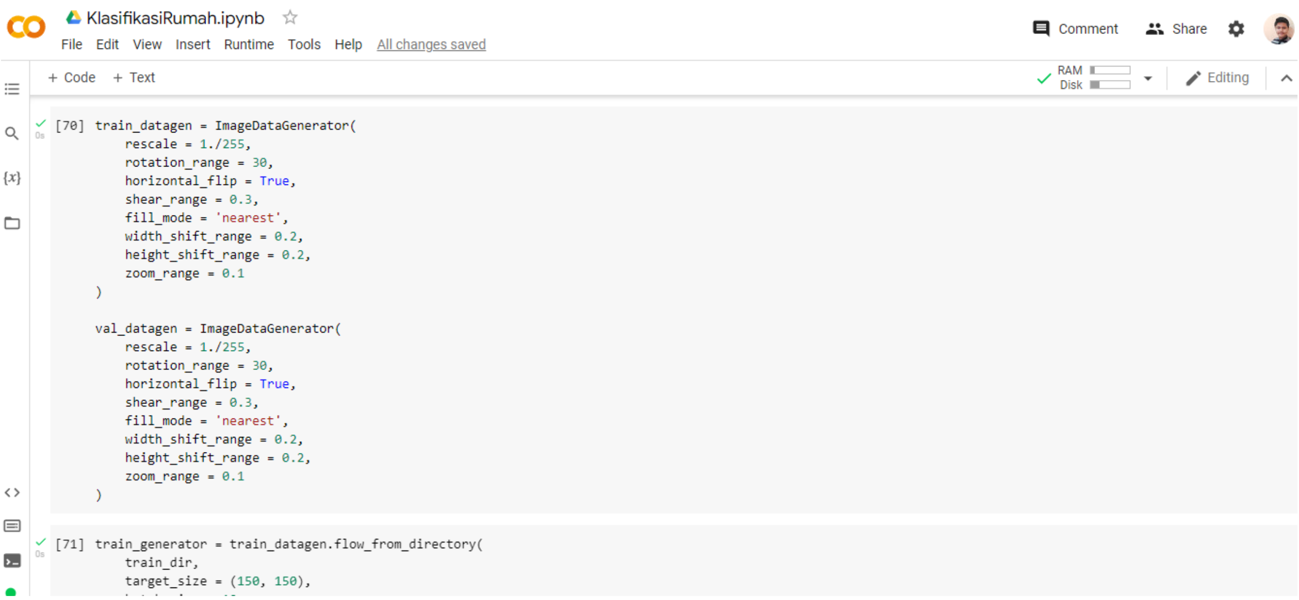
```
[65] classic_dir = os.path.join(bahan_dir, 'classic/')
     modern_dir = os.path.join(bahan_dir, 'modern/')

     print("Jumlah Data Train Tiap Kelas")
     print('Jumlah gambar Rumah classic :', len(os.listdir(classic_dir)))
     print('Jumlah gambar Rumah modern :', len(os.listdir(modern_dir)))

     Jumlah Data Train Tiap Kelas
     Jumlah gambar Rumah classic : 25
     Jumlah gambar Rumah modern : 20

[66] # Direktori isi latih/training
     train_classic = os.path.join(train_dir,'classic/')
     train_modern = os.path.join(train_dir,'modern/')

     # Direktori isi validasi
     validation_classic = os.path.join(validation_dir,'classic/')
     validation_modern = os.path.join(validation_dir,'modern/')

[67] import random
     from shutil import copyfile

     def train_val_split(source, train, val, train_ratio):
```



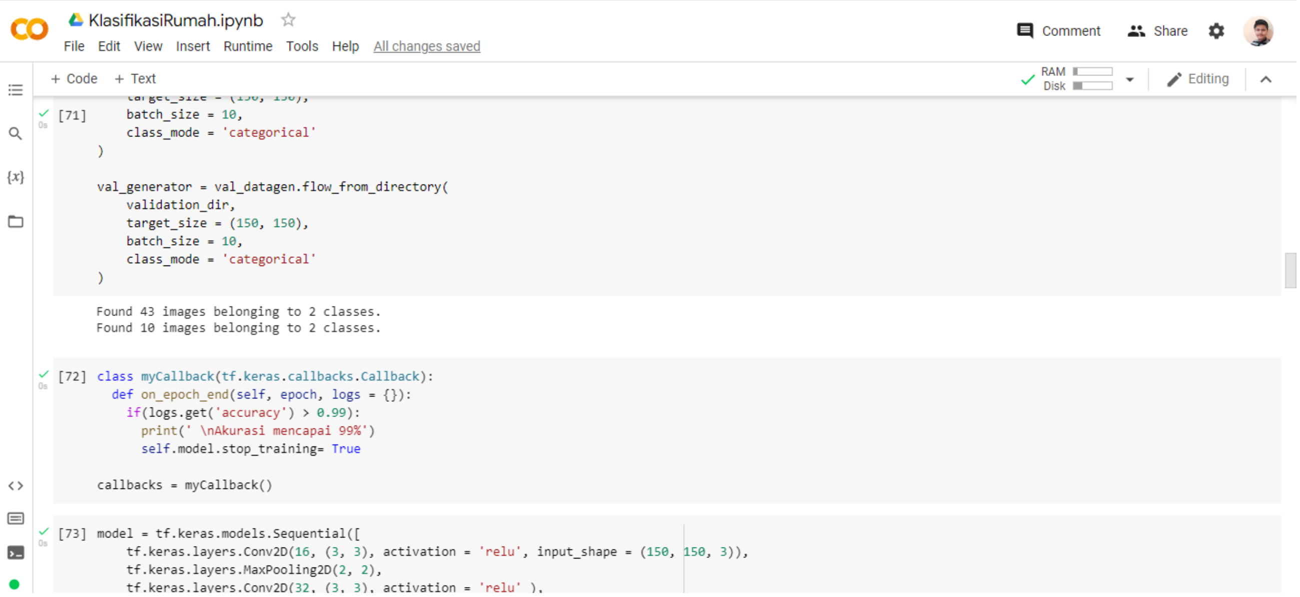
```
[67]   train_size = int(train_ratio * total_size)
       val_size = total_size - train_size

       randomized = random.sample(os.listdir(source), total_size)
       train_files = randomized[0:train_size]
       val_files = randomized[train_size:total_size]

       for i in train_files:
         i_file = source + i
         destination = train + i
         copyfile(i_file, destination)

         for i in val_files:
           i_files = source + i
           destination = val + i
           copyfile(i_file, destination)

     #jumlah pembagian data training dan testing
     train_ratio = 0.9

     #pembagian Training dan Validasi
     # Training
     source_00 = classic_dir
     train_00 = train_classic
```



```
[67] val_00 = validation_classic
     train_val_split(source_00, train_00, val_00, train_ratio)

     # Validasi
     source_01 = modern_dir
     train_01 = train_modern
     val_01 = validation_modern
     train_val_split(source_01, train_01, val_01, train_ratio)

[68] print('Jumlah All classic :', len(os.listdir(classic_dir)))
     print('Jumlah Train classic :', len(os.listdir(train_classic)))
     print('Jumlah Val classic :', len(os.listdir(validation_classic)))

     Jumlah All classic : 25
     Jumlah Train classic : 23
     Jumlah Val classic : 5

[69] import tensorflow as tf
     from tensorflow.keras.optimizers import Adam
     from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

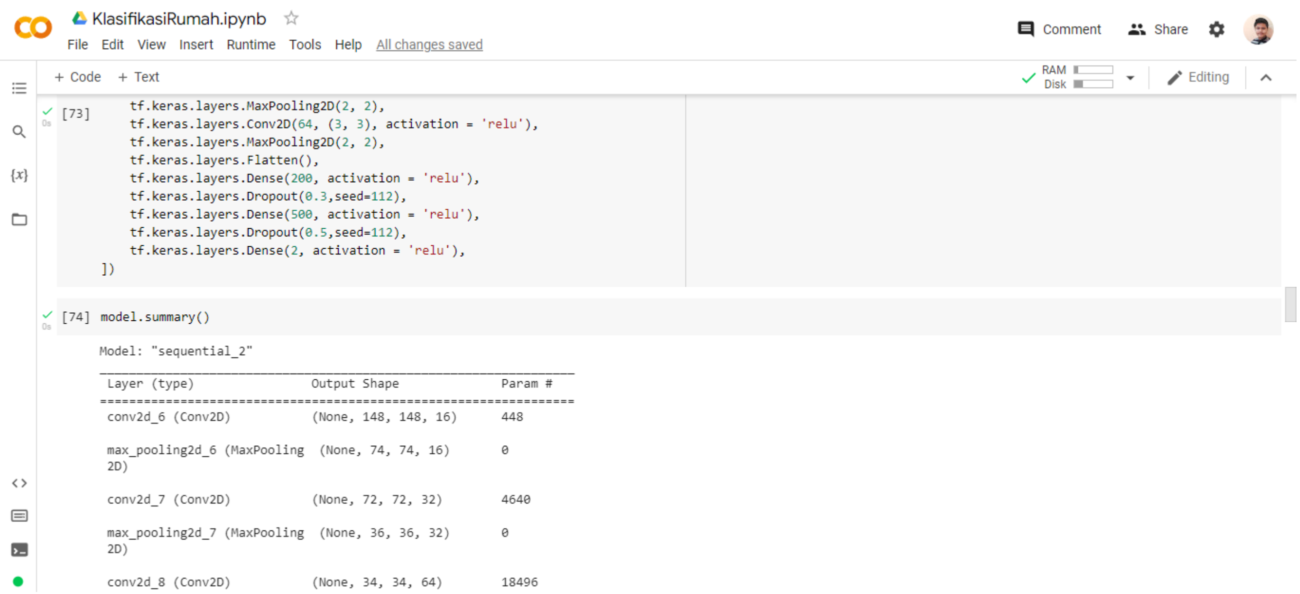


```
[70] train_datagen = ImageDataGenerator(
         rescale = 1./255,
         rotation_range = 30,
         horizontal_flip = True,
         shear_range = 0.3,
         fill_mode = 'nearest',
         width_shift_range = 0.2,
         height_shift_range = 0.2,
         zoom_range = 0.1
     )

     val_datagen = ImageDataGenerator(
         rescale = 1./255,
         rotation_range = 30,
         horizontal_flip = True,
         shear_range = 0.3,
         fill_mode = 'nearest',
         width_shift_range = 0.2,
         height_shift_range = 0.2,
         zoom_range = 0.1
     )

[71] train_generator = train_datagen.flow_from_directory(
         train_dir,
         target_size = (150, 150),
```

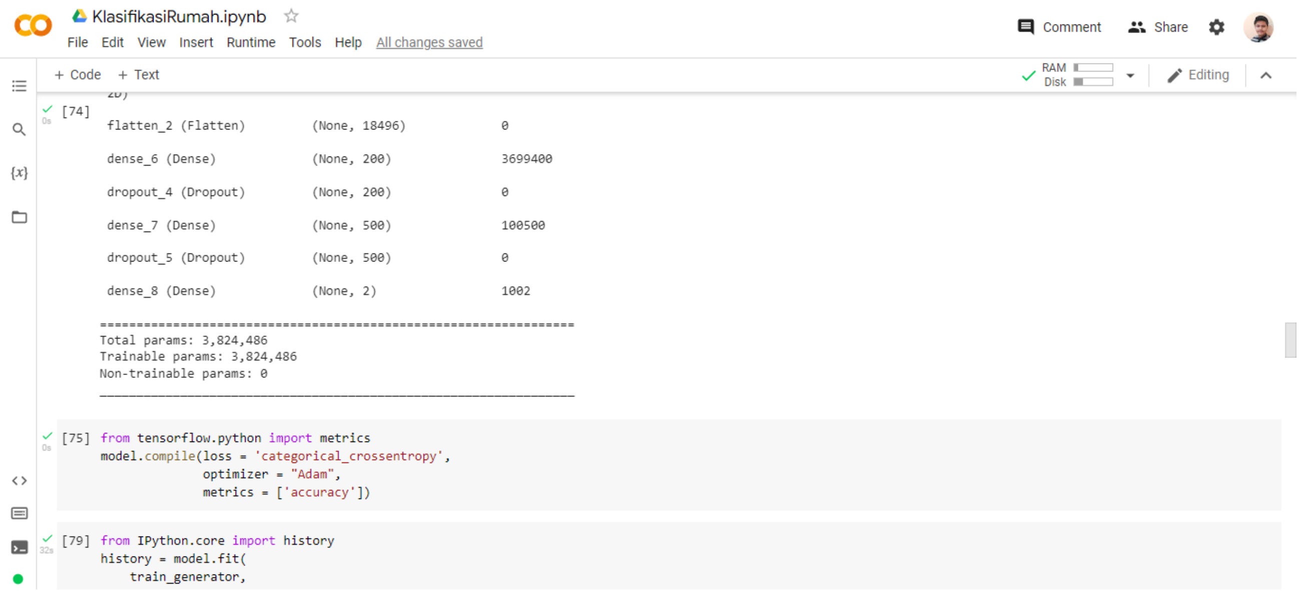


```
[71]     batch_size = 10,
         class_mode = 'categorical'
     )

     val_generator = val_datagen.flow_from_directory(
         validation_dir,
         target_size = (150, 150),
         batch_size = 10,
         class_mode = 'categorical'
     )

     Found 43 images belonging to 2 classes.
     Found 10 images belonging to 2 classes.

[72] class myCallback(tf.keras.callbacks.Callback):
       def on_epoch_end(self, epoch, logs = {}):
         if(logs.get('accuracy') > 0.99):
           print(' \nAkurasi mencapai 99%')
           self.model.stop_training= True

     callbacks = myCallback()

[73] model = tf.keras.models.Sequential([
         tf.keras.layers.Conv2D(16, (3, 3), activation = 'relu', input_shape = (150, 150, 3)),
         tf.keras.layers.MaxPooling2D(2, 2),
         tf.keras.layers.Conv2D(32, (3, 3), activation = 'relu' ),
```

Activation yang digunakan adalah relu



```
[73]     tf.keras.layers.MaxPooling2D(2, 2),
         tf.keras.layers.Conv2D(64, (3, 3), activation = 'relu'),
         tf.keras.layers.MaxPooling2D(2, 2),
         tf.keras.layers.Flatten(),
         tf.keras.layers.Dense(200, activation = 'relu'),
         tf.keras.layers.Dropout(0.3,seed=112),
         tf.keras.layers.Dense(500, activation = 'relu'),
         tf.keras.layers.Dropout(0.5,seed=112),
         tf.keras.layers.Dense(2, activation = 'relu'),
     ])

[74] model.summary()

     Model: "sequential_2"
     _________________________________________________________________
      Layer (type)                Output Shape              Param #
     =================================================================
      conv2d_6 (Conv2D)           (None, 148, 148, 16)      448

      max_pooling2d_6 (MaxPooling  (None, 74, 74, 16)       0
      2D)

      conv2d_7 (Conv2D)           (None, 72, 72, 32)        4640

      max_pooling2d_7 (MaxPooling  (None, 36, 36, 32)       0
      2D)

      conv2d_8 (Conv2D)           (None, 34, 34, 64)        18496
```

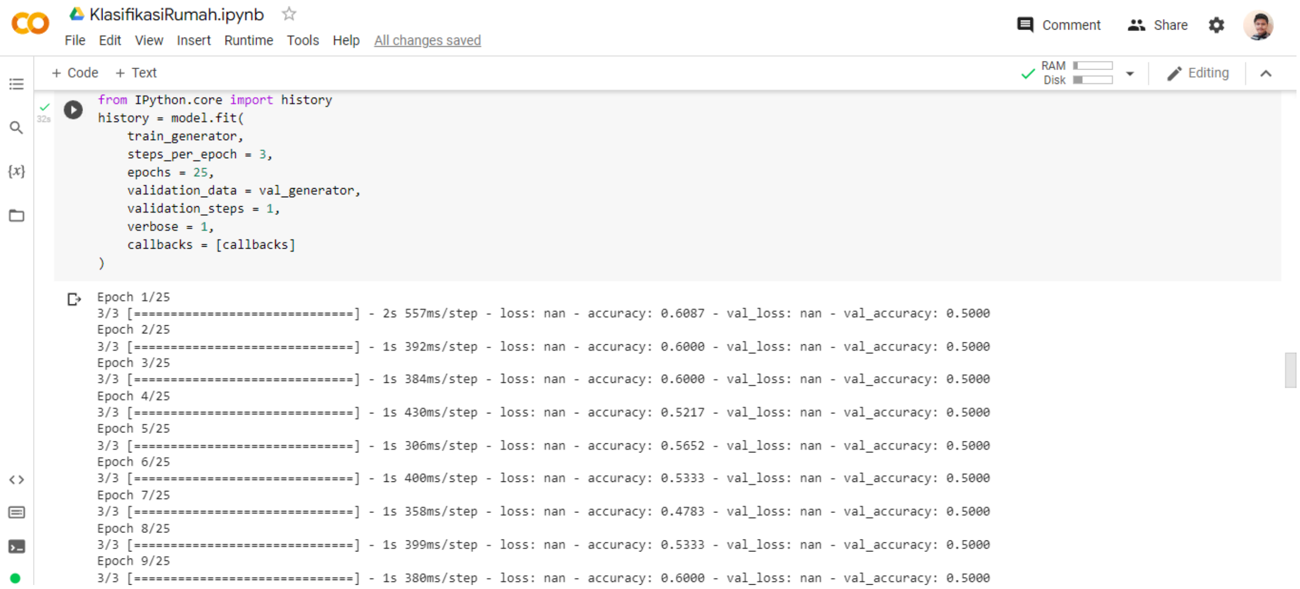


```
[74]
 flatten_2 (Flatten)          (None, 18496)             0

 dense_6 (Dense)              (None, 200)               3699400

 dropout_4 (Dropout)          (None, 200)               0

 dense_7 (Dense)              (None, 500)               100500

 dropout_5 (Dropout)          (None, 500)               0

 dense_8 (Dense)              (None, 2)                 1002

=================================================================
Total params: 3,824,486
Trainable params: 3,824,486
Non-trainable params: 0
_________________________________________________________________
```

```
[75] from tensorflow.python import metrics
     model.compile(loss = 'categorical_crossentropy',
                   optimizer = "Adam",
                   metrics = ['accuracy'])
```

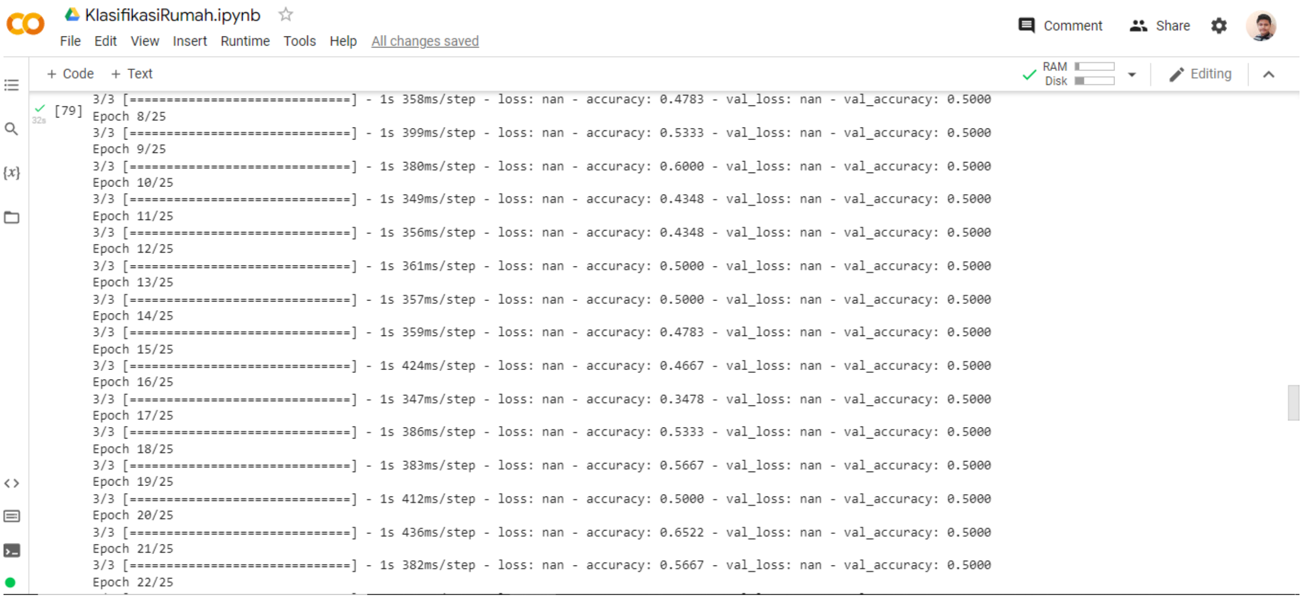
```
[79] from IPython.core import history
     history = model.fit(
         train_generator,
```

Epoch yang dipakai adalah 25 dan staps per epoch adalah 3

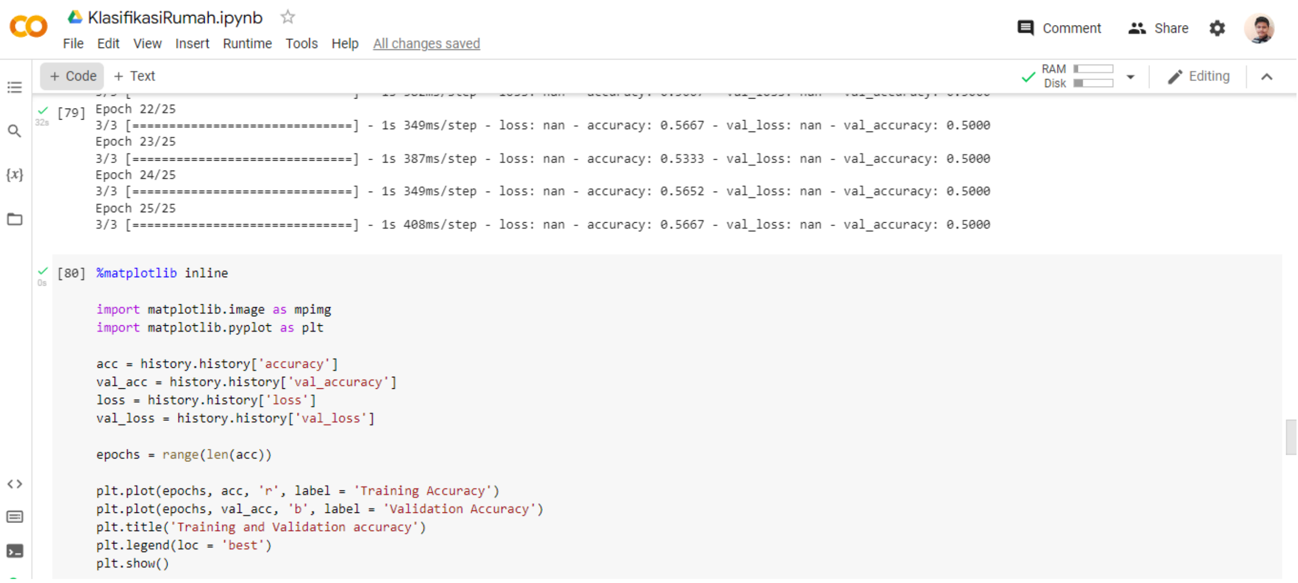


```
from IPython.core import history
history = model.fit(
    train_generator,
    steps_per_epoch = 3,
    epochs = 25,
    validation_data = val_generator,
    validation_steps = 1,
    verbose = 1,
    callbacks = [callbacks]
)
```

```
Epoch 1/25
3/3 [==============================] - 2s 557ms/step - loss: nan - accuracy: 0.6087 - val_loss: nan - val_accuracy: 0.5000
Epoch 2/25
3/3 [==============================] - 1s 392ms/step - loss: nan - accuracy: 0.6000 - val_loss: nan - val_accuracy: 0.5000
Epoch 3/25
3/3 [==============================] - 1s 384ms/step - loss: nan - accuracy: 0.6000 - val_loss: nan - val_accuracy: 0.5000
Epoch 4/25
3/3 [==============================] - 1s 430ms/step - loss: nan - accuracy: 0.5217 - val_loss: nan - val_accuracy: 0.5000
Epoch 5/25
3/3 [==============================] - 1s 306ms/step - loss: nan - accuracy: 0.5652 - val_loss: nan - val_accuracy: 0.5000
Epoch 6/25
3/3 [==============================] - 1s 400ms/step - loss: nan - accuracy: 0.5333 - val_loss: nan - val_accuracy: 0.5000
Epoch 7/25
3/3 [==============================] - 1s 358ms/step - loss: nan - accuracy: 0.4783 - val_loss: nan - val_accuracy: 0.5000
Epoch 8/25
3/3 [==============================] - 1s 399ms/step - loss: nan - accuracy: 0.5333 - val_loss: nan - val_accuracy: 0.5000
Epoch 9/25
3/3 [==============================] - 1s 380ms/step - loss: nan - accuracy: 0.6000 - val_loss: nan - val_accuracy: 0.5000
```



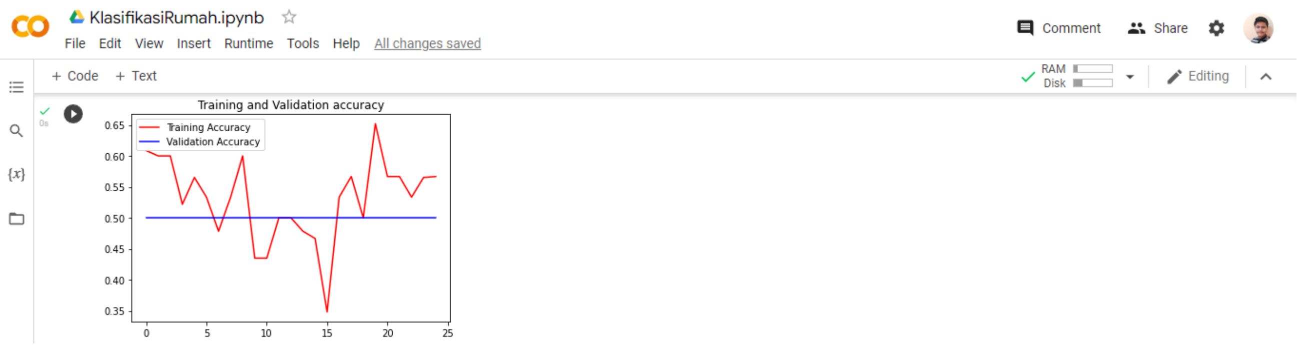
```
[79]
3/3 [==============================] - 1s 358ms/step - loss: nan - accuracy: 0.4783 - val_loss: nan - val_accuracy: 0.5000
Epoch 8/25
3/3 [==============================] - 1s 399ms/step - loss: nan - accuracy: 0.5333 - val_loss: nan - val_accuracy: 0.5000
Epoch 9/25
3/3 [==============================] - 1s 380ms/step - loss: nan - accuracy: 0.6000 - val_loss: nan - val_accuracy: 0.5000
Epoch 10/25
3/3 [==============================] - 1s 349ms/step - loss: nan - accuracy: 0.4348 - val_loss: nan - val_accuracy: 0.5000
Epoch 11/25
3/3 [==============================] - 1s 356ms/step - loss: nan - accuracy: 0.4348 - val_loss: nan - val_accuracy: 0.5000
Epoch 12/25
3/3 [==============================] - 1s 361ms/step - loss: nan - accuracy: 0.5000 - val_loss: nan - val_accuracy: 0.5000
Epoch 13/25
3/3 [==============================] - 1s 357ms/step - loss: nan - accuracy: 0.5000 - val_loss: nan - val_accuracy: 0.5000
Epoch 14/25
3/3 [==============================] - 1s 359ms/step - loss: nan - accuracy: 0.4783 - val_loss: nan - val_accuracy: 0.5000
Epoch 15/25
3/3 [==============================] - 1s 424ms/step - loss: nan - accuracy: 0.4667 - val_loss: nan - val_accuracy: 0.5000
Epoch 16/25
3/3 [==============================] - 1s 347ms/step - loss: nan - accuracy: 0.3478 - val_loss: nan - val_accuracy: 0.5000
Epoch 17/25
3/3 [==============================] - 1s 386ms/step - loss: nan - accuracy: 0.5333 - val_loss: nan - val_accuracy: 0.5000
Epoch 18/25
3/3 [==============================] - 1s 383ms/step - loss: nan - accuracy: 0.5667 - val_loss: nan - val_accuracy: 0.5000
Epoch 19/25
3/3 [==============================] - 1s 412ms/step - loss: nan - accuracy: 0.5000 - val_loss: nan - val_accuracy: 0.5000
Epoch 20/25
3/3 [==============================] - 1s 436ms/step - loss: nan - accuracy: 0.6522 - val_loss: nan - val_accuracy: 0.5000
Epoch 21/25
3/3 [==============================] - 1s 382ms/step - loss: nan - accuracy: 0.5667 - val_loss: nan - val_accuracy: 0.5000
Epoch 22/25
```

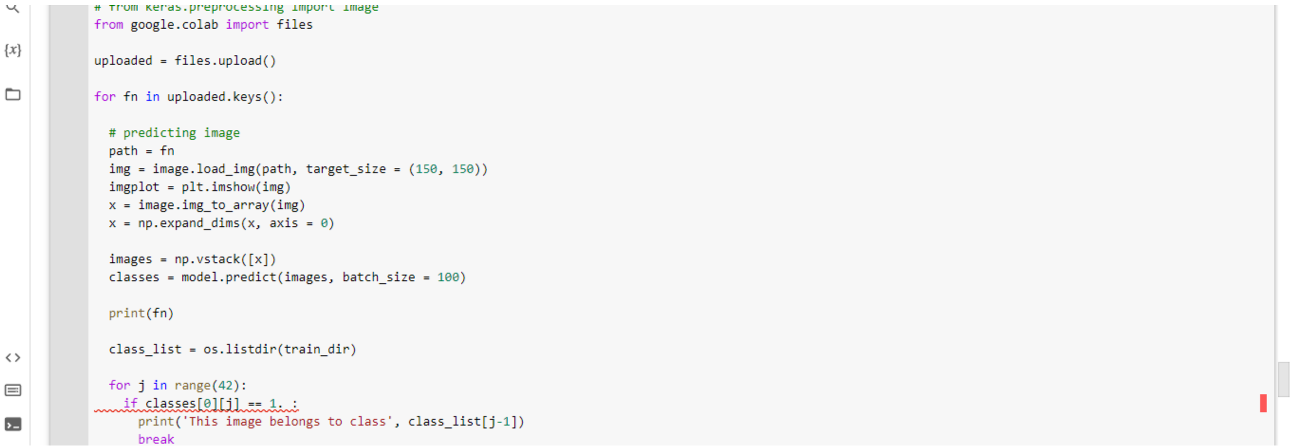


```
Epoch 22/25
3/3 [==============================] - 1s 349ms/step - loss: nan - accuracy: 0.5667 - val_loss: nan - val_accuracy: 0.5000
Epoch 23/25
3/3 [==============================] - 1s 387ms/step - loss: nan - accuracy: 0.5333 - val_loss: nan - val_accuracy: 0.5000
Epoch 24/25
3/3 [==============================] - 1s 349ms/step - loss: nan - accuracy: 0.5652 - val_loss: nan - val_accuracy: 0.5000
Epoch 25/25
3/3 [==============================] - 1s 408ms/step - loss: nan - accuracy: 0.5667 - val_loss: nan - val_accuracy: 0.5000
```

```
%matplotlib inline

import matplotlib.image as mpimg
import matplotlib.pyplot as plt

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label = 'Training Accuracy')
plt.plot(epochs, val_acc, 'b', label = 'Validation Accuracy')
plt.title('Training and Validation accuracy')
plt.legend(loc = 'best')
plt.show()
```

Hasil Akurasi Grafik Training dan Validasi





```
# from keras.preprocessing import image
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():

  # predicting image
  path = fn
  img = image.load_img(path, target_size = (150, 150))
  imgplot = plt.imshow(img)
  x = image.img_to_array(img)
  x = np.expand_dims(x, axis = 0)

  images = np.vstack([x])
  classes = model.predict(images, batch_size = 100)

  print(fn)

  class_list = os.listdir(train_dir)

  for j in range(42):
    if classes[0][j] == 1. :
      print('This image belongs to class', class_list[j-1])
      break
```

0
20
40
60
80
100
120