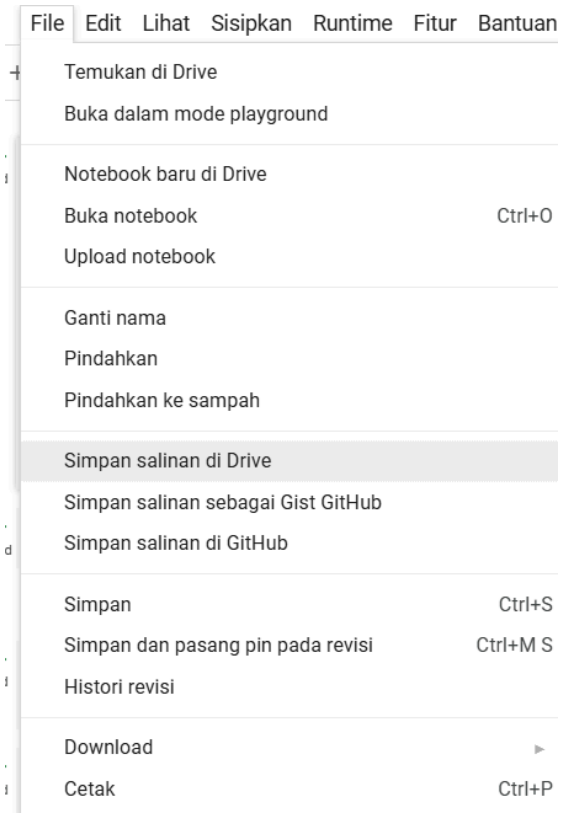
Kelompok Nangka - BMI Prediction Launching Guide
LINK GOOGLE COLAB:
https://colab.research.google.com/drive/1s2gerl_n4-Bw4RRUV7dZOT5sPd4GH4rr?usp=sharing
LINK GOOGLE DRIVE DATASET:
https://drive.google.com/drive/folders/1T_1Ie5g1y1zYlPHkD1jJlplhql0seIWW?usp=drive_link

Berikut adalah langkah-langkah agar dapat run program BMI Prediction:

1. Sign in ke Google Colab agar dapat mengakses source codenya
2. Buat salinan code, penamaan bebas yang penting jadi terbentuk notebook baru dengan isi source code



3. Copy isi dari setiap data yang ada di link Google Drive yang telah diberikan ke Google Drive pribadi.
4. Setelah semua berhasil di save berdasarkan urutan folder yang ada, coba run setiap code yang ada di Google Colab. Apabila terjadi kendala, coba refresh di Google Drive atau start run dari paling atas kembali. Run semua code sampai muncul tanda centang seperti ini

```
from google.colab import drive
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
from tensorflow.keras.preprocessing import image
```

```
[3]  drive.mount ('/content/drive')
```

```
Mounted at /content/drive
```

```
[4]  dataset_directory = '/content/drive/MyDrive/AOL_AI_BMI'
     # os.listdir(dataset_directory)
```

```
[5]  training_directory = os.path.join(dataset_directory, 'datatraining')
     testing_directory = os.path.join(dataset_directory, 'datatest')
     # os.listdir(training_directory)
     # os.listdir(testing_directory)
```

```
[6]  image_width, image_height = 250,250
     batch_size = 32
```

Output yang dihasilkan setelah semua di run:


```
from google.colab import drive
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
from tensorflow.keras.preprocessing import image
```

```
[3]  drive.mount ('/content/drive')
```

```
Mounted at /content/drive
```

```
[4]  dataset_directory = '/content/drive/MyDrive/AOL_AI_BMI'
     # os.listdir(dataset_directory)
```

```
[5]  training_directory = os.path.join(dataset_directory, 'datatraining')
     testing_directory = os.path.join(dataset_directory, 'datatest')
     # os.listdir(training_directory)
     # os.listdir(testing_directory)
```

```
image_width, image_height = 250,250
batch_size = 32

# untuk training saat gambarnya diotakatik juga
train_generator = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True

)
test_generator = ImageDataGenerator(rescale=1.0/255.0)

#generate train data from drive
train_datagenerator = train_generator.flow_from_directory(
    training_directory,
    target_size=(image_width, image_height),
    batch_size=batch_size,
    class_mode='categorical' #karena pake foto, klo pake numeric sparse
)

#generate test data from drive
test_datagenerator = test_generator.flow_from_directory(
    testing_directory,
    target_size=(image_width, image_height),
    batch_size=batch_size,
    class_mode='categorical'
)
```

```
Found 24 images belonging to 3 classes.
Found 6 images belonging to 3 classes.
```

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(image_height, image_width, 3)),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(3, activation='softmax')  # 3 neurons for 3 classes
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarnin
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 248, 248, 32)	896
max_pooling2d (MaxPooling2D)	(None, 124, 124, 32)	0
conv2d_1 (Conv2D)	(None, 122, 122, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 61, 61, 64)	0
flatten (Flatten)	(None, 238144)	0
dense (Dense)	(None, 128)	30,482,560
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 3)	387

Total params: 30,502,339 (116.36 MB)
Trainable params: 30,502,339 (116.36 MB)
Non-trainable params: 0 (0.00 B)

```
history = model.fit(
    train_datagenerator,
    steps_per_epoch=train_datagenerator.samples // batch_size,
    epochs=10,
    validation_data=test_datagenerator,
    validation_steps=test_datagenerator.samples // batch_size
)
```

```
Epoch 1/10
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:122: UserWarning: Y
  self._warn_if_super_not_called()
1/1 ━━━━━━━━━━━━━━━━━━━━ 8s 8s/step - accuracy: 0.4167 - loss: 1.0773 - val_accuracy: 0.3333 - val_loss: 9.7405
Epoch 2/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 8s 8s/step - accuracy: 0.2500 - loss: 18.0924 - val_accuracy: 0.3333 - val_loss: 18.3911
Epoch 3/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step - accuracy: 0.4167 - loss: 19.5450 - val_accuracy: 0.3333 - val_loss: 11.554
Epoch 4/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 6s 6s/step - accuracy: 0.3750 - loss: 13.4732 - val_accuracy: 0.3333 - val_loss: 7.3731
Epoch 5/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step - accuracy: 0.3333 - loss: 9.2235 - val_accuracy: 0.3333 - val_loss: 2.7952
Epoch 6/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 6s 6s/step - accuracy: 0.4167 - loss: 4.1744 - val_accuracy: 0.1667 - val_loss: 1.1730
Epoch 7/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step - accuracy: 0.4167 - loss: 2.1674 - val_accuracy: 0.3333 - val_loss: 1.8370
Epoch 8/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step - accuracy: 0.4167 - loss: 1.6364 - val_accuracy: 0.3333 - val_loss: 1.4338
Epoch 9/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 5s 5s/step - accuracy: 0.4167 - loss: 1.3628 - val_accuracy: 0.3333 - val_loss: 1.0706
Epoch 10/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 5s 5s/step - accuracy: 0.5833 - loss: 0.8202 - val_accuracy: 0.5000 - val_loss: 0.9575
```

```
#debugging
import os
from PIL import Image

# Replace with your dataset directory
dataset_dir = training_directory  # or val_dir

# Walk through the dataset
for root, dirs, files in os.walk(dataset_dir):
    for file in files:
        file_path = os.path.join(root, file)
        try:
            # Attempt to open the image
            img = Image.open(file_path)
            img.verify()  # Verify if it's a valid image
        except Exception as e:
            print(f"Error with file: {file_path} -> {e}")
```

```
[10] validation_loss, validation_accuracy = model.evaluate(test_datagenerator)
print(f"Validation Loss: {validation_loss}")
print(f"Validation Accuracy: {validation_accuracy * 100:.2f}%")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 259ms/step - accuracy: 0.5000 - loss: 0.9575
Validation Loss: 0.9575257301330566
Validation Accuracy: 50.00%
```

```
image_path = '/content/drive/My Drive/AOL_AI_BMI/dataprediction/prediction2.jpg'

img = mpimg.imread(image_path)
imgplot = plt.imshow(img)
plt.show()

img = image.load_img(image_path, target_size=(image_width, image_height))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)

predictions = model.predict(img_array)
predicted_class = np.argmax(predictions)

class_labels = ['Normal', 'Overweight', 'Underweight']
predicted_label = class_labels[predicted_class]

print(f"Predicted Class: {predicted_label}")

if predicted_label == 'Underweight':
  print('bmi < 18.5')
elif predicted_class == 'Normal':
  print('bmi = 18.5 - 24.9')
elif predicted_class == 'Overweight':
  print('bmi > 25.0')
else:
  print('Obesity')
```

1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 114ms/step
Predicted Class: Underweight
bmi < 18.5