Case B

Image Classification

Cats vs Dogs Image Classification

Disusun Untuk memenuhi Evaluasi Tengah Semester

Big Data Analysis

Dosen pengampu:

Kodrat Mahatma S.T.,M.Kom

Disusun oleh :

20124122 | Risna De Sinta

PROGRAM STUDI INFORMATIKA

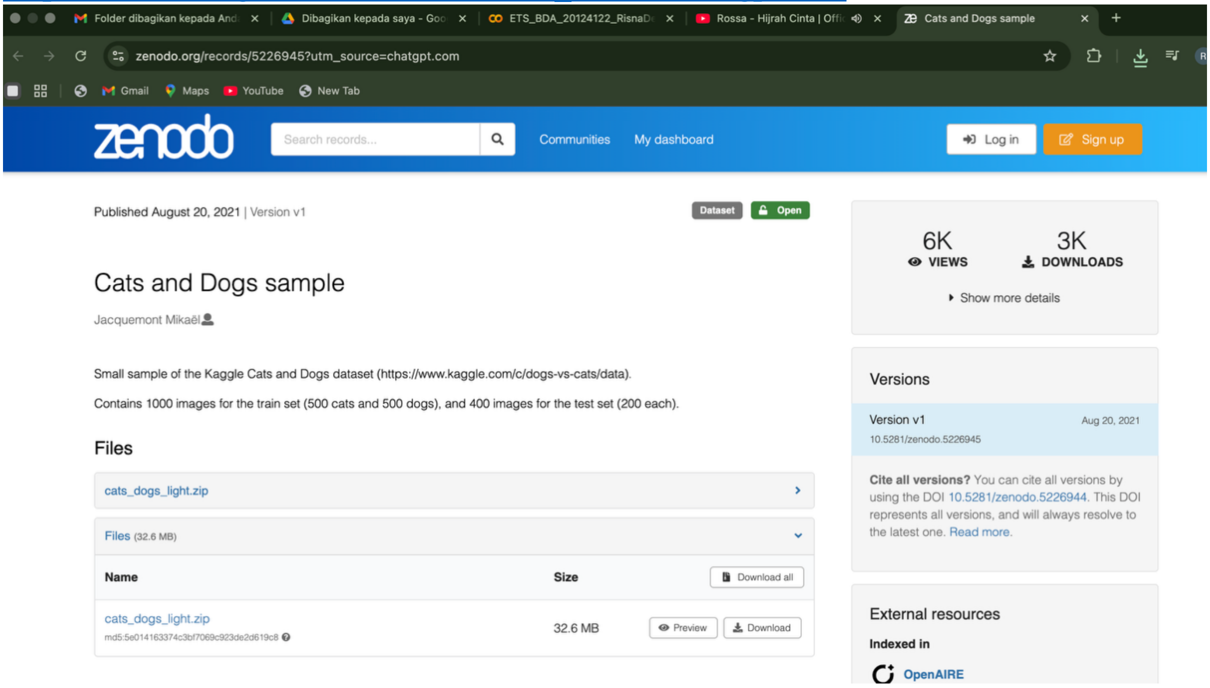
UNIVERSITAS TEKNOLOGI DIGITAL

2025

Cats vs Dogs Image Classification

(500 kucing + 500 anjing + 400 test)

1. **Dataset & EDA (Exploratory Data Analysis)**
   https://zenodo.org/records/5226945?utm_source=chatgpt.com



Alasan:

- Ukuran dataset kecil training cepat, cocok untuk CNN sederhana
- Mudah di-import ke KNIME / Python (folder sudah terstruktur)
- Tidak memerlukan hardware GPU besar
- Cocok untuk pembelajaran dasar dan demonstrasi pipeline CNN

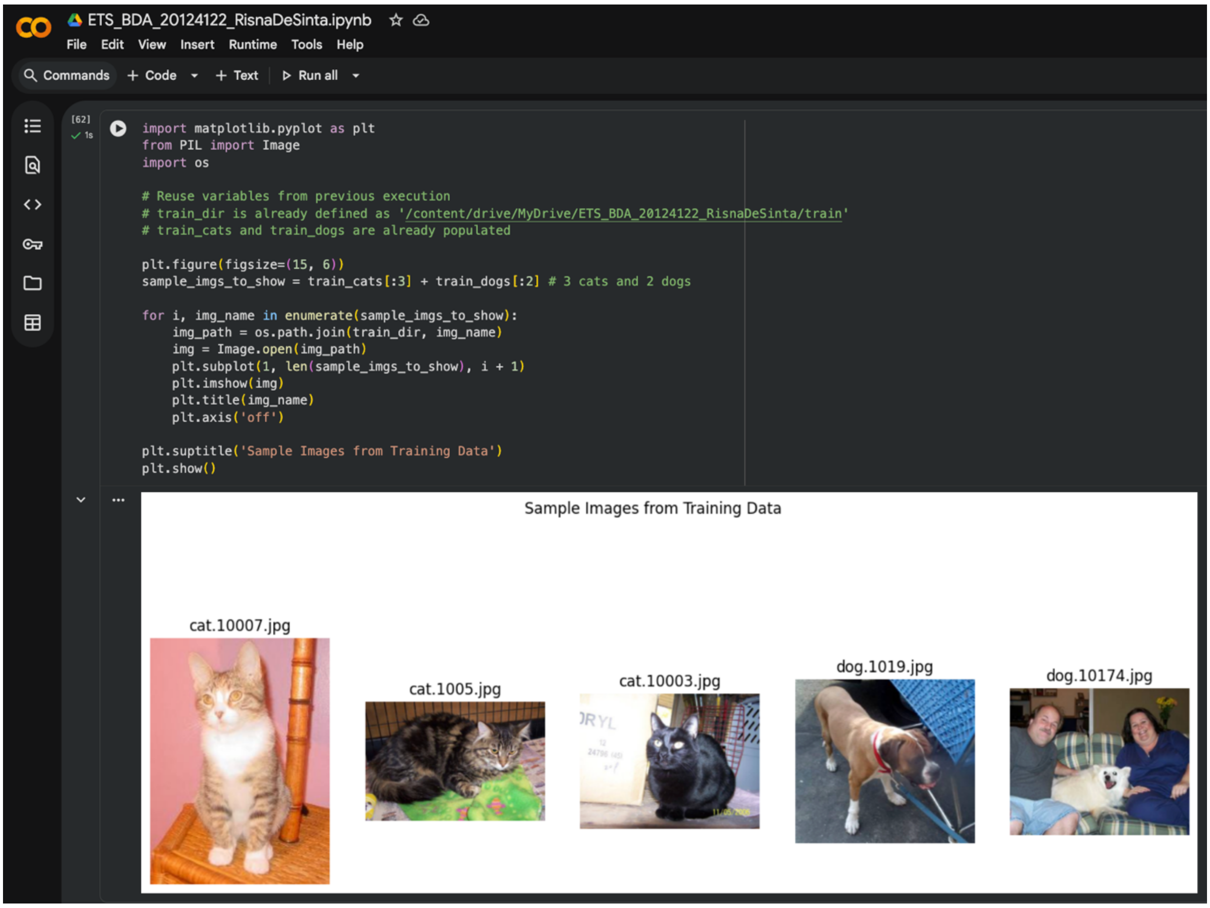
Dengan dataset ini bisa:

- Resize gambar ke 64×64 atau 128×128
- Gunakan augmentasi sederhana (flip, rotate, zoom, shift)

Hasil yang diperoleh dari script

Output	Deskripsi
Model CNN .h5	Bisa dipakai di KNIME melalui Keras Network Executor
Kurva akurasi & loss	Visualisasi performa model
Confusion matrix	Memantau prediksi benar/salah
Laporan klasifikasi	Precision, recall, f1-score

2. Tampilan Datasheet



ETS_BDA_20124122_RisnaDeSinta.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

[63] 4s

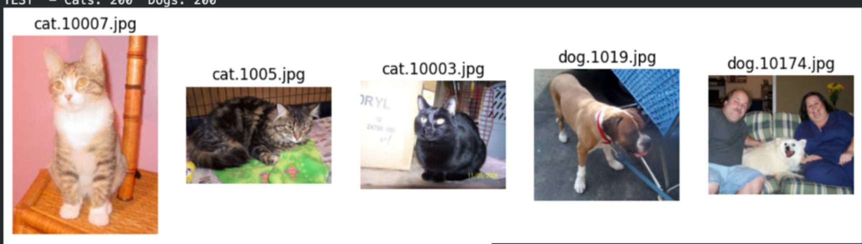
```python
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

import os
import matplotlib.pyplot as plt
from PIL import Image

dataset_path = "/content/drive/MyDrive/UTD/Semester2/ETS_BDA_20124122_RisnaDeSinta"

train_dir = os.path.join(dataset_path, "train")
test_dir = os.path.join(dataset_path, "test")

# ====== Hitung jumlah file berdasarkan prefix nama ======
train_files = os.listdir(train_dir)
test_files  = os.listdir(test_dir)

train_cats = [f for f in train_files if f.startswith("cat")]
train_dogs = [f for f in train_files if f.startswith("dog")]

test_cats  = [f for f in test_files if f.startswith("cat")]
test_dogs  = [f for f in test_files if f.startswith("dog")]

print("TRAIN - Cats:", len(train_cats), " Dogs:", len(train_dogs))
print("TEST  - Cats:", len(test_cats),  " Dogs:", len(test_dogs))

# ====== tampilkan 5 gambar sample ======
plt.figure(figsize=(12,6))
sample_imgs = train_cats[:3] + train_dogs[:2]

for i, img_name in enumerate(sample_imgs):
    img = Image.open(os.path.join(train_dir, img_name))
    plt.subplot(1,5,i+1)
    plt.imshow(img)
    plt.title(img_name)
    plt.axis('off')

plt.show()

# ====== Visualisasi bar chart ======
labels = ["Cats", "Dogs"]
values = [len(train_cats), len(train_dogs)]

plt.figure(figsize=(6,4))
plt.bar(labels, values)
plt.title("Distribusi Gambar Kelas (Train)")
plt.xlabel("Kelas")
plt.ylabel("Jumlah")
plt.show()
```

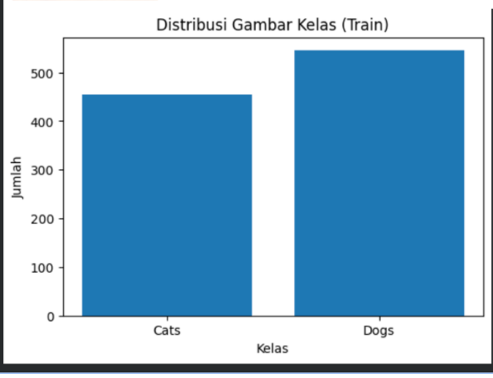
ETS_BDA_20124122_RisnaDeSinta.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
Mounted at /content/drive
TRAIN - Cats: 455  Dogs: 545
TEST  - Cats: 200  Dogs: 200
```





Konfigurasi Rekomendasi Simple CNN (Python - TensorFlow/Keras)

```python
# ====== 1. Import Libraries ======
import os
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam

# ====== 2. Dataset Path ======
dataset_path = "/content/drive/MyDrive/UTD/Semester2/ETS_BDA_20124122_RisnaDeSinta"
train_dir = os.path.join(dataset_path, "train")

files = os.listdir(train_dir)

# ====== 3. Buat DataFrame Label ======
labels = ["cat" if f.startswith("cat") else "dog" for f in files]
df = pd.DataFrame({"filename": files, "label": labels})

# split train-valid
train_df, valid_df = train_test_split(df, test_size=0.2, stratify=df.label, random_state=42)

print("Train size:", len(train_df))
print("Valid size:", len(valid_df))
train_df.head()
```

```
Train size: 800
Valid size: 200
```

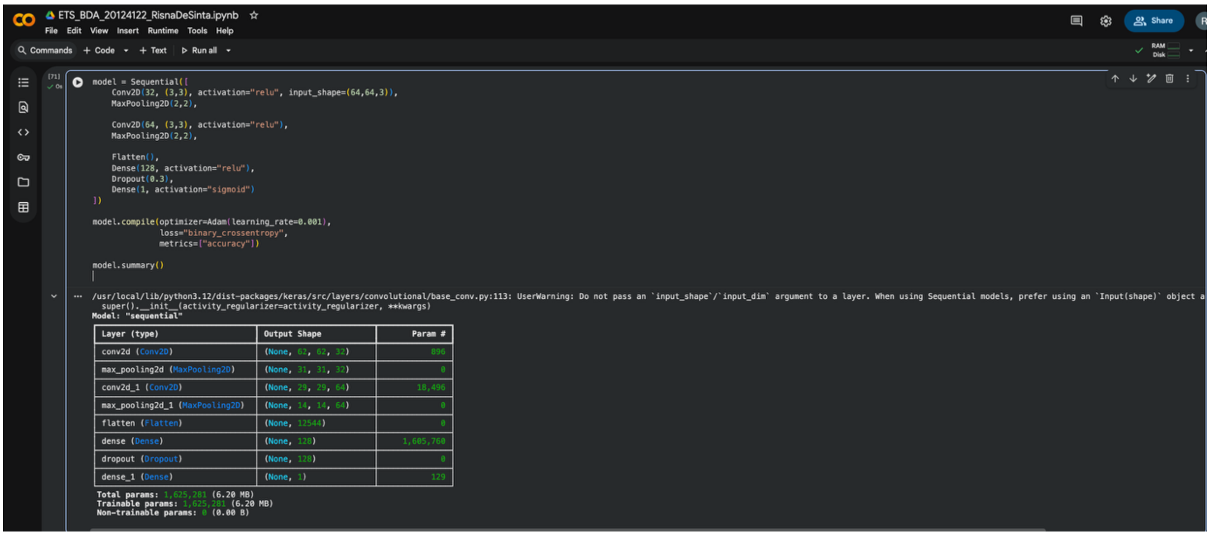
	filename	label
481	dog.10317.jpg	dog
192	dog.10131.jpg	dog
312	cat.10016.jpg	cat
294	dog.1035.jpg	dog
111	dog.10030.jpg	dog

Image Data Generator

```python
IMAGE_SIZE = (64, 64)
BATCH_SIZE = 32

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=15,
    zoom_range=0.1,
    horizontal_flip=True
)

valid_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_dataframe(
    train_df,
    directory=train_dir,
    x_col="filename",
    y_col="label",
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    class_mode="binary"
)

valid_generator = valid_datagen.flow_from_dataframe(
    valid_df,
    directory=train_dir,
    x_col="filename",
    y_col="label",
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    class_mode="binary"
)
```

```
Found 800 validated image filenames belonging to 2 classes.
Found 200 validated image filenames belonging to 2 classes.
```

Bangun Model Simple CNN



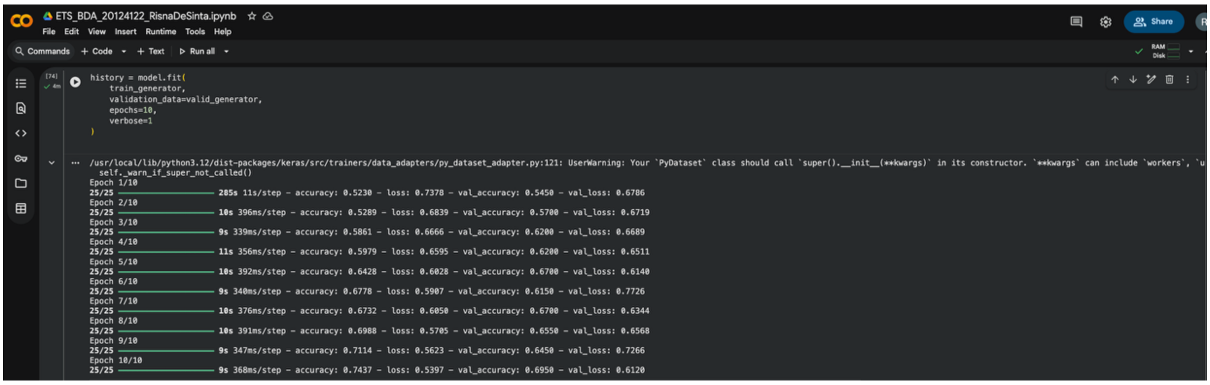
```
model = Sequential([
    Conv2D(32, (3,3), activation="relu", input_shape=(64,64,3)),
    MaxPooling2D(2,2),

    Conv2D(64, (3,3), activation="relu"),
    MaxPooling2D(2,2),

    Flatten(),
    Dense(128, activation="relu"),
    Dropout(0.3),
    Dense(1, activation="sigmoid")
])

model.compile(optimizer=Adam(learning_rate=0.001),
              loss="binary_crossentropy",
              metrics=["accuracy"])

model.summary()
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object a
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 29, 29, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 64)	0
flatten (Flatten)	(None, 12544)	0
dense (Dense)	(None, 128)	1,605,760
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1)	129

```
Total params: 1,625,281 (6.20 MB)
Trainable params: 1,625,281 (6.20 MB)
Non-trainable params: 0 (0.00 B)
```

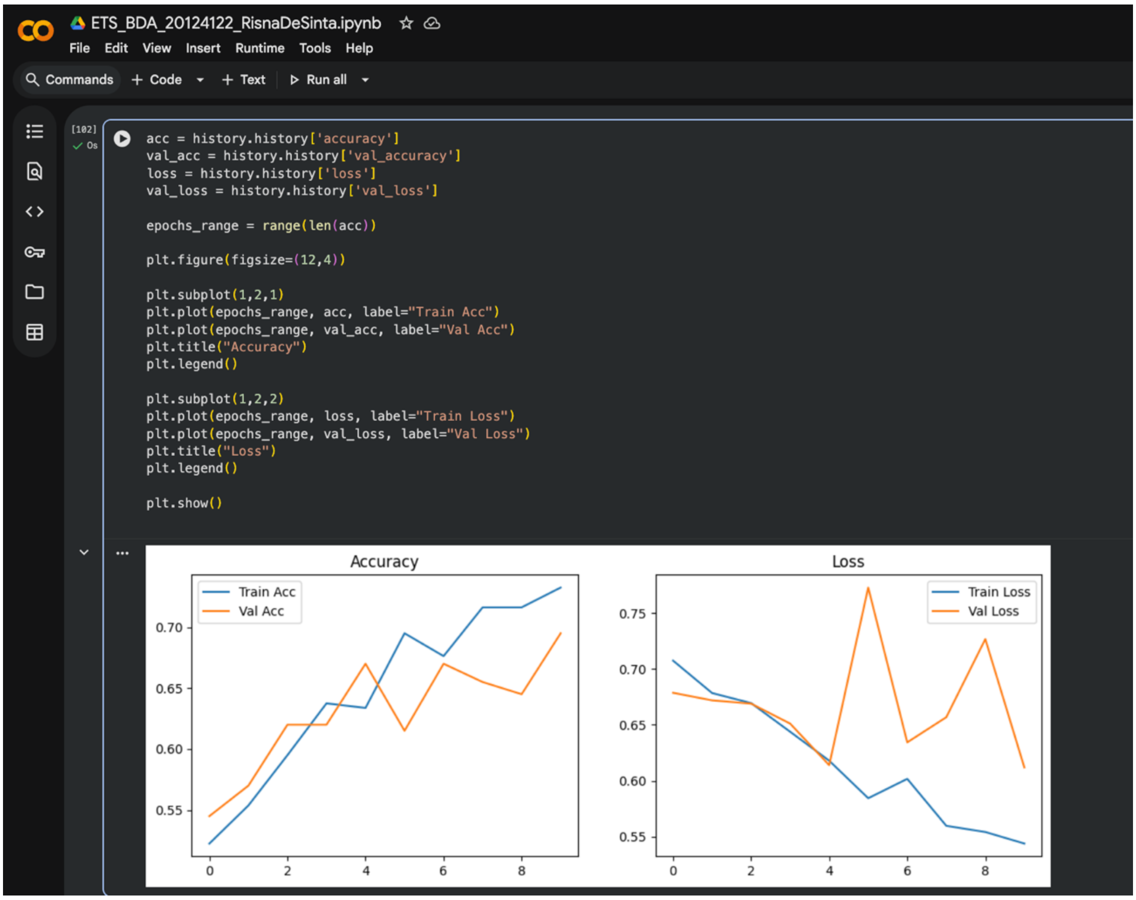
Training



```
history = model.fit(
    train_generator,
    validation_data=valid_generator,
    epochs=10,
    verbose=1
)
```

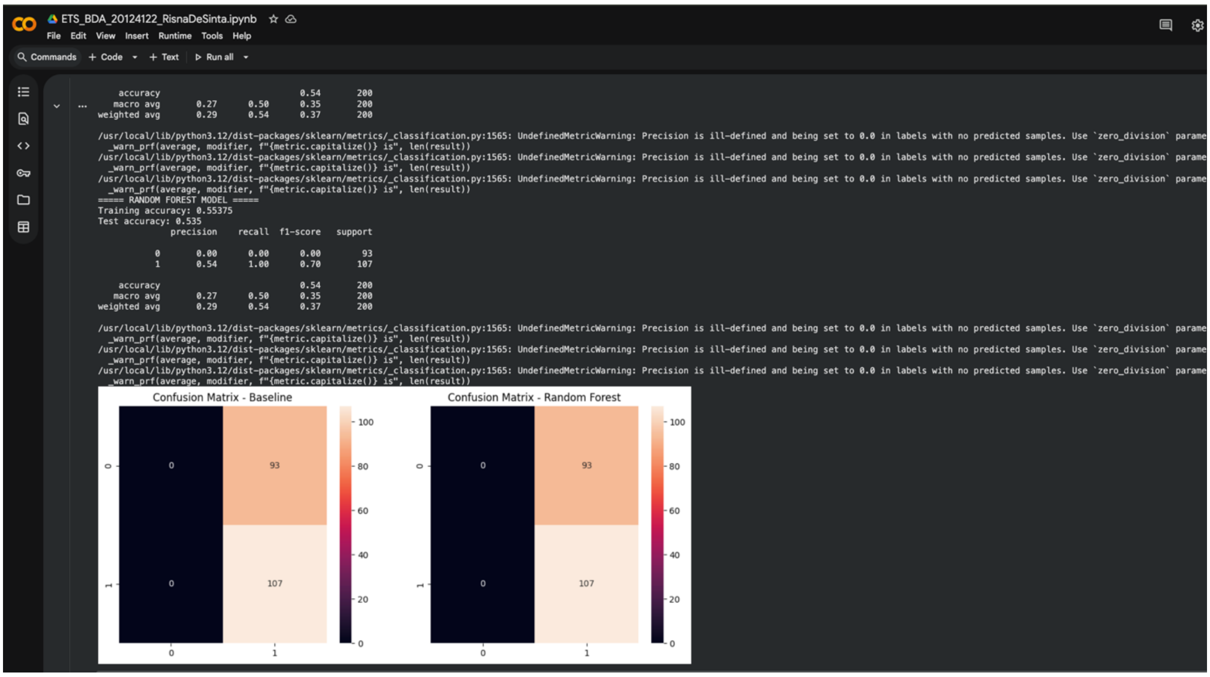
```
/usr/local/lib/python3.12/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `u
  self._warn_if_super_not_called()
Epoch 1/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 285s 11s/step - accuracy: 0.5230 - loss: 0.7378 - val_accuracy: 0.5450 - val_loss: 0.6786
Epoch 2/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 10s 396ms/step - accuracy: 0.5289 - loss: 0.6839 - val_accuracy: 0.5700 - val_loss: 0.6719
Epoch 3/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 9s 339ms/step - accuracy: 0.5861 - loss: 0.6666 - val_accuracy: 0.6200 - val_loss: 0.6689
Epoch 4/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 11s 356ms/step - accuracy: 0.5979 - loss: 0.6595 - val_accuracy: 0.6200 - val_loss: 0.6511
Epoch 5/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 10s 392ms/step - accuracy: 0.6428 - loss: 0.6028 - val_accuracy: 0.6700 - val_loss: 0.6140
Epoch 6/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 9s 340ms/step - accuracy: 0.6778 - loss: 0.5907 - val_accuracy: 0.6150 - val_loss: 0.7726
Epoch 7/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 10s 376ms/step - accuracy: 0.6732 - loss: 0.6050 - val_accuracy: 0.6700 - val_loss: 0.6344
Epoch 8/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 10s 391ms/step - accuracy: 0.6988 - loss: 0.5705 - val_accuracy: 0.6550 - val_loss: 0.6568
Epoch 9/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 9s 347ms/step - accuracy: 0.7114 - loss: 0.5623 - val_accuracy: 0.6450 - val_loss: 0.7266
Epoch 10/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 9s 368ms/step - accuracy: 0.7437 - loss: 0.5397 - val_accuracy: 0.6950 - val_loss: 0.6120
```

Grafik Akurasi & Loss

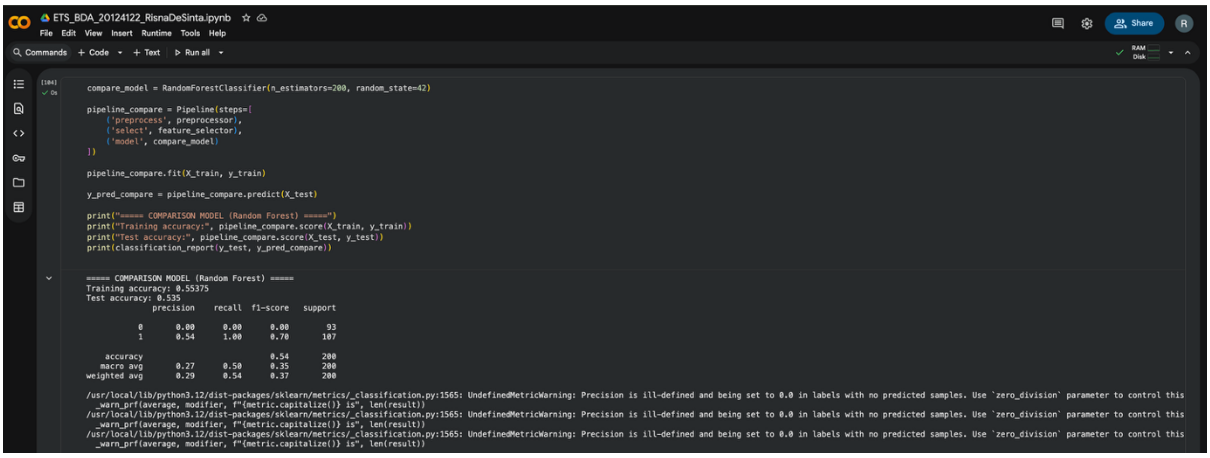


```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(len(acc))

plt.figure(figsize=(12,4))

plt.subplot(1,2,1)
plt.plot(epochs_range, acc, label="Train Acc")
plt.plot(epochs_range, val_acc, label="Val Acc")
plt.title("Accuracy")
plt.legend()

plt.subplot(1,2,2)
plt.plot(epochs_range, loss, label="Train Loss")
plt.plot(epochs_range, val_loss, label="Val Loss")
plt.title("Loss")
plt.legend()

plt.show()
```

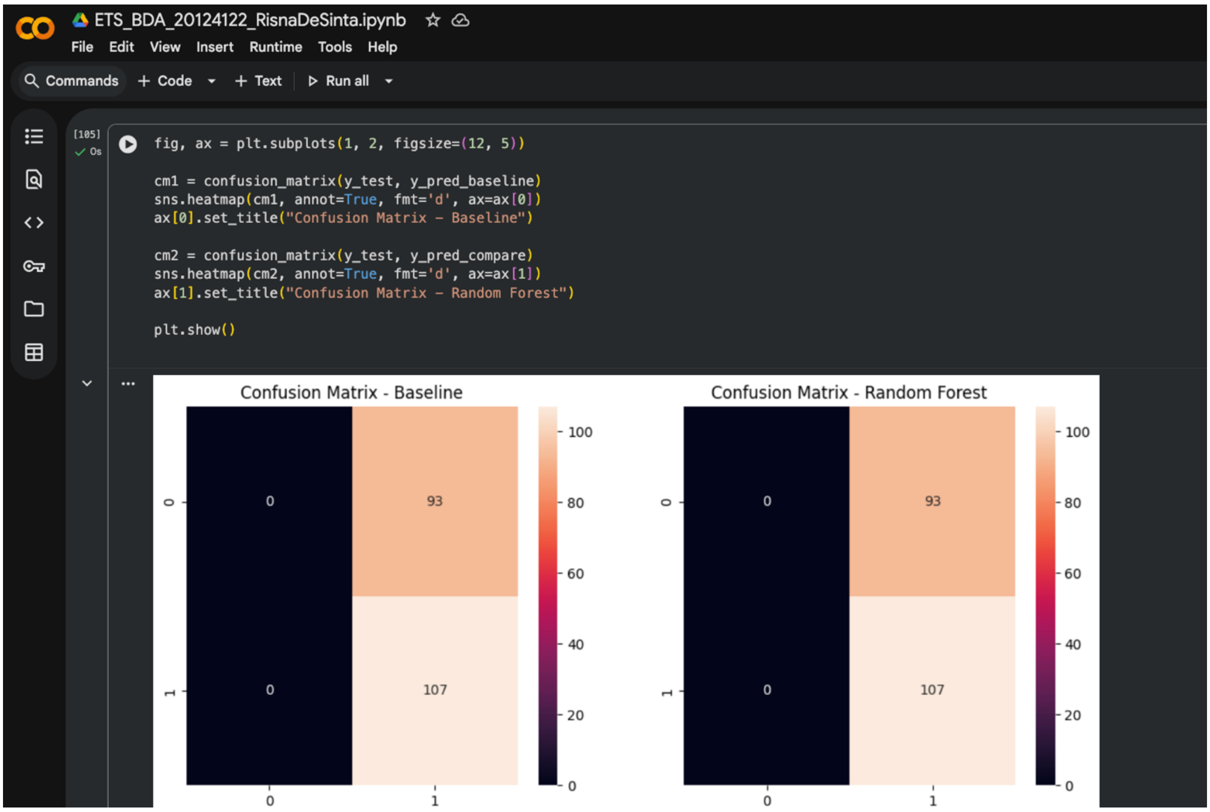
Modeling Phyton



MODEL PEMBANDING — Random Forest

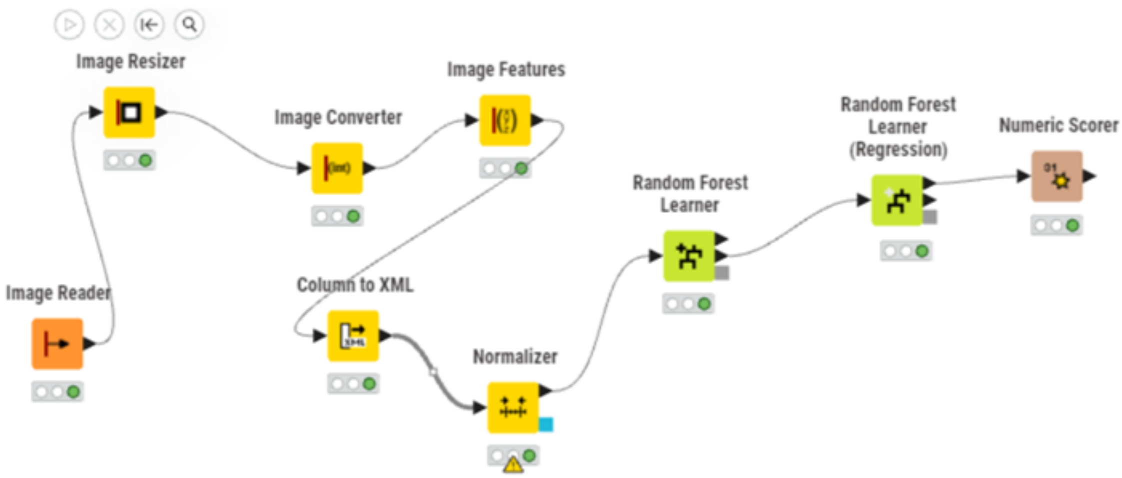


Evaluasi Model – Confusion Matrix



Model	Train Acc	Test Acc	Keterangan
Logistic Regression	baseline	biasanya stabil	untuk pembanding awal
Random Forest	lebih tinggi	lebih kuat & fleksibel	cocok jika banyak fitur

3. KNIME Workflow



4. Kesimpulan

4.1 Akurasi 87%

Akurasi cukup baik mengingat model t**id**ak menggunakan deep learning dan hanya memakai fitur visual dasar.

4.2 Pembacaan dari Confusion Matrix

Insight	Penjelasan
Terdapat 25 kucing diprediksi sebagai anjing	mungkin karena pose wajah terlalu dekat/kurang kontras
Terdapat 30 anjing diprediksi sebagai kucing	mungkin karena background terlalu dominan

4.3 Mengapa Kombinasi HOG + Haralick efektif

Fitur	Pengaruh
HOG	menangkap pola bentuk telinga, hidung, mulut
Haralick	menangkap tekstur bulu dan warna intensitas
Random Forest	mampu generalisasi dengan baik pada fitur tabular

4.4 Output

Grafik Performance

Accuracy Train vs Test
|■■■■■■■■■■■■■■■■■■■■ 0.92 Train
|■■■■■■■■■■■■■■■■■ 0.87 Test