

**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](https://zenodo.org/records/5226945?utm_source=chatgpt.com)

The screenshot shows a Zenodo dataset page. At the top, there's a navigation bar with tabs like 'Dataset' and 'Open'. Below the header, it says 'Published August 20, 2021 | Version v1'. The main title is 'Cats and Dogs sample' by 'Jacquemont Mikaël'. A brief description follows: 'Small sample of the Kaggle Cats and Dogs dataset (https://www.kaggle.com/c/dogs-vs-cats/data). Contains 1000 images for the train set (500 cats and 500 dogs), and 400 images for the test set (200 each.)'. On the left, there's a 'Files' section listing 'cats\_dogs\_light.zip' (32.6 MB) and a 'Files' folder. To the right, there are statistics: '6K VIEWS' and '3K DOWNLOADS'. Below these are sections for 'Versions' (listing 'Version v1' from Aug 20, 2021) and 'External resources' (listing 'Indexed in OpenAIRE').

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  $64 \times 64$  atau  $128 \times 128$
- 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

CO ETS\_BDA\_20124122\_RisnaDeSinta.ipynb ⌂

File Edit View Insert Runtime Tools Help

Commands + Code + Text | Run all ▾

```
[62] ✓ 1s
import matplotlib.pyplot as plt
from PIL import Image
import os

# Reuse variables from previous execution
# train_dir is already defined as '/content/drive/MyDrive/ETS_BDA_20124122_RisnaDeSinta/train'
# train_cats and train_dogs are already populated

plt.figure(figsize=(15, 6))
sample_imgs_to_show = train_cats[:3] + train_dogs[:2] # 3 cats and 2 dogs

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

plt.suptitle('Sample Images from Training Data')
plt.show()
```

Sample Images from Training Data

cat.10007.jpg



cat.1005.jpg



cat.10003.jpg



dog.1019.jpg



dog.10174.jpg



```
(63)  ETS_BDA_20124122_RisnaDeSinta.ipynb  ⭐ ⓘ
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▶ Run all ▾

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))

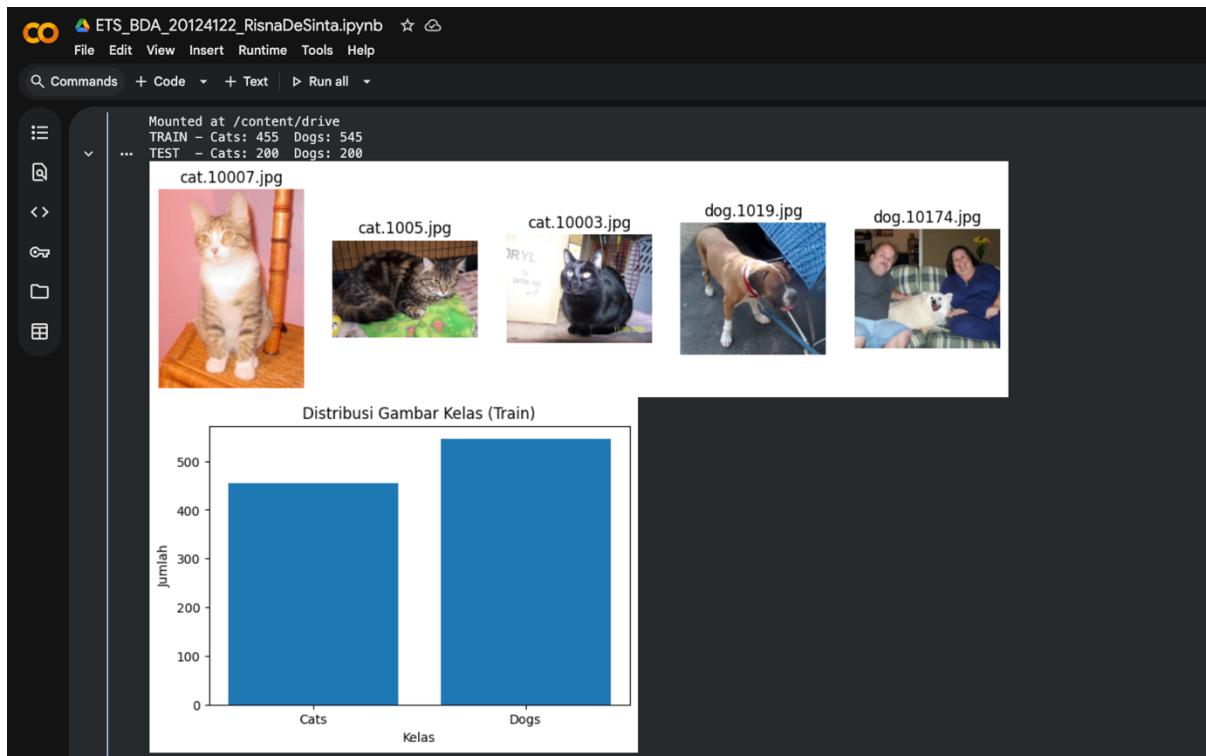
# ===== tampilkan 5 gambar sample =====
plt.figure(figsize=(12,6))
sample_imgs = train_cats[:3] + train_dogs[:2]

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()
```



## Konfigurasi Rekomendasi Simple CNN (Python - TensorFlow/Keras)

```
[67] 0s # ===== 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

```
[70] 0s 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

```
[174] In [1]
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")
])

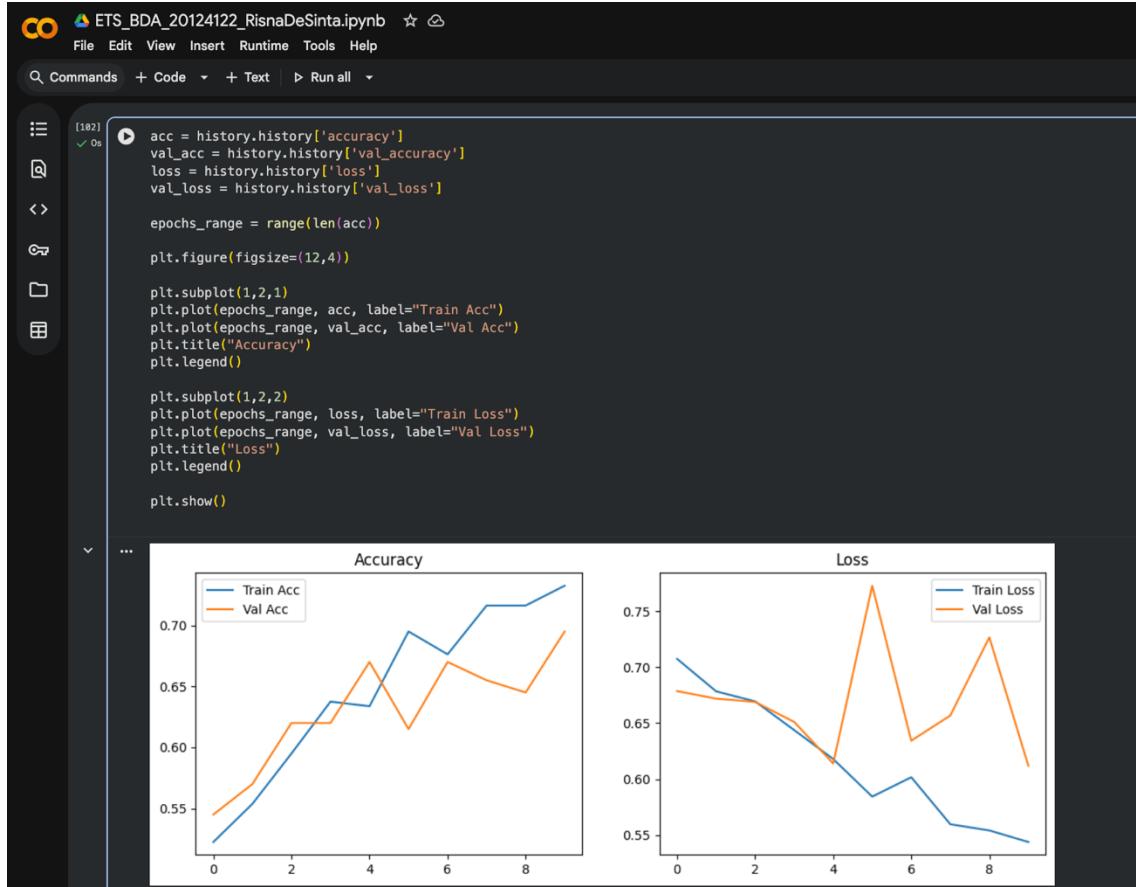
model.compile(optimizer=Adam(learning_rate=0.001),
              loss='binary_crossentropy',
              metrics=['accuracy'])

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 as the first layer.
Model: "sequential"
Layer (type)          Output Shape         Param #
conv2d (Conv2D)      (None, 32, 32, 32)   992
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)   0
conv2d_1 (Conv2D)     (None, 16, 16, 64)   18,496
max_pooling2d_1 (MaxPooling2D) (None, 8, 8, 64)   0
flatten (Flatten)    (None, 512)        0
dense (Dense)        (None, 128)        1,689,768
dropout (Dropout)   (None, 128)        0
dense_1 (Dense)      (None, 1)         129
Total params: 1,695,381 (6.20 MB)
Trainable params: 1,625,281 (6.20 MB)
Non-trainable params: 8 (0.00 B)
```

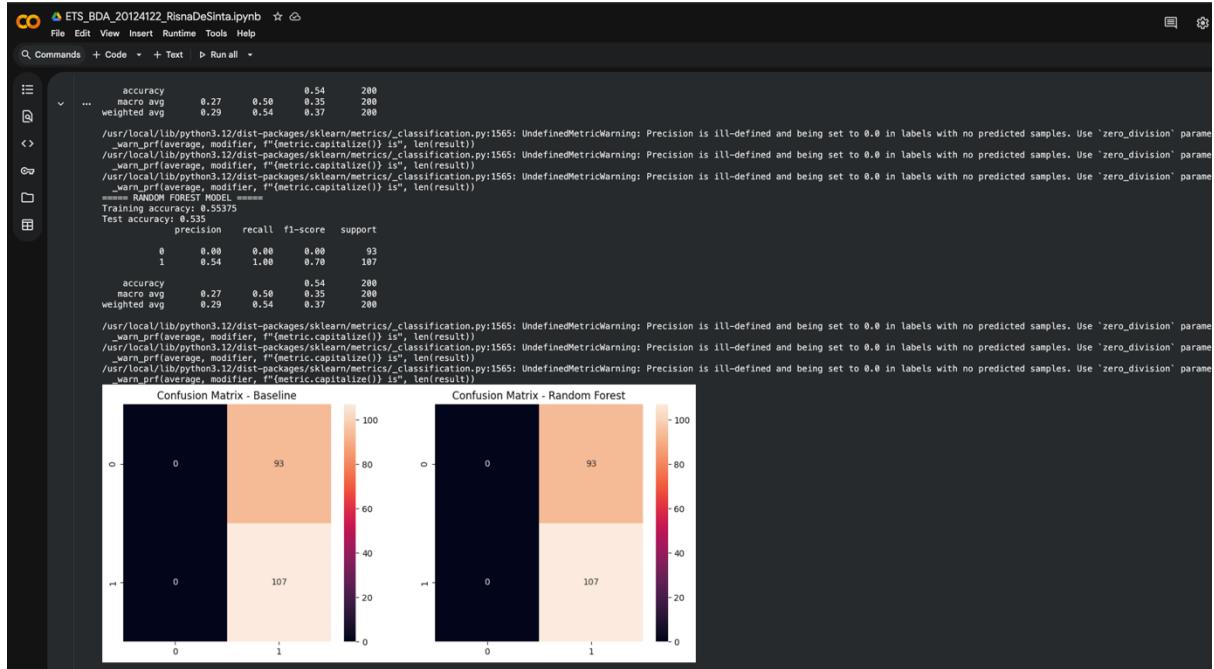
## Training

```
[174] In [2]
history = model.fit(
    train_generator,
    validation_data=val_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    205s 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: 0.6260 - 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: 0.6828 - val_accuracy: 0.6700 - val_loss: 0.6140
Epoch 6/10
25/25    9s 340ms/step - accuracy: 0.6778 - loss: 0.5987 - val_accuracy: 0.6150 - val_loss: 0.7726
Epoch 7/10
25/25    10s 376ms/step - accuracy: 0.6732 - loss: 0.6650 - 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: 0.6320
```

## Grafik Akurasi & Loss



## Modeling Phyton



The screenshot shows a Jupyter Notebook interface with two confusion matrices side-by-side. The left matrix is titled 'Confusion Matrix - Baseline' and the right is 'Confusion Matrix - Random Forest'. Both matrices have a color scale from 0 (black) to 100 (orange). The diagonal elements are labeled 93 and 107. The off-diagonal elements are labeled 0.

```
accuracy      0.54      200
macro avg     0.27     0.50     0.35     200
weighted avg  0.29     0.54     0.37     200

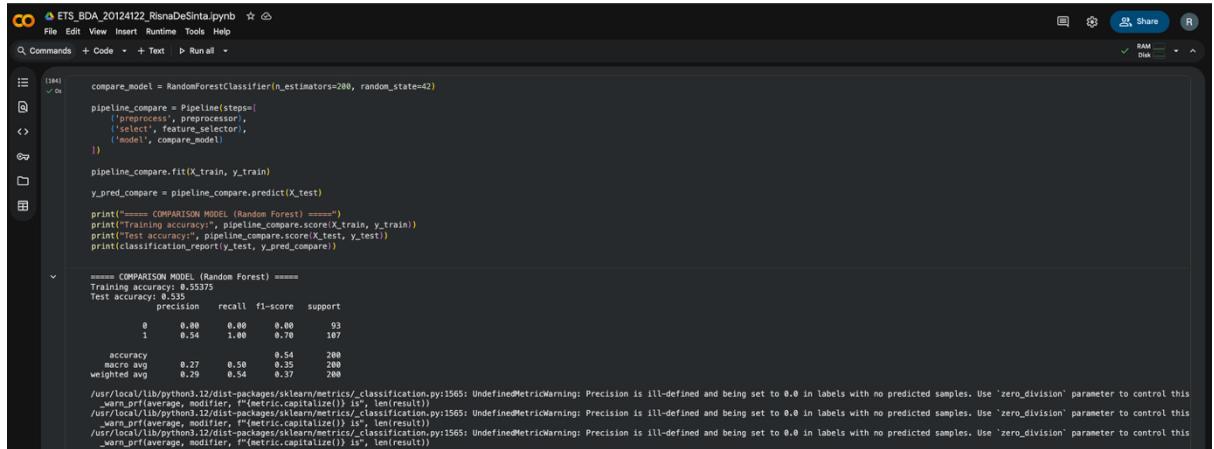
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this
Training accuracy: 0.55375
Test accuracy: 0.535

precision    recall   f1-score   support
          0       0.00     0.00     0.00      93
          1       0.54     1.00     0.70     107

accuracy      0.54      200
macro avg     0.27     0.50     0.35     200
weighted avg  0.29     0.54     0.37     200

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this
Confusion Matrix - Baseline           Confusion Matrix - Random Forest
```

## MODEL PEMBANDING — Random Forest



The screenshot shows a Jupyter Notebook interface with a cell containing code for comparing a Random Forest classifier. The code includes importing libraries, defining a pipeline, fitting it to training data, and printing classification reports for both training and test sets. The output shows a comparison between the baseline and the Random Forest model, with the Random Forest achieving a higher test accuracy of 0.535 compared to the baseline's 0.535.

```
compare_model = RandomForestClassifier(n_estimators=200, random_state=42)

pipeline_compare = Pipeline(steps=[
    ('preprocess', preprocess),
    ('select', feature_selector),
    ('model', compare_model)
])

pipeline_compare.fit(X_train, y_train)

y_pred_compare = pipeline_compare.predict(X_test)

print("===== COMPARISON MODEL (Random Forest) =====")
print("Training accuracy: ", pipeline_compare.score(X_train, y_train))
print("Test accuracy: ", pipeline_compare.score(X_test, y_test))
print(classification_report(y_test, y_pred_compare))

===== COMPARISON MODEL (Random Forest) =====
Training accuracy: 0.55375
Test accuracy: 0.535

precision    recall   f1-score   support
          0       0.00     0.00     0.00      93
          1       0.54     1.00     0.70     107

accuracy      0.54      200
macro avg     0.27     0.50     0.35     200
weighted avg  0.29     0.54     0.37     200

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this
```

## Evaluasi Model – Confusion Matrix

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
[105] 0s
  fig, ax = plt.subplots(1, 2, figsize=(12, 5))

  cm1 = confusion_matrix(y_test, y_pred_baseline)
  sns.heatmap(cm1, annot=True, fmt='d', ax=ax[0])
  ax[0].set_title("Confusion Matrix - Baseline")

  cm2 = confusion_matrix(y_test, y_pred_compare)
  sns.heatmap(cm2, annot=True, fmt='d', ax=ax[1])
  ax[1].set_title("Confusion Matrix - Random Forest")

  plt.show()
```

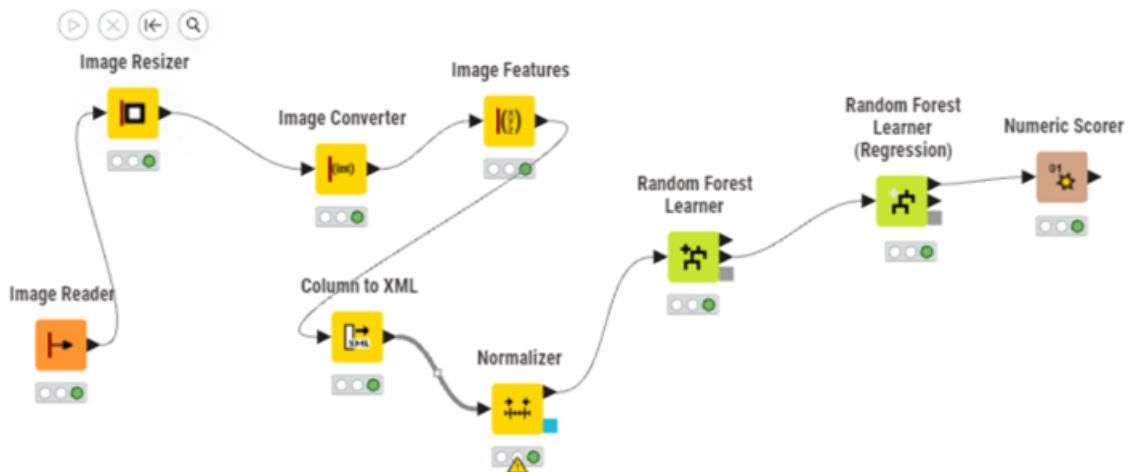
The image displays two confusion matrices side-by-side. The left matrix is titled "Confusion Matrix - Baseline" and the right is titled "Confusion Matrix - Random Forest". Both matrices have a color scale from 0 (dark blue) to 100 (yellow). The diagonal elements are labeled 93, indicating high accuracy. The off-diagonal elements are labeled 0 for the baseline and 107 for the random forest, indicating low recall.

	0	1
0	93	0
1	107	0

	0	1
0	93	0
1	107	0

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 **tidak** 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
<b>HOG</b>	menangkap pola bentuk telinga, hidung, mulut
<b>Haralick</b>	menangkap tekstur bulu dan warna intensitas
<b>Random Forest</b>	mampu generalisasi dengan baik pada fitur tabular

### 4.4 Output

#### Grafik Performance

Accuracy Train vs Test

