```python
import os

# Create a directory for Kaggle config
os.makedirs("/root/.kaggle", exist_ok=True)

# Upload `kaggle.json`
from google.colab import files
files.upload()  # Select and upload the downloaded kaggle.json file

# Move `kaggle.json` to the correct directory
!mv kaggle.json /root/.kaggle/

# Set permissions
!chmod 600 /root/.kaggle/kaggle.json

# Verify Kaggle API works
!kaggle datasets list
```

Choose Files | kaggle.json
* **kaggle.json**(application/json) - 67 bytes, last modified: 3/20/2025 - 100% done
Saving kaggle.json to kaggle.json

```
ref                                                            title                                                   size
-------------------------------------------------------------  ------------------------------------------------------  ---------
atharvasoundankar/chocolate-sales                              Chocolate Sales Data 📊 🍫                                 14
abdulmalik1518/mobiles-dataset-2025                            Mobiles Dataset (2025)                                  20314
mahmoudelhemaly/students-grading-dataset                       Student Performance & Behavior Dataset                  520428
atharvasoundankar/global-water-consumption-dataset-2000-2024   Global Water Consumption Dataset (2000-2024) 🌍 💧         1?
adilshamim8/student-depression-dataset                         Student Depression Dataset                              467026
atharvasoundankar/global-food-wastage-dataset-2018-2024        🌍 Global Food Wastage Dataset (2018-2024) 🍽             108(
adilshamim8/temperature                                        Global Environmental Trends 2000-2024                   22391
parsabahramsari/wdi-education-health-and-employment-2011-2021  WDI: Education, Health & Employment (2011-2021)         136185
bhargavchirumamilla/netflix-movies-and-tv-shows-till-2025      Netflix Movies and TV shows till 2025                   6471169
smayanj/netflix-users-database                                 Netflix Users Database                                  362559
atharvasoundankar/global-energy-consumption-2000-2024          Global Energy Consumption (2000-2024) 🔥 ⚡               25?
ricgomes/global-fashion-retail-stores-dataset                  Global Fashion Retail Sales                             234910599
aniruddhawankhede/mental-heath-analysis-among-teenagers        Mental_Heath_Analysis_Among_Teenagers                   177089
atharvasoundankar/global-music-streaming-trends-and-listener-insights  Global Music Streaming Trends & Listener Insights  97474
abdulmoiz12/amazon-stock-data-2025                             Amazon Stock Data 2025                                  160519
brsahan/genomic-data-for-cancer                                Genomic Data for Cancer                                 9134
atharvasoundankar/viral-social-media-trends-and-engagement-analysis  🚀 Viral Social Media Trends & Engagement Analysis  1072
adilshamim8/student-performance-on-an-entrance-examination     Student Performance on an Entrance Examination          4402
anandshaw2001/video-game-sales                                 Video Game Sales                                        390286
meharshanali/amazon-stocks-2025                                Amazon Stocks 2025                                      165462
```

```python
!kaggle datasets download -d jaiharish11499/wastedata
```

Dataset URL: https://www.kaggle.com/datasets/jaiharish11499/wastedata
License(s): CC0-1.0

```python
import zipfile

with zipfile.ZipFile("wastedata.zip", 'r') as zip_ref:
    zip_ref.extractall("waste_data")
```

```python
import os
print(os.listdir("/content/"))
```

['.config', 'wastedata.zip', 'waste_data', 'sample_data']

```python
train_folder = "/content/waste_data/d/Train"
test_folder = "/content/waste_data/d/Test"
```

```python
import pandas as pd
import numpy as np
import glob
import os
from datetime import datetime
from packaging import version

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.preprocessing import image_dataset_from_directory
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.callbacks import ModelCheckpoint, History
```

```python
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, Lambda, MaxPooling2D, Dense, Dropout, Flatten
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import to_categorical

from skimage.io import imread, imshow
from skimage.transform import resize
from IPython import display
import matplotlib.pyplot as plt
import seaborn as sns
from seaborn import heatmap
from sklearn.metrics import confusion_matrix
```

```python
# Data augmentation for training
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')


# No augmentation for validation/test
test_datagen = ImageDataGenerator(rescale=1./255)


# Load dataset
train_generator = train_datagen.flow_from_directory(
    train_folder,
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary')
```

```
Found 336 images belonging to 2 classes.
```

```python
test_generator = test_datagen.flow_from_directory(
    test_folder,
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',
    shuffle=False)
```

```
Found 64 images belonging to 2 classes.
```

```python
from sklearn.utils.class_weight import compute_class_weight
# Compute class weights to address imbalance
class_labels = np.array(train_generator.classes)
class_weights = compute_class_weight(class_weight='balanced', classes=np.unique(class_labels), y=class_labels)
class_weight_dict = {i: class_weights[i] for i in range(len(class_weights))}
```

```python
# Define ResNet model
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
base_model.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernel
94765736/94765736 ──────────────── 0s 0us/step
```

```python
model = keras.Sequential([
    base_model,
    keras.layers.GlobalAveragePooling2D(),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(1, activation='sigmoid')
])
```

```python
from tensorflow.keras.optimizers import Adam
#Compile the model
model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['accuracy'])
```

```python
# Train model
history = model.fit(
    train_generator,
```

```
        epochs=10,
        validation_data=test_generator,
        class_weight=class_weight_dict)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` cl
  self._warn_if_super_not_called()
Epoch 1/10
11/11 ──────────────── 99s 8s/step - accuracy: 0.3282 - loss: 0.8113 - val_accuracy: 0.6719 - val_loss: 0.6813
Epoch 2/10
11/11 ──────────────── 76s 7s/step - accuracy: 0.5233 - loss: 0.7920 - val_accuracy: 0.7812 - val_loss: 0.6284
Epoch 3/10
11/11 ──────────────── 75s 7s/step - accuracy: 0.6214 - loss: 0.7105 - val_accuracy: 0.7812 - val_loss: 0.6395
Epoch 4/10
11/11 ──────────────── 85s 8s/step - accuracy: 0.5802 - loss: 0.6663 - val_accuracy: 0.7344 - val_loss: 0.6497
Epoch 5/10
11/11 ──────────────── 89s 8s/step - accuracy: 0.5389 - loss: 0.6869 - val_accuracy: 0.7656 - val_loss: 0.6288
Epoch 6/10
11/11 ──────────────── 86s 8s/step - accuracy: 0.5986 - loss: 0.7123 - val_accuracy: 0.7656 - val_loss: 0.6144
Epoch 7/10
11/11 ──────────────── 86s 8s/step - accuracy: 0.6071 - loss: 0.7377 - val_accuracy: 0.7344 - val_loss: 0.6293
Epoch 8/10
11/11 ──────────────── 77s 7s/step - accuracy: 0.5792 - loss: 0.7442 - val_accuracy: 0.7500 - val_loss: 0.6267
Epoch 9/10
11/11 ──────────────── 86s 8s/step - accuracy: 0.5945 - loss: 0.6370 - val_accuracy: 0.7500 - val_loss: 0.6207
Epoch 10/10
11/11 ──────────────── 76s 7s/step - accuracy: 0.4807 - loss: 0.7716 - val_accuracy: 0.5938 - val_loss: 0.6740
```

```python
# Plot accuracy and loss
fig, axes = plt.subplots(1, 2, figsize=(12, 5))
axes[0].plot(history.history['accuracy'], label='Train Accuracy')
axes[0].plot(history.history['val_accuracy'], label='Validation Accuracy')
axes[0].set_title('Model Accuracy')
axes[0].legend()
axes[1].plot(history.history['loss'], label='Train Loss')
axes[1].plot(history.history['val_loss'], label='Validation Loss')
axes[1].set_title('Model Loss')
axes[1].legend()
plt.show()
```
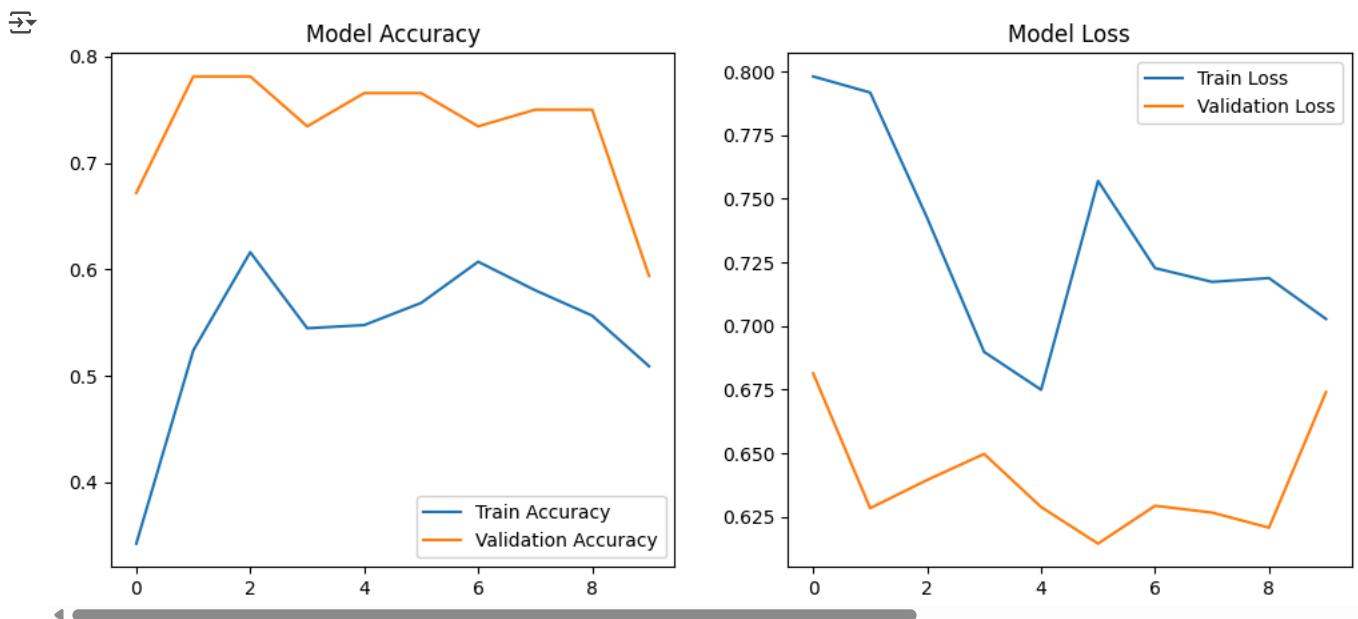


```python
# Confusion matrix
y_true = test_generator.classes
y_pred = model.predict(test_generator) > 0.5
cm = confusion_matrix(y_true, y_pred)
```

```
2/2 ──────────────── 14s 6s/step
```

```python
# Display confusion matrix with labels and percentages
fig, ax = plt.subplots(figsize=(6, 6))
cm_percent = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis] * 100
sns.heatmap(cm_percent, annot=True, fmt='.2f', cmap='Blues', xticklabels=['Pred Non-Recyclable', 'Pred Recyclable'],
            yticklabels=['Actual Non-Recyclable', 'Actual Recyclable'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
```

Text(0.5, 1.0, 'Confusion Matrix')
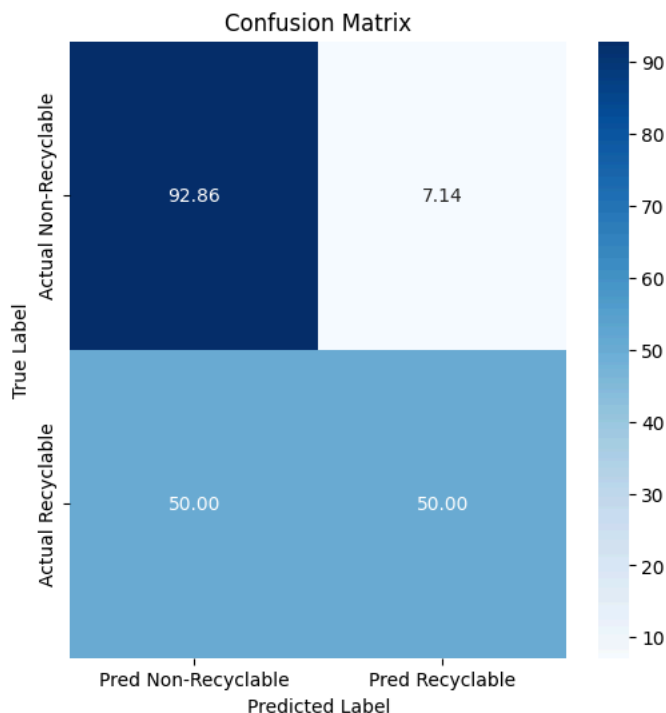

```python
from sklearn.metrics import classification_report
# Classification report
print("Classification Report:")
print(classification_report(y_true, y_pred, target_names=['Non-Recyclable', 'Recyclable']))
```

Classification Report:

	precision	recall	f1-score	support
Non-Recyclable	0.34	0.93	0.50	14
Recyclable	0.96	0.50	0.66	50
accuracy			0.59	64
macro avg	0.65	0.71	0.58	64
weighted avg	0.83	0.59	0.62	64

```python
# Convert accuracy and loss to percentage
train_acc = [x * 100 for x in history.history['accuracy']]
val_acc = [x * 100 for x in history.history['val_accuracy']]
train_loss = [x * 100 for x in history.history['loss']]
val_loss = [x * 100 for x in history.history['val_loss']]
# Print accuracy and loss values
print("Final Training Accuracy: {:.2f}%".format(train_acc[-1]))
print("Final Validation Accuracy: {:.2f}%".format(val_acc[-1]))
print("Final Training Loss: {:.2f}%".format(train_loss[-1]))
print("Final Validation Loss: {:.2f}%".format(val_loss[-1]))
```

Final Training Accuracy: 50.89%
Final Validation Accuracy: 59.38%
Final Training Loss: 70.28%
Final Validation Loss: 67.40%