Full Project Code - Smart Sorting (Colab)

1. Import Libraries

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np
import os
```

2. Data Preparation

```
train_path = '/content/dataset/train'
val_path = '/content/dataset/val'
test_path = '/content/dataset/test'

train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(train_path, target_size=(224, 224), batch_size=32, class_mode='binary')
val_generator = val_datagen.flow_from_directory(val_path, target_size=(224, 224), batch_size=32, class_mode='binary')
test_generator = test_datagen.flow_from_directory(test_path, target_size=(224, 224), batch_size=32, class_mode='binary')
```

3. Build Model with Transfer Learning

```
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
predictions = Dense(1, activation='sigmoid')(x)
model = Model(inputs=base_model.input, outputs=predictions)

for layer in base_model.layers:
    layer.trainable = False

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

4. Train the Model

```
history = model.fit(train_generator, validation_data=val_generator, epochs=10)
```

5. Evaluate and Save

```
loss, acc = model.evaluate(test_generator)
```

```
print(f"Test Accuracy: {acc}")

model.save('fruit_quality_model.h5')
```

6. Flask Web App (app.py)

```
from flask import Flask, render_template, request
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np
import os

app = Flask(__name__)
model = load_model('fruit_quality_model.h5')

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        file = request.files['file']
        filename = file.filename
        file_path = os.path.join('static', filename)
        file.save(file_path)

        image = load_img(file_path, target_size=(224, 224))
        image = img_to_array(image)
        image = np.expand_dims(image, axis=0) / 255.0

        prediction = model.predict(image)
        result = "Fresh" if prediction[0][0] < 0.5 else "Rotten"

        return render_template('index.html', prediction=result, image_path=file_path)

    return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True)
```