Split_dataset.py

```
import os
import shutil
import random
SOURCE_DIR = 'dataset'
TARGET_DIR = 'final_dataset'


SPLIT_RATIOS = (0.7, 0.15, 0.15)  # 70% train, 15% validation, 15% test


categories = ['Fresh', 'Rotten']

for category in categories:
    category_path = os.path.join(SOURCE_DIR, category)
    files = os.listdir(category_path)
    random.shuffle(files)

    total = len(files)
    train_end = int(SPLIT_RATIOS[0] * total)
    val_end = train_end + int(SPLIT_RATIOS[1] * total)

    train_files = files[:train_end]
    val_files = files[train_end:val_end]
    test_files = files[val_end:]

    for split_name, split_files in zip(['train', 'validation', 'test'], [train_files, val_files, test_files]):
        dest_dir = os.path.join(TARGET_DIR, split_name, category)
        os.makedirs(dest_dir, exist_ok=True)

        for file in split_files:
            src_file = os.path.join(category_path, file)
            dst_file = os.path.join(dest_dir, file)
            shutil.copy2(src_file, dst_file)

print(" ✅ Dataset split completed!")
```

train_model.py

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint
import os
image_size = 224
```

```
batch_size = 32
epochs = 10


train_dir = 'final_dataset/train'
val_dir = 'final_dataset/validation'
test_dir = 'final_dataset/test'


datagen = ImageDataGenerator(rescale=1./255)

train_generator = datagen.flow_from_directory(
    train_dir,
    target_size=(image_size, image_size),
    batch_size=batch_size,
    class_mode='categorical'
)

val_generator = datagen.flow_from_directory(
    val_dir,
    target_size=(image_size, image_size),
    batch_size=batch_size,
    class_mode='categorical'
)

test_generator = datagen.flow_from_directory(
    test_dir,
    target_size=(image_size, image_size),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)


base_model = MobileNetV2(weights='imagenet', include_top=False,
input_shape=(image_size, image_size, 3))
base_model.trainable = False  # Freeze base layers


x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(64, activation='relu')(x)
output = Dense(2, activation='softmax')(x)  # 2 classes: Fresh, Rotten

model = Model(inputs=base_model.input, outputs=output)
```

```
model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy',
metrics=['accuracy'])


checkpoint = ModelCheckpoint('healthy_vs_rotten.h5', monitor='val_accuracy',
save_best_only=True, mode='max')
model.fit(train_generator, validation_data=val_generator, epochs=epochs,
callbacks=[checkpoint])


loss, accuracy = model.evaluate(test_generator)
print(f"✅ Test Accuracy: {accuracy * 100:.2f}%")
```

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("healthy_vs_rotten.h5")


class_names = ['Fresh', 'Rotten']

UPLOAD_FOLDER = 'static/uploads'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    if 'image' not in request.files:
        return 'No image file found'

    file = request.files['image']
    if file.filename == '':
        return 'No file selected'

    filepath = os.path.join(UPLOAD_FOLDER, file.filename)
```

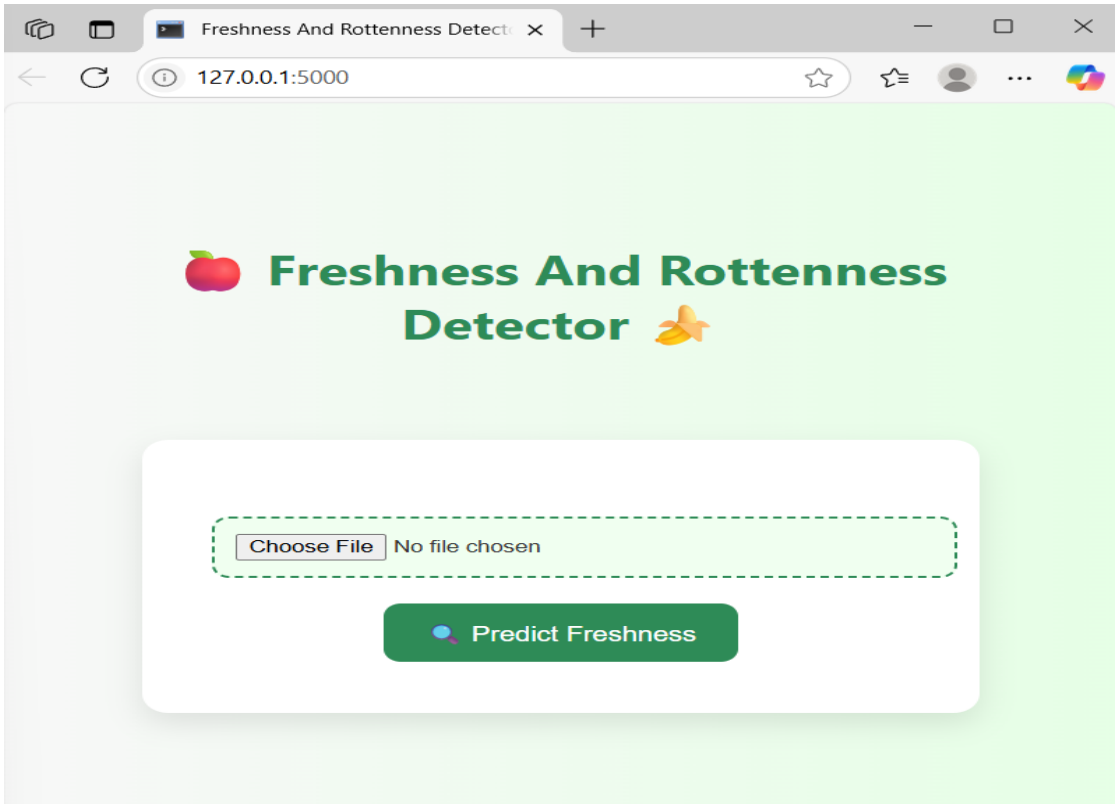
```
    file.save(filepath)

    img = load_img(filepath, target_size=(224, 224))
    img_array = img_to_array(img) / 255.0
    img_array = np.expand_dims(img_array, axis=0)

    prediction = model.predict(img_array)[0]
    class_idx = np.argmax(prediction)
    confidence = prediction[class_idx]

    result = f"{class_names[class_idx]} ({confidence * 100:.2f}%)"
    return render_template('result.html', image_path=filepath, prediction=result)

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

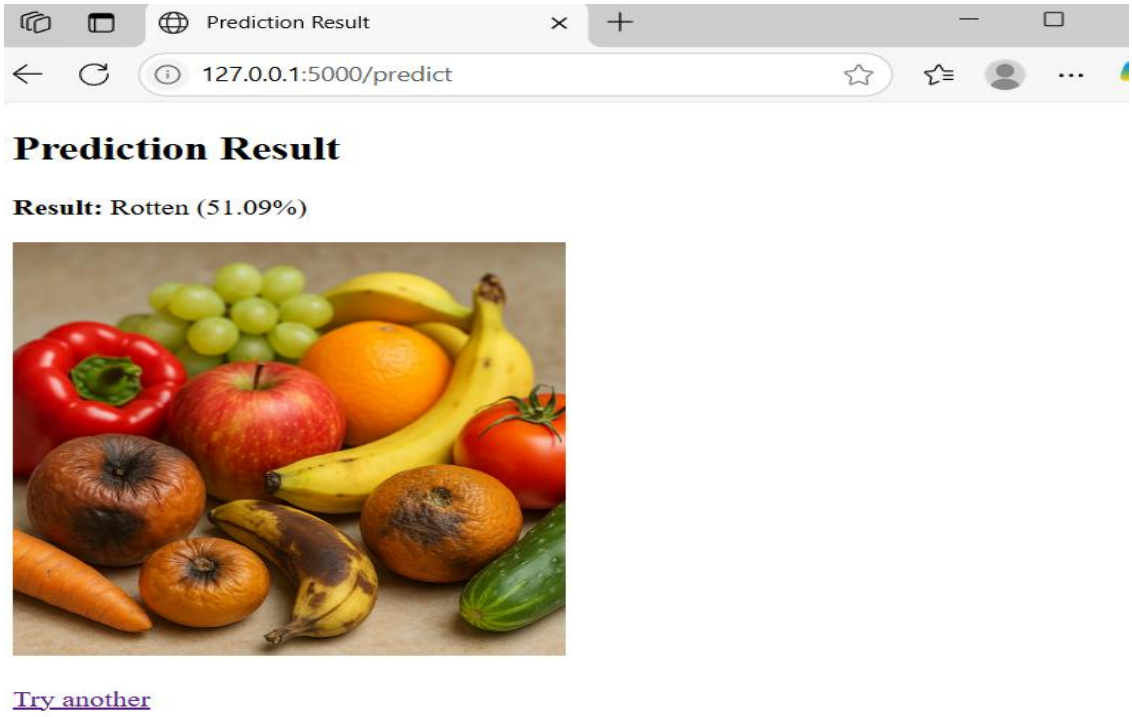
index.html

```
<!DOCTYPE html>
<html>
<head>
    <title> Freshness And Rottenness Detector</title>
    <style>
        body {
            background: linear-gradient(to right, #f8f8f8, #e6ffe6);
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            text-align: center;
            padding: 50px;
            color: #333;
        }

        h2 {
            margin-top:50px;
            font-size: 32px;
            margin-bottom: 70px;
            color: #2e8b57;
        }

        form {
            background-color: #ffffff;
            border-radius: 15px;
            padding: 40px;
            box-shadow: 0 8px 20px rgba(0, 0, 0, 0.1);
            display: inline-block;
            max-width: 400px;
            width: 100%;
        }

        input[type="file"] {
            padding: 12px;
```

```
            margin: 20px 0;
            border: 2px dashed #2e8b57;
            border-radius: 10px;
            background-color: #f0fff0;
            cursor: pointer;
            width: 100%;
            transition: 0.3s ease-in-out;
        }

        input[type="file"]:hover {
            background-color: #e0ffe0;
        }

        button {
            padding: 12px 24px;
            background-color: #2e8b57;
            color: white;
            border: none;
            border-radius: 10px;
            font-size: 16px;
            cursor: pointer;
            transition: background-color 0.3s, transform 0.2s;
        }

        button:hover {
            background-color: #246b45;
            transform: scale(1.05);
        }

        @media (max-width: 600px) {
            body {
                padding: 20px;
            }

            form {
                padding: 25px;
            }
        }
    </style>
</head>
<body>
    <h2>🍎 Freshness And Rottenness Detector 🍌</h2>
    <form action="/predict" method="post" enctype="multipart/form-data">
        <input type="file" name="image" accept="image/*" required>
        <br>
        <button type="submit">🔍 Predict Freshness</button>
    </form>
</body>
</html>
```

Output


Freshness And Rottenness Detect
127.0.0.1:5000
Freshness And Rottenness Detector
Choose File
No file chosen
Predict Freshness


Prediction Result
127.0.0.1:5000/predict
Prediction Result
Result: Rotten (51.09%)
Try another