In [174…

```
import tensorflow as tf
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
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.losses import BinaryCrossentropy
import matplotlib.pyplot as plt
import os
from sklearn.metrics import classification_report, f1_score, accuracy_score, precision_score, recall_score
import numpy as np
```

In [175…

```
fresh = len(os.listdir("/Users/alyaaljarallah/Desktop/Fruit_project/fruits_dataset/fresh"))
rotten = len(os.listdir("/Users/alyaaljarallah/Desktop/Fruit_project/fruits_dataset/rotten"))

print(f"Fresh: {fresh}, Rotten: {rotten}")
```

```
Fresh: 105, Rotten: 121
```

In [176…

```
dataset_path = "/Users/alyaaljarallah/Desktop/Fruit_project/fruits_dataset"

train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2, zoom_range=0.3,
    horizontal_flip=True,
    brightness_range=[0.6, 1.4]
)

val_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2
)

train_generator = train_datagen.flow_from_directory( dataset_path,
target_size=(150, 150), batch_size=32, class_mode='binary', subset='training', shuffle=True)

val_generator = val_datagen.flow_from_directory( dataset_path,
target_size=(150, 150), batch_size=32, class_mode='binary', subset='validation', shuffle=False )
```

```
Found 181 images belonging to 2 classes.
Found 44 images belonging to 2 classes.
```

In [177…

```
model = Sequential([
       Conv2D(32, (3, 3), activation='relu',
       input_shape=(150, 150, 3)), MaxPooling2D(2, 2),

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

       Conv2D(128, (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=1e-4),
              loss=BinaryCrossentropy(label_smoothing=0.1), metrics=['accuracy'] )
model.summary()
```

```
/Users/alyaaljarallah/opt/anaconda3/lib/python3.9/site-packages/keras/src/layers/convolutional/base_conv.py:107: User
Warning: 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 in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential_23"

Layer (type)	Output Shape	Param #
conv2d_56 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_46 (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_57 (Conv2D)	(None, 72, 72, 64)	18,496
max_pooling2d_47 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_58 (Conv2D)	(None, 34, 34, 128)	73,856
max_pooling2d_48 (MaxPooling2D)	(None, 17, 17, 128)	0
flatten_23 (Flatten)	(None, 36992)	0
dense_48 (Dense)	(None, 128)	4,735,104
dropout_19 (Dropout)	(None, 128)	0
dense_49 (Dense)	(None, 1)	129

Total params: 4,828,481 (18.42 MB)
Trainable params: 4,828,481 (18.42 MB)
Non-trainable params: 0 (0.00 B)

In [178…

```
early_stop = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)

checkpoint = ModelCheckpoint( 'best_model.keras', monitor='val_accuracy', save_best_only=True, mode='max')

history = model.fit( train_generator, validation_data=val_generator, epochs=20, callbacks=[early_stop, checkpoint] )
```

```
/Users/alyaaljarallah/opt/anaconda3/lib/python3.9/site-packages/keras/src/trainers/data_adapters/py_dataset_adapter.p
y:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` ca
n include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be
ignored.
  self._warn_if_super_not_called()
```

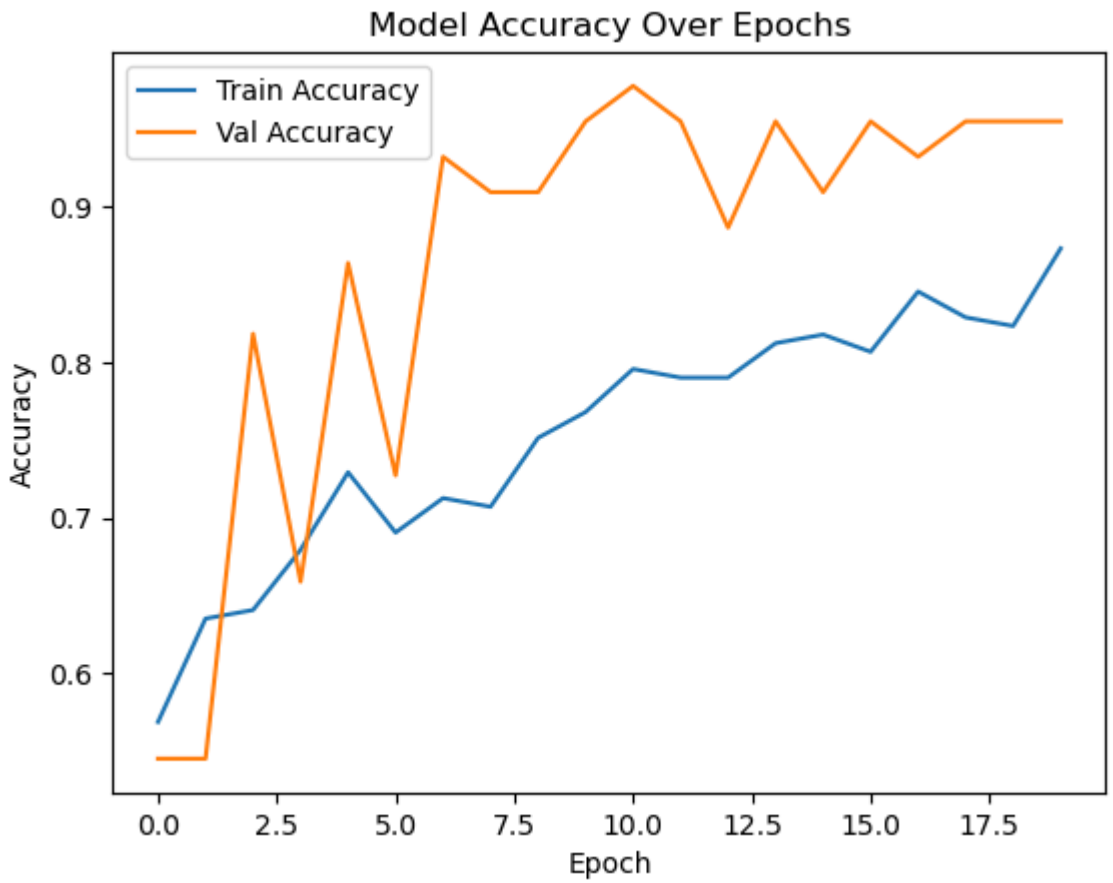
```
Epoch 1/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 10s 982ms/step - accuracy: 0.5701 - loss: 0.6855 - val_accuracy: 0.5455 - val_loss: 0.6593
Epoch 2/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 6s 860ms/step - accuracy: 0.6048 - loss: 0.6414 - val_accuracy: 0.5455 - val_loss: 0.6544
Epoch 3/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 12s 1s/step - accuracy: 0.6467 - loss: 0.6385 - val_accuracy: 0.8182 - val_loss: 0.5700
Epoch 4/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 7s 944ms/step - accuracy: 0.6496 - loss: 0.6403 - val_accuracy: 0.6591 - val_loss: 0.6176
Epoch 5/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 6s 1s/step - accuracy: 0.7192 - loss: 0.6049 - val_accuracy: 0.8636 - val_loss: 0.5215
Epoch 6/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 6s 958ms/step - accuracy: 0.6688 - loss: 0.6110 - val_accuracy: 0.7273 - val_loss: 0.5704
Epoch 7/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 7s 1s/step - accuracy: 0.6960 - loss: 0.5845 - val_accuracy: 0.9318 - val_loss: 0.5035
Epoch 8/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 7s 1s/step - accuracy: 0.7139 - loss: 0.5537 - val_accuracy: 0.9091 - val_loss: 0.4950
Epoch 9/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 9s 881ms/step - accuracy: 0.7741 - loss: 0.5568 - val_accuracy: 0.9091 - val_loss: 0.4440
Epoch 10/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 6s 1s/step - accuracy: 0.7679 - loss: 0.5398 - val_accuracy: 0.9545 - val_loss: 0.4471
Epoch 11/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 7s 1s/step - accuracy: 0.7532 - loss: 0.5659 - val_accuracy: 0.9773 - val_loss: 0.4105
Epoch 12/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 5s 808ms/step - accuracy: 0.7827 - loss: 0.5110 - val_accuracy: 0.9545 - val_loss: 0.3975
Epoch 13/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 6s 912ms/step - accuracy: 0.8088 - loss: 0.5162 - val_accuracy: 0.8864 - val_loss: 0.4228
Epoch 14/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 6s 970ms/step - accuracy: 0.8012 - loss: 0.5212 - val_accuracy: 0.9545 - val_loss: 0.3943
Epoch 15/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 7s 962ms/step - accuracy: 0.8032 - loss: 0.5300 - val_accuracy: 0.9091 - val_loss: 0.3823
Epoch 16/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 7s 1s/step - accuracy: 0.8314 - loss: 0.4487 - val_accuracy: 0.9545 - val_loss: 0.3741
Epoch 17/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 9s 854ms/step - accuracy: 0.8347 - loss: 0.4897 - val_accuracy: 0.9318 - val_loss: 0.3561
Epoch 18/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 5s 707ms/step - accuracy: 0.8288 - loss: 0.4550 - val_accuracy: 0.9545 - val_loss: 0.3242
Epoch 19/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 4s 689ms/step - accuracy: 0.8223 - loss: 0.4702 - val_accuracy: 0.9545 - val_loss: 0.3164
Epoch 20/20
6/6 ━━━━━━━━━━━━━━━━━━━━ 4s 738ms/step - accuracy: 0.8680 - loss: 0.4463 - val_accuracy: 0.9545 - val_loss: 0.3315
```

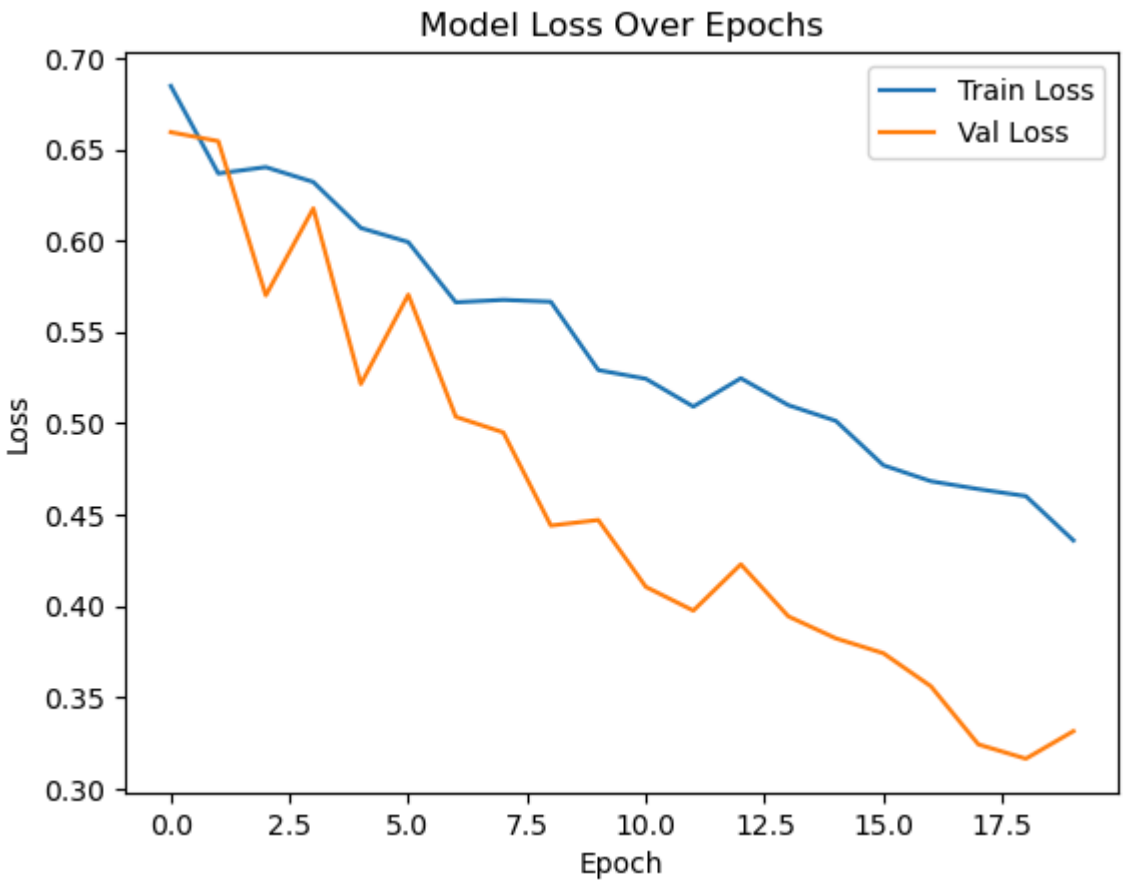
In [181…

```
val_generator.reset()
y_probs = model.predict(val_generator)
y_preds = (y_probs > 0.5).astype(int).flatten()
```

```python
y_true = val_generator.classes[:len(y_preds)]

accuracy = accuracy_score(y_true, y_preds)
precision = precision_score(y_true, y_preds)
recall = recall_score(y_true, y_preds)
f1 = f1_score(y_true, y_preds)

print(f"Accuracy:  {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall:    {recall:.4f}")
print(f"F1 Score:  {f1:.4f}")

# Accuracy
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.title('Model Accuracy Over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# Loss
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Model Loss Over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 152ms/step
Accuracy:  0.9545
Precision: 1.0000
Recall:    0.9167
F1 Score:  0.9565
```


Model Accuracy Over Epochs
Train Accuracy
Val Accuracy
Accuracy
0.9
0.8
0.7
0.6
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch



In [184…

```
plt.figure(figsize=(10, 6))
for i in range(6):
    plt.subplot(2, 3, i + 1)
    plt.imshow(x_val[i])
    plt.title("Fresh" if y_val[i] == 0 else "Rotten")
    plt.axis('off')

plt.tight_layout()
plt.show()
```

Fresh

Fresh

Fresh

Fresh

Fresh

Fresh