FilipeAMarques / ML-in-Biotech-Project

Code Issues Pull requests Actions Projects Wiki Security Insigh

ML-in-Biotech-Project / Model3(Severity)v2.ipynb

FilipeAMarques Add files via upload 2d8a576 · 1 minute ago

1 lines (1 loc) · 280 KB

Preview Code Blame Raw

Using all severity classes

In [ ]:

```
import pandas as pd
import requests

url = "https://raw.githubusercontent.com/FilipeAMarques/ML-in-Biotech-Project/m
response = requests.get(url)

# Save the content to a local file
with open("Merged.csv", "w") as f:
    f.write(response.text)

# Load the merged dataset into a pandas DataFrame
merged = pd.read_csv("Merged.csv")

# Display the first few rows of the merged DataFrame
display(merged.head())

# Define features (X) and labels (y)
# 'Severity' is the target variable
y = merged['Severity']

# Drop 'ID', 'Diagnosis', 'Recurrence', and 'Severity' to get the features (X)
X = merged.drop(['ID', 'Diagnosis', 'Recurrence', 'Severity'], axis=1)

# ============================
# Data Preprocessing
# ============================
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Split data into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.4, random
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.7,

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)


# ============================
# Improved Model 3: Severity prediction
# ============================

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from sklearn.utils.class_weight import compute_class_weight
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# --- Prepare labels for Severity ---
# Ensure Severity is integer encoded
y_train_sev = y_train.values.astype(int)
y_val_sev   = y_val.values.astype(int)
y_test_sev  = y_test.values.astype(int)

# --- Compute class weights (to handle imbalance) ---
```

```python
# --- Compute class weights (to handle imbalance)
class_weights = compute_class_weight(
    class_weight='balanced',
    classes=np.unique(y_train_sev),
    y=y_train_sev
)
class_weights = dict(enumerate(class_weights))
print("Class Weights:", class_weights)

# --- Build improved Model 3 ---
model_3 = Sequential([
    Dense(128, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    BatchNormalization(),
    Dropout(0.3),

    Dense(64, activation='relu'),
    BatchNormalization(),
    Dropout(0.3),

    Dense(16, activation='relu'),
    BatchNormalization(),
    Dropout(0.2),

    Dense(4, activation='softmax')  # 4 severity classes
])

model_3.compile(
    optimizer=Adam(learning_rate=1e-3),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

model_3.summary()

# --- Callbacks ---
early_stop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weight
reduce_lr  = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, min_

# --- Train ---
history_3 = model_3.fit(
    X_train_scaled, y_train_sev,
    validation_data=(X_val_scaled, y_val_sev),
    epochs=500,
    batch_size=32,
    class_weight=class_weights,
    callbacks=[early_stop, reduce_lr],
    verbose=1
)

# --- Evaluate ---
loss_3, acc_3 = model_3.evaluate(X_test_scaled, y_test_sev, verbose=0)
print(f"Model 3 Test Loss: {loss_3:.4f}")
print(f"Model 3 Test Accuracy: {acc_3:.4f}")

# --- Predictions ---
y_pred_proba_3 = model_3.predict(X_test_scaled)
y_pred_3 = np.argmax(y_pred_proba_3, axis=1)

# --- Confusion Matrix ---
conf_matrix_3 = confusion_matrix(y_test_sev, y_pred_3)
plt.figure(figsize=(8,6))
sns.heatmap(conf_matrix_3, annot=True, fmt='d', cmap='Blues',
            xticklabels=[f'Pred {i}' for i in range(4)],
            yticklabels=[f'Actual {i}' for i in range(4)])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Model 3 (Severity)")
```

```
plt.show()

# --- Classification Report ---
report = classification_report(
    y_test_sev, y_pred_3,
    target_names=[f"Class {i}" for i in range(4)],
    digits=4
)
print(report)

# --- Performance Metrics DataFrame ---
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_s

accuracy_3 = accuracy_score(y_test_sev, y_pred_3)
precision_3 = precision_score(y_test_sev, y_pred_3, average='weighted')
recall_3 = recall_score(y_test_sev, y_pred_3, average='weighted')
f1_3 = f1_score(y_test_sev, y_pred_3, average='weighted')

performance_metrics_3 = pd.DataFrame({
    'Metric': ['Accuracy', 'Precision (weighted)', 'Recall (weighted)', 'F1-sco
    'Score': [accuracy_3, precision_3, recall_3, f1_3]
})
display(performance_metrics_3)
```

	ID	radius1	texture1	perimeter1	area1	smoothness1	compactness1	concavity1
0	842302	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001
1	842517	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869
2	84300903	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974
3	84348301	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414
4	84358402	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980

5 rows × 34 columns

```
Class Weights: {0: np.float64(0.3983644859813084), 1: np.float64(0.936813186813186
8), 2: np.float64(3.044642857142857), 3: np.float64(10.65625)}
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWar
ning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Seq
uential models, prefer using an `Input(shape)` object as the first layer in the mo
del instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential_21"

Layer (type)	Output Shape	Param #
dense_87 (Dense)	(None, 128)	3,968
batch_normalization_66 (BatchNormalization)	(None, 128)	512
dropout_66 (Dropout)	(None, 128)	0
dense_88 (Dense)	(None, 64)	8,256
batch_normalization_67 (BatchNormalization)	(None, 64)	256
dropout_67 (Dropout)	(None, 64)	0
dense_89 (Dense)	(None, 16)	1,040
batch_normalization_68 (BatchNormalization)	(None, 16)	64

dropout_68 (Dropout)	(None, 16)	0
dense_90 (Dense)	(None, 4)	68

Total params: 14,164 (55.33 KB)

Trainable params: 13,748 (53.70 KB)

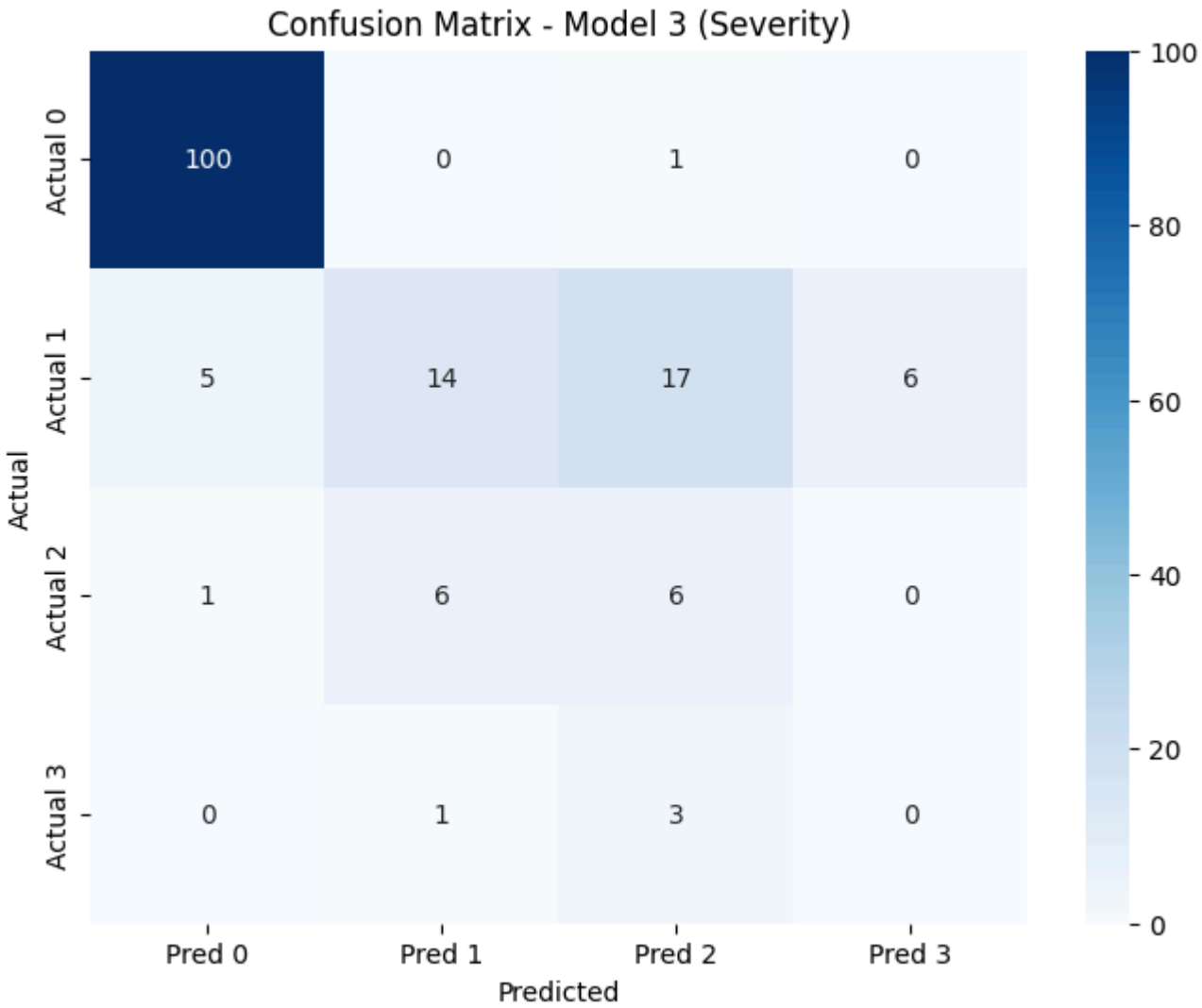
Non-trainable params: 416 (1.62 KB)

```
Epoch 1/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 3s 35ms/step - accuracy: 0.2360 - loss: 2.4197 - val_ac
curacy: 0.5147 - val_loss: 1.2622 - learning_rate: 0.0010
Epoch 2/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.3489 - loss: 1.6470 - val_ac
curacy: 0.4853 - val_loss: 1.1991 - learning_rate: 0.0010
Epoch 3/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.3660 - loss: 1.4183 - val_ac
curacy: 0.6029 - val_loss: 1.1530 - learning_rate: 0.0010
Epoch 4/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.4811 - loss: 1.3257 - val_ac
curacy: 0.6471 - val_loss: 1.1051 - learning_rate: 0.0010
Epoch 5/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.4864 - loss: 1.4933 - val_ac
curacy: 0.6029 - val_loss: 1.0715 - learning_rate: 0.0010
Epoch 6/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.5188 - loss: 1.1005 - val_ac
curacy: 0.5735 - val_loss: 1.0447 - learning_rate: 0.0010
Epoch 7/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.5446 - loss: 1.0280 - val_ac
curacy: 0.5882 - val_loss: 1.0160 - learning_rate: 0.0010
Epoch 8/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.5008 - loss: 1.2593 - val_ac
curacy: 0.5294 - val_loss: 1.0018 - learning_rate: 0.0010
Epoch 9/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.5311 - loss: 0.9617 - val_acc
uracy: 0.5441 - val_loss: 0.9660 - learning_rate: 0.0010
Epoch 10/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.6208 - loss: 1.0270 - val_acc
uracy: 0.5294 - val_loss: 0.9331 - learning_rate: 0.0010
Epoch 11/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.5853 - loss: 1.2221 - val_ac
curacy: 0.5441 - val_loss: 0.9094 - learning_rate: 0.0010
Epoch 12/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.5836 - loss: 0.9573 - val_ac
curacy: 0.5588 - val_loss: 0.8834 - learning_rate: 0.0010
Epoch 13/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6238 - loss: 0.8329 - val_ac
curacy: 0.6176 - val_loss: 0.8504 - learning_rate: 0.0010
Epoch 14/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.6078 - loss: 1.0507 - val_acc
uracy: 0.6471 - val_loss: 0.8242 - learning_rate: 0.0010
Epoch 15/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6211 - loss: 0.8130 - val_ac
curacy: 0.6471 - val_loss: 0.7995 - learning_rate: 0.0010
Epoch 16/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6537 - loss: 0.9084 - val_ac
curacy: 0.6471 - val_loss: 0.7762 - learning_rate: 0.0010
Epoch 17/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.6299 - loss: 0.9040 - val_ac
curacy: 0.6471 - val_loss: 0.7464 - learning_rate: 0.0010
Epoch 18/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6506 - loss: 1.0603 - val_ac
curacy: 0.6618 - val_loss: 0.7274 - learning_rate: 0.0010
Epoch 19/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6837 - loss: 0.8401 - val_ac
curacy: 0.6765 - val_loss: 0.7046 - learning_rate: 0.0010
Epoch 20/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6917 - loss: 1.0156 - val_ac
curacy: 0.6765 - val_loss: 0.6874 - learning_rate: 0.0010
```

```
Epoch 21/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7178 - loss: 0.8952 - val_ac
curacy: 0.6765 - val_loss: 0.6775 - learning_rate: 0.0010
Epoch 22/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7442 - loss: 0.8552 - val_ac
curacy: 0.6765 - val_loss: 0.6706 - learning_rate: 0.0010
Epoch 23/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7467 - loss: 0.7061 - val_ac
curacy: 0.6765 - val_loss: 0.6525 - learning_rate: 0.0010
Epoch 24/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7334 - loss: 0.7734 - val_ac
curacy: 0.6765 - val_loss: 0.6506 - learning_rate: 0.0010
Epoch 25/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.6988 - loss: 0.8236 - val_ac
curacy: 0.6765 - val_loss: 0.6557 - learning_rate: 0.0010
Epoch 26/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6899 - loss: 0.9138 - val_ac
curacy: 0.6765 - val_loss: 0.6444 - learning_rate: 0.0010
Epoch 27/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7166 - loss: 0.7995 - val_ac
curacy: 0.6765 - val_loss: 0.6392 - learning_rate: 0.0010
Epoch 28/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7539 - loss: 0.7297 - val_ac
curacy: 0.6765 - val_loss: 0.6255 - learning_rate: 0.0010
Epoch 29/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7423 - loss: 0.7816 - val_ac
curacy: 0.6765 - val_loss: 0.6158 - learning_rate: 0.0010
Epoch 30/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7568 - loss: 0.7736 - val_acc
uracy: 0.6765 - val_loss: 0.6257 - learning_rate: 0.0010
Epoch 31/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7518 - loss: 0.6785 - val_ac
curacy: 0.6912 - val_loss: 0.6124 - learning_rate: 0.0010
Epoch 32/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7314 - loss: 0.7549 - val_ac
curacy: 0.7059 - val_loss: 0.5991 - learning_rate: 0.0010
Epoch 33/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7283 - loss: 0.7284 - val_ac
curacy: 0.7059 - val_loss: 0.5847 - learning_rate: 0.0010
Epoch 34/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7284 - loss: 0.6702 - val_ac
curacy: 0.6912 - val_loss: 0.5740 - learning_rate: 0.0010
Epoch 35/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7352 - loss: 0.7483 - val_acc
uracy: 0.6912 - val_loss: 0.5588 - learning_rate: 0.0010
Epoch 36/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7465 - loss: 0.8904 - val_ac
curacy: 0.7059 - val_loss: 0.5686 - learning_rate: 0.0010
Epoch 37/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7287 - loss: 0.7122 - val_acc
uracy: 0.7059 - val_loss: 0.5748 - learning_rate: 0.0010
Epoch 38/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7359 - loss: 0.7822 - val_acc
uracy: 0.7059 - val_loss: 0.5869 - learning_rate: 0.0010
Epoch 39/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7316 - loss: 0.6677 - val_ac
curacy: 0.7059 - val_loss: 0.5743 - learning_rate: 0.0010
Epoch 40/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7986 - loss: 0.6926 - val_ac
curacy: 0.7206 - val_loss: 0.5760 - learning_rate: 0.0010
Epoch 41/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7664 - loss: 0.7078 - val_ac
curacy: 0.7206 - val_loss: 0.5778 - learning_rate: 5.0000e-04
Epoch 42/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7348 - loss: 0.6139 - val_acc
uracy: 0.7206 - val_loss: 0.5666 - learning_rate: 5.0000e-04
Epoch 43/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7404 - loss: 0.7469 - val_ac
curacy: 0.7206 - val_loss: 0.5574 - learning_rate: 5.0000e-04
```

```
Epoch 44/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7600 - loss: 0.6810 - val_ac
curacy: 0.7206 - val_loss: 0.5489 - learning_rate: 5.0000e-04
Epoch 45/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7923 - loss: 0.5742 - val_ac
curacy: 0.7353 - val_loss: 0.5441 - learning_rate: 5.0000e-04
Epoch 46/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7674 - loss: 0.6231 - val_acc
uracy: 0.7353 - val_loss: 0.5504 - learning_rate: 5.0000e-04
Epoch 47/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7696 - loss: 0.7871 - val_ac
curacy: 0.7353 - val_loss: 0.5563 - learning_rate: 5.0000e-04
Epoch 48/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7914 - loss: 0.6406 - val_ac
curacy: 0.7353 - val_loss: 0.5506 - learning_rate: 5.0000e-04
Epoch 49/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8312 - loss: 0.6002 - val_ac
curacy: 0.7353 - val_loss: 0.5408 - learning_rate: 5.0000e-04
Epoch 50/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7938 - loss: 0.6685 - val_ac
curacy: 0.7353 - val_loss: 0.5359 - learning_rate: 5.0000e-04
Epoch 51/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7652 - loss: 0.6203 - val_acc
uracy: 0.7353 - val_loss: 0.5368 - learning_rate: 5.0000e-04
Epoch 52/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7481 - loss: 0.6602 - val_ac
curacy: 0.7353 - val_loss: 0.5411 - learning_rate: 5.0000e-04
Epoch 53/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7976 - loss: 0.5625 - val_acc
uracy: 0.7353 - val_loss: 0.5362 - learning_rate: 5.0000e-04
Epoch 54/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7699 - loss: 0.5642 - val_acc
uracy: 0.7353 - val_loss: 0.5255 - learning_rate: 5.0000e-04
Epoch 55/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7845 - loss: 0.6328 - val_acc
uracy: 0.7353 - val_loss: 0.5227 - learning_rate: 5.0000e-04
Epoch 56/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7506 - loss: 0.6718 - val_ac
curacy: 0.7353 - val_loss: 0.5240 - learning_rate: 5.0000e-04
Epoch 57/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7970 - loss: 0.6448 - val_acc
uracy: 0.7206 - val_loss: 0.5236 - learning_rate: 5.0000e-04
Epoch 58/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8083 - loss: 0.6080 - val_ac
curacy: 0.7206 - val_loss: 0.5211 - learning_rate: 5.0000e-04
Epoch 59/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7864 - loss: 0.6248 - val_ac
curacy: 0.7206 - val_loss: 0.5121 - learning_rate: 5.0000e-04
Epoch 60/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7909 - loss: 0.5880 - val_ac
curacy: 0.7206 - val_loss: 0.5091 - learning_rate: 5.0000e-04
Epoch 61/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8536 - loss: 0.5051 - val_acc
uracy: 0.7206 - val_loss: 0.5074 - learning_rate: 5.0000e-04
Epoch 62/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7854 - loss: 0.5869 - val_acc
uracy: 0.7353 - val_loss: 0.4999 - learning_rate: 5.0000e-04
Epoch 63/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8004 - loss: 0.6429 - val_acc
uracy: 0.7206 - val_loss: 0.5048 - learning_rate: 5.0000e-04
Epoch 64/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8357 - loss: 0.7982 - val_ac
curacy: 0.7206 - val_loss: 0.5154 - learning_rate: 5.0000e-04
Epoch 65/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7879 - loss: 0.5493 - val_ac
curacy: 0.7353 - val_loss: 0.5106 - learning_rate: 5.0000e-04
Epoch 66/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - accuracy: 0.7823 - loss: 0.5733 - val_ac
curacy: 0.7500 - val_loss: 0.5082 - learning_rate: 5.0000e-04
Epoch 67/500
```

```
Epoch 67/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.8085 - loss: 0.7316 - val_ac
curacy: 0.7500 - val_loss: 0.5139 - learning_rate: 5.0000e-04
Epoch 68/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.8089 - loss: 0.5759 - val_ac
curacy: 0.7500 - val_loss: 0.5151 - learning_rate: 2.5000e-04
Epoch 69/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.7735 - loss: 0.5749 - val_ac
curacy: 0.7500 - val_loss: 0.5097 - learning_rate: 2.5000e-04
Epoch 70/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.8247 - loss: 0.4607 - val_ac
curacy: 0.7500 - val_loss: 0.5071 - learning_rate: 2.5000e-04
Epoch 71/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.8047 - loss: 0.5716 - val_ac
curacy: 0.7500 - val_loss: 0.5091 - learning_rate: 2.5000e-04
Epoch 72/500
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - accuracy: 0.8123 - loss: 0.6208 - val_ac
curacy: 0.7353 - val_loss: 0.5068 - learning_rate: 2.5000e-04
Model 3 Test Loss: 0.5433
Model 3 Test Accuracy: 0.7500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
```




```
              precision    recall  f1-score   support

     Class 0     0.9434    0.9901    0.9662       101
     Class 1     0.6667    0.3333    0.4444        42
     Class 2     0.2222    0.4615    0.3000        13
     Class 3     0.0000    0.0000    0.0000         4

    accuracy                         0.7500       160
   macro avg     0.4581    0.4462    0.4277       160
weighted avg     0.7886    0.7500    0.7509       160
```

	Metric	Score
0	Accuracy	0.750000

1	Precision (weighted)	0.788574
2	Recall (weighted)	0.750000
3	F1-score (weighted)	0.750945

Using ADASYN

In [ ]:

```
import pandas as pd
import requests
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.utils.class_weight import compute_class_weight
from sklearn.metrics import confusion_matrix, classification_report, precision_

# Import ADASYN
from imblearn.over_sampling import ADASYN


# --- Data Loading ---
url = "https://raw.githubusercontent.com/FilipeAMarques/ML-in-Biotech-Project/ma
response = requests.get(url)

# Save the content to a local file
with open("Merged.csv", "w") as f:
    f.write(response.text)

# Load the merged dataset into a pandas DataFrame
merged = pd.read_csv("Merged.csv")

# Display the first few rows of the merged DataFrame
display(merged.head())

# --- Remove rows where Severity is 3 ---
merged_filtered = merged[merged['Severity'] != 3].copy()
print(f"Original dataset size: {len(merged)}")
print(f"Dataset size after removing Severity 3: {len(merged_filtered)}")

# --- Define Features (X) and Labels (y) ---
# 'Severity' is the target variable
y = merged_filtered['Severity']

# Drop 'ID', 'Diagnosis', 'Recurrence', and 'Severity' to get the features (X)
X = merged_filtered.drop(['ID', 'Diagnosis', 'Recurrence', 'Severity'], axis=1)

# --- Data Preprocessing ---
# Split data into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.4, random_
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.7,

# Apply ADASYN only on the training data
adasyn = ADASYN(random_state=42)
X_train_res, y_train_res = adasyn.fit_resample(X_train, y_train)

# Scale the features
scaler = StandardScaler()
# Use original split if not using ADASYN
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)
```

```python
# ============================
# Improved Model 3: Severity prediction
# ============================

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from sklearn.utils.class_weight import compute_class_weight
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# --- Prepare labels for Severity ---
# Ensure Severity is integer encoded
# Use original split if not using ADASYN
y_train_sev = y_train.values.astype(int)
y_val_sev   = y_val.values.astype(int)
y_test_sev  = y_test.values.astype(int)

# --- Compute class weights (to handle imbalance) ---
# Recompute class weights with the filtered data
class_weights = compute_class_weight(
    class_weight='balanced',
    classes=np.unique(y_train_sev),
    y=y_train_sev
)
class_weights = dict(enumerate(class_weights))
print("Class Weights (after removing Severity 3):", class_weights)


# --- Build improved Model 3 ---
model_3 = Sequential([
    Dense(128, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    BatchNormalization(),
    Dropout(0.3),

    Dense(64, activation='relu'),
    BatchNormalization(),
    Dropout(0.3),

    Dense(16, activation='relu'),
    BatchNormalization(),
    Dropout(0.2),

    Dense(3, activation='softmax')  # Now only 3 severity classes
])

model_3.compile(
    optimizer=Adam(learning_rate=1e-3),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

model_3.summary()

# --- Callbacks ---
early_stop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weight
reduce_lr  = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, min_

# --- Train ---
history_3 = model_3.fit(
    X_train_scaled, y_train_sev,
```

```
    X_train_scaled, y_train_sev,
    validation_data=(X_val_scaled, y_val_sev),
    epochs=200,
    batch_size=32,
    class_weight=class_weights,
    callbacks=[early_stop, reduce_lr],
    verbose=1
)

# --- Evaluate ---
loss_3, acc_3 = model_3.evaluate(X_test_scaled, y_test_sev, verbose=0)
print(f"Model 3 Test Loss (after removing Severity 3): {loss_3:.4f}")
print(f"Model 3 Test Accuracy (after removing Severity 3): {acc_3:.4f}")

# --- Predictions ---
y_pred_proba_3 = model_3.predict(X_test_scaled)
y_pred_3 = np.argmax(y_pred_proba_3, axis=1)

# --- Confusion Matrix ---
conf_matrix_3 = confusion_matrix(y_test_sev, y_pred_3)
plt.figure(figsize=(8,6))
sns.heatmap(conf_matrix_3, annot=True, fmt='d', cmap='Blues',
            xticklabels=[f'Pred {i}' for i in np.unique(y_test_sev)],
            yticklabels=[f'Actual {i}' for i in np.unique(y_test_sev)])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Model 3 (Severity) using ADASYN")
plt.show()

# --- Classification Report ---
report = classification_report(
    y_test_sev, y_pred_3,
    target_names=[f"Class {i}" for i in np.unique(y_test_sev)],
    digits=4
)
print(report)

# --- Performance Metrics DataFrame ---
performance_metrics_3 = pd.DataFrame({
    'Metric': ['Accuracy', 'Precision (weighted)', 'Recall (weighted)', 'F1-sco
    'Score': [accuracy_score(y_test_sev, y_pred_3),
              precision_score(y_test_sev, y_pred_3, average='weighted'),
              recall_score(y_test_sev, y_pred_3, average='weighted'),
              f1_score(y_test_sev, y_pred_3, average='weighted')]
})
display(performance_metrics_3)
```

	ID	radius1	texture1	perimeter1	area1	smoothness1	compactness1	concavity1
0	842302	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001
1	842517	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869
2	84300903	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974
3	84348301	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414
4	84358402	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980

5 rows × 34 columns

```
Original dataset size: 569
Dataset size after removing Severity 3: 555
Class Weights (after removing Severity 3): {0: np.float64(0.5186915887850467), 1:
np.float64(1.2197802197802199), 2: np.float64(3.9642857142857144)}
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWar
ning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Seq
uential models, prefer using an `Input(shape)` object as the first layer in the mo
```

```
dential models, prefer using an `Input(shape)` object as the first layer in the mo
del instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential_20"

Layer (type)	Output Shape	Param #
dense_83 (Dense)	(None, 128)	3,968
batch_normalization_63 (BatchNormalization)	(None, 128)	512
dropout_63 (Dropout)	(None, 128)	0
dense_84 (Dense)	(None, 64)	8,256
batch_normalization_64 (BatchNormalization)	(None, 64)	256
dropout_64 (Dropout)	(None, 64)	0
dense_85 (Dense)	(None, 16)	1,040
batch_normalization_65 (BatchNormalization)	(None, 16)	64
dropout_65 (Dropout)	(None, 16)	0
dense_86 (Dense)	(None, 3)	51

Total params: 14,147 (55.26 KB)

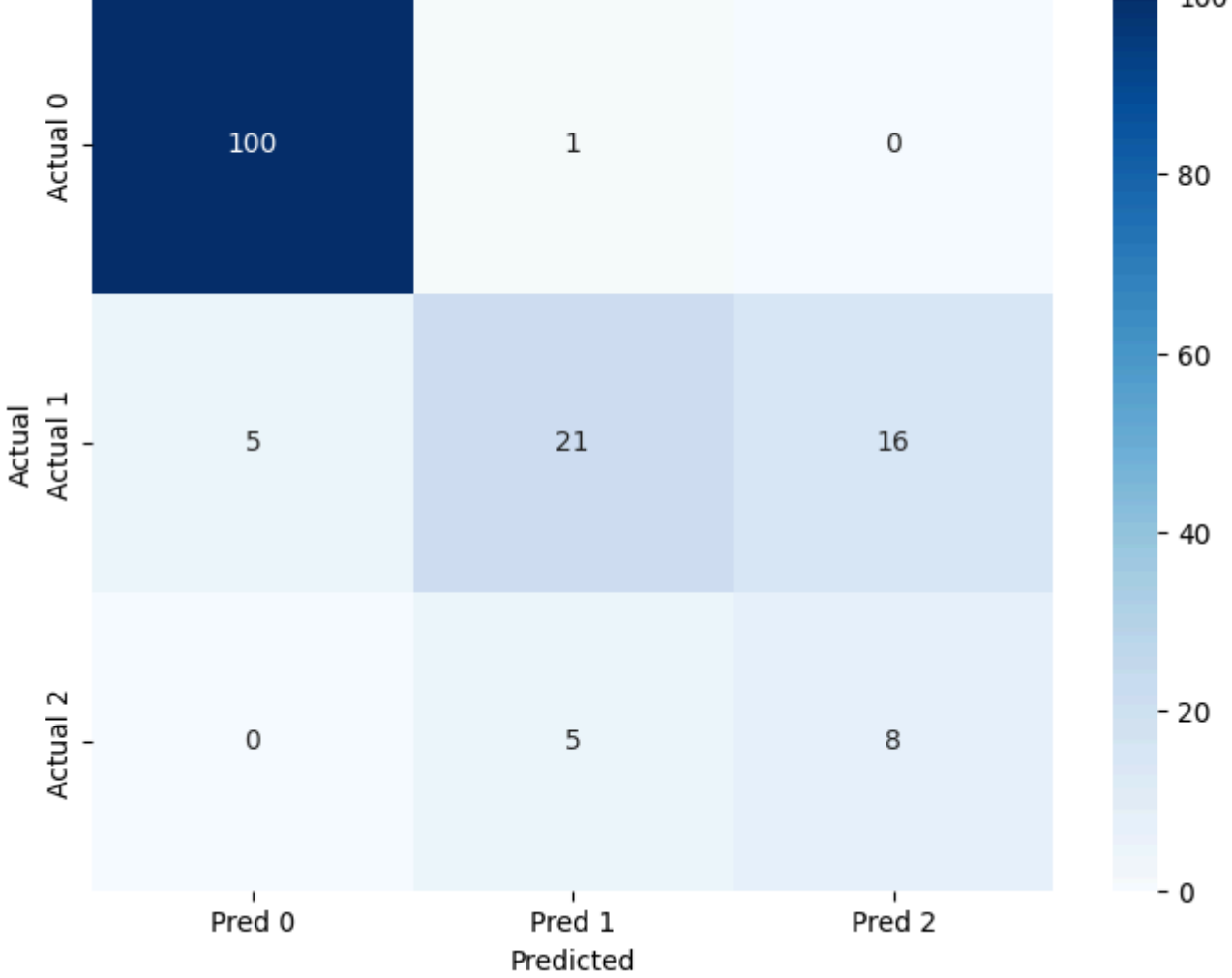
Trainable params: 13,731 (53.64 KB)

Non-trainable params: 416 (1.62 KB)

```
Epoch 1/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 3s 35ms/step - accuracy: 0.3224 - loss: 1.6635 - val_ac
curacy: 0.5909 - val_loss: 0.8920 - learning_rate: 0.0010
Epoch 2/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6026 - loss: 0.9903 - val_ac
curacy: 0.8030 - val_loss: 0.7696 - learning_rate: 0.0010
Epoch 3/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.5828 - loss: 0.8905 - val_ac
curacy: 0.8030 - val_loss: 0.6932 - learning_rate: 0.0010
Epoch 4/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.6573 - loss: 0.8286 - val_ac
curacy: 0.8030 - val_loss: 0.6269 - learning_rate: 0.0010
Epoch 5/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.6891 - loss: 0.7115 - val_ac
curacy: 0.7727 - val_loss: 0.5811 - learning_rate: 0.0010
Epoch 6/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.6688 - loss: 0.7172 - val_ac
curacy: 0.7879 - val_loss: 0.5530 - learning_rate: 0.0010
Epoch 7/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.7081 - loss: 0.6523 - val_ac
curacy: 0.7576 - val_loss: 0.5213 - learning_rate: 0.0010
Epoch 8/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.7176 - loss: 0.6539 - val_ac
curacy: 0.7576 - val_loss: 0.5042 - learning_rate: 0.0010
Epoch 9/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7285 - loss: 0.6421 - val_ac
curacy: 0.7727 - val_loss: 0.4812 - learning_rate: 0.0010
Epoch 10/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7593 - loss: 0.6353 - val_ac
curacy: 0.7727 - val_loss: 0.4549 - learning_rate: 0.0010
Epoch 11/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - accuracy: 0.7660 - loss: 0.6553 - val_ac
curacy: 0.7727 - val_loss: 0.4425 - learning_rate: 0.0010
Epoch 12/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.8009 - loss: 0.5539 - val_ac
```

```
curacy: 0.7727 - val_loss: 0.4275 - learning_rate: 0.0010
Epoch 13/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7427 - loss: 0.5785 - val_ac
curacy: 0.7727 - val_loss: 0.4178 - learning_rate: 0.0010
Epoch 14/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7549 - loss: 0.6762 - val_acc
uracy: 0.7727 - val_loss: 0.4120 - learning_rate: 0.0010
Epoch 15/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7829 - loss: 0.5311 - val_ac
curacy: 0.7879 - val_loss: 0.3987 - learning_rate: 0.0010
Epoch 16/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7899 - loss: 0.5546 - val_ac
curacy: 0.7879 - val_loss: 0.3908 - learning_rate: 0.0010
Epoch 17/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8013 - loss: 0.5068 - val_acc
uracy: 0.7727 - val_loss: 0.3949 - learning_rate: 0.0010
Epoch 18/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8312 - loss: 0.4800 - val_acc
uracy: 0.7727 - val_loss: 0.3992 - learning_rate: 0.0010
Epoch 19/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7866 - loss: 0.5684 - val_acc
uracy: 0.7727 - val_loss: 0.3901 - learning_rate: 0.0010
Epoch 20/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8104 - loss: 0.6133 - val_ac
curacy: 0.7727 - val_loss: 0.3908 - learning_rate: 0.0010
Epoch 21/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.8079 - loss: 0.4690 - val_ac
curacy: 0.7576 - val_loss: 0.3789 - learning_rate: 0.0010
Epoch 22/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8173 - loss: 0.5470 - val_acc
uracy: 0.7576 - val_loss: 0.3805 - learning_rate: 0.0010
Epoch 23/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8076 - loss: 0.6519 - val_acc
uracy: 0.7727 - val_loss: 0.3918 - learning_rate: 0.0010
Epoch 24/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8412 - loss: 0.4959 - val_acc
uracy: 0.7576 - val_loss: 0.3872 - learning_rate: 0.0010
Epoch 25/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8459 - loss: 0.4611 - val_acc
uracy: 0.7576 - val_loss: 0.3869 - learning_rate: 0.0010
Epoch 26/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8150 - loss: 0.5354 - val_acc
uracy: 0.7576 - val_loss: 0.3837 - learning_rate: 0.0010
Epoch 27/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8469 - loss: 0.4190 - val_acc
uracy: 0.7727 - val_loss: 0.3802 - learning_rate: 5.0000e-04
Epoch 28/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8496 - loss: 0.4219 - val_ac
curacy: 0.7727 - val_loss: 0.3765 - learning_rate: 5.0000e-04
Epoch 29/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8597 - loss: 0.4577 - val_ac
curacy: 0.7727 - val_loss: 0.3744 - learning_rate: 5.0000e-04
Epoch 30/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8544 - loss: 0.4413 - val_ac
curacy: 0.7727 - val_loss: 0.3713 - learning_rate: 5.0000e-04
Epoch 31/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8266 - loss: 0.4860 - val_acc
uracy: 0.7727 - val_loss: 0.3742 - learning_rate: 5.0000e-04
Epoch 32/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8095 - loss: 0.5833 - val_acc
uracy: 0.7727 - val_loss: 0.3814 - learning_rate: 5.0000e-04
Epoch 33/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8566 - loss: 0.4215 - val_acc
uracy: 0.7727 - val_loss: 0.3748 - learning_rate: 5.0000e-04
Epoch 34/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8425 - loss: 0.4356 - val_ac
curacy: 0.7727 - val_loss: 0.3696 - learning_rate: 5.0000e-04
Epoch 35/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8996 - loss: 0.3844 - val_ac
```

```
curacy: 0.7727 - val_loss: 0.3659 - learning_rate: 5.0000e-04
Epoch 36/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8703 - loss: 0.4492 - val_ac
curacy: 0.7727 - val_loss: 0.3696 - learning_rate: 5.0000e-04
Epoch 37/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8180 - loss: 0.4709 - val_ac
curacy: 0.7879 - val_loss: 0.3682 - learning_rate: 5.0000e-04
Epoch 38/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8657 - loss: 0.4076 - val_ac
curacy: 0.7727 - val_loss: 0.3722 - learning_rate: 5.0000e-04
Epoch 39/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8610 - loss: 0.3746 - val_acc
uracy: 0.7727 - val_loss: 0.3683 - learning_rate: 5.0000e-04
Epoch 40/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8696 - loss: 0.4637 - val_ac
curacy: 0.7727 - val_loss: 0.3658 - learning_rate: 5.0000e-04
Epoch 41/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8508 - loss: 0.4780 - val_acc
uracy: 0.7727 - val_loss: 0.3638 - learning_rate: 2.5000e-04
Epoch 42/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8646 - loss: 0.4379 - val_ac
curacy: 0.7727 - val_loss: 0.3603 - learning_rate: 2.5000e-04
Epoch 43/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8847 - loss: 0.4158 - val_acc
uracy: 0.7727 - val_loss: 0.3606 - learning_rate: 2.5000e-04
Epoch 44/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8600 - loss: 0.4413 - val_ac
curacy: 0.7727 - val_loss: 0.3592 - learning_rate: 2.5000e-04
Epoch 45/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8839 - loss: 0.3557 - val_ac
curacy: 0.7727 - val_loss: 0.3591 - learning_rate: 2.5000e-04
Epoch 46/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8659 - loss: 0.3777 - val_ac
curacy: 0.7879 - val_loss: 0.3564 - learning_rate: 2.5000e-04
Epoch 47/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8649 - loss: 0.4851 - val_acc
uracy: 0.7879 - val_loss: 0.3590 - learning_rate: 2.5000e-04
Epoch 48/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8940 - loss: 0.4478 - val_acc
uracy: 0.7879 - val_loss: 0.3644 - learning_rate: 2.5000e-04
Epoch 49/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8839 - loss: 0.3810 - val_ac
curacy: 0.7879 - val_loss: 0.3659 - learning_rate: 2.5000e-04
Epoch 50/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8462 - loss: 0.3749 - val_acc
uracy: 0.7727 - val_loss: 0.3680 - learning_rate: 2.5000e-04
Epoch 51/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8271 - loss: 0.4702 - val_acc
uracy: 0.7727 - val_loss: 0.3672 - learning_rate: 2.5000e-04
Epoch 52/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8633 - loss: 0.3756 - val_acc
uracy: 0.7879 - val_loss: 0.3687 - learning_rate: 1.2500e-04
Epoch 53/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8474 - loss: 0.4499 - val_ac
curacy: 0.7879 - val_loss: 0.3680 - learning_rate: 1.2500e-04
Epoch 54/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8630 - loss: 0.4157 - val_ac
curacy: 0.7879 - val_loss: 0.3701 - learning_rate: 1.2500e-04
Epoch 55/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8522 - loss: 0.4183 - val_acc
uracy: 0.7879 - val_loss: 0.3704 - learning_rate: 1.2500e-04
Epoch 56/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8727 - loss: 0.4552 - val_acc
uracy: 0.7879 - val_loss: 0.3699 - learning_rate: 1.2500e-04
Model 3 Test Loss (after removing Severity 3): 0.3734
Model 3 Test Accuracy (after removing Severity 3): 0.8269
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
```




```
              precision    recall  f1-score   support

     Class 0     0.9524    0.9901    0.9709       101
     Class 1     0.7778    0.5000    0.6087        42
     Class 2     0.3333    0.6154    0.4324        13

    accuracy                         0.8269       156
   macro avg     0.6878    0.7018    0.6707       156
weighted avg     0.8538    0.8269    0.8285       156
```

	Metric	Score
0	Accuracy	0.826923
1	Precision (weighted)	0.853785
2	Recall (weighted)	0.826923
3	F1-score (weighted)	0.828494

Using SMOTE

In [ ]:

```
import pandas as pd
import requests
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.utils.class_weight import compute_class_weight
from sklearn.metrics import confusion_matrix, classification_report, precision_

# Import SMOTE
from imblearn.over_sampling import SMOTE
```

```
# --- Data Loading ---
url = "https://raw.githubusercontent.com/FilipeAMarques/ML-in-Biotech-Project/ma
response = requests.get(url)

# Save the content to a local file
with open("Merged.csv", "w") as f:
    f.write(response.text)

# Load the merged dataset into a pandas DataFrame
merged = pd.read_csv("Merged.csv")

# Display the first few rows of the merged DataFrame
display(merged.head())

# --- Remove rows where Severity is 3 ---
merged_filtered = merged[merged['Severity'] != 3].copy()
print(f"Original dataset size: {len(merged)}")
print(f"Dataset size after removing Severity 3: {len(merged_filtered)}")

# --- Define Features (X) and Labels (y) ---
# 'Severity' is the target variable
y = merged_filtered['Severity']

# Drop 'ID', 'Diagnosis', 'Recurrence', and 'Severity' to get the features (X)
X = merged_filtered.drop(['ID', 'Diagnosis', 'Recurrence', 'Severity'], axis=1)

# --- Data Preprocessing ---
# Split data into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.4, random_
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.7,

# Print class distribution in training data before SMOTE
print("Class distribution in y_train before SMOTE:")
print(y_train.value_counts())


# Apply SMOTE only on the training data
smote = SMOTE(random_state=42, k_neighbors=3) # Reduced k_neighbors
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)


# Scale the features
scaler = StandardScaler()
# Use original split if not using ADASYN
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)


# ============================
# Improved Model 3: Severity prediction
# ============================

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from sklearn.utils.class_weight import compute_class_weight
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

```python
# --- Prepare labels for Severity ---
# Ensure Severity is integer encoded
# Use original split if not using ADASYN
y_train_sev = y_train.values.astype(int)
y_val_sev   = y_val.values.astype(int)
y_test_sev  = y_test.values.astype(int)

# --- Compute class weights (to handle imbalance) ---
# Recompute class weights with the filtered data
class_weights = compute_class_weight(
    class_weight='balanced',
    classes=np.unique(y_train_sev),
    y=y_train_sev
)
class_weights = dict(enumerate(class_weights))
print("Class Weights (after removing Severity 3):", class_weights)


# --- Build improved Model 3 ---
model_3 = Sequential([
    Dense(128, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    BatchNormalization(),
    Dropout(0.3),

    Dense(64, activation='relu'),
    BatchNormalization(),
    Dropout(0.3),

    Dense(16, activation='relu'),
    BatchNormalization(),
    Dropout(0.2),

    Dense(3, activation='softmax')  # Now only 3 severity classes
])

model_3.compile(
    optimizer=Adam(learning_rate=1e-3),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

model_3.summary()

# --- Callbacks ---
early_stop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weight
reduce_lr  = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, min_

# --- Train ---
history_3 = model_3.fit(
    X_train_scaled, y_train_sev,
    validation_data=(X_val_scaled, y_val_sev),
    epochs=200,
    batch_size=32,
    class_weight=class_weights,
    callbacks=[early_stop, reduce_lr],
    verbose=1
)

# --- Evaluate ---
loss_3, acc_3 = model_3.evaluate(X_test_scaled, y_test_sev, verbose=0)
print(f"Model 3 Test Loss (after removing Severity 3): {loss_3:.4f}")
print(f"Model 3 Test Accuracy (after removing Severity 3): {acc_3:.4f}")

# --- Predictions ---
y_pred_proba_3 = model_3.predict(X_test_scaled)
y_pred_3 = np.argmax(y_pred_proba_3, axis=1)
```

```python
# --- Confusion Matrix ---
conf_matrix_3 = confusion_matrix(y_test_sev, y_pred_3)
plt.figure(figsize=(8,6))
sns.heatmap(conf_matrix_3, annot=True, fmt='d', cmap='Blues',
            xticklabels=[f'Pred {i}' for i in np.unique(y_test_sev)],
            yticklabels=[f'Actual {i}' for i in np.unique(y_test_sev)])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Model 3 (Severity) using SMOTE")
plt.show()

# --- Classification Report ---
report = classification_report(
    y_test_sev, y_pred_3,
    target_names=[f"Class {i}" for i in np.unique(y_test_sev)],
    digits=4
)
print(report)

# --- Performance Metrics DataFrame ---
performance_metrics_3 = pd.DataFrame({
    'Metric': ['Accuracy', 'Precision (weighted)', 'Recall (weighted)', 'F1-sco
    'Score': [accuracy_score(y_test_sev, y_pred_3),
              precision_score(y_test_sev, y_pred_3, average='weighted'),
              recall_score(y_test_sev, y_pred_3, average='weighted'),
              f1_score(y_test_sev, y_pred_3, average='weighted')]
})
display(performance_metrics_3)
```

	ID	radius1	texture1	perimeter1	area1	smoothness1	compactness1	concavity1
0	842302	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001
1	842517	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869
2	84300903	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974
3	84348301	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414
4	84358402	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980

5 rows × 34 columns

```
Original dataset size: 569
Dataset size after removing Severity 3: 555
Class distribution in y_train before SMOTE:
Severity
0    214
1     91
2     28
Name: count, dtype: int64
Class Weights (after removing Severity 3): {0: np.float64(0.5186915887850467), 1:
np.float64(1.2197802197802199), 2: np.float64(3.9642857142857144)}
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWar
ning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Seq
uential models, prefer using an `Input(shape)` object as the first layer in the mo
del instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential_25"

Layer (type)	Output Shape	Param #
dense_106 (Dense)	(None, 128)	3,968
batch_normalization_81 (BatchNormalization)	(None, 128)	512

dropout_81 (Dropout)	(None, 128)	0
dense_107 (Dense)	(None, 64)	8,256
batch_normalization_82 (BatchNormalization)	(None, 64)	256
dropout_82 (Dropout)	(None, 64)	0
dense_108 (Dense)	(None, 16)	1,040
batch_normalization_83 (BatchNormalization)	(None, 16)	64
dropout_83 (Dropout)	(None, 16)	0
dense_109 (Dense)	(None, 3)	51

Total params: 14,147 (55.26 KB)

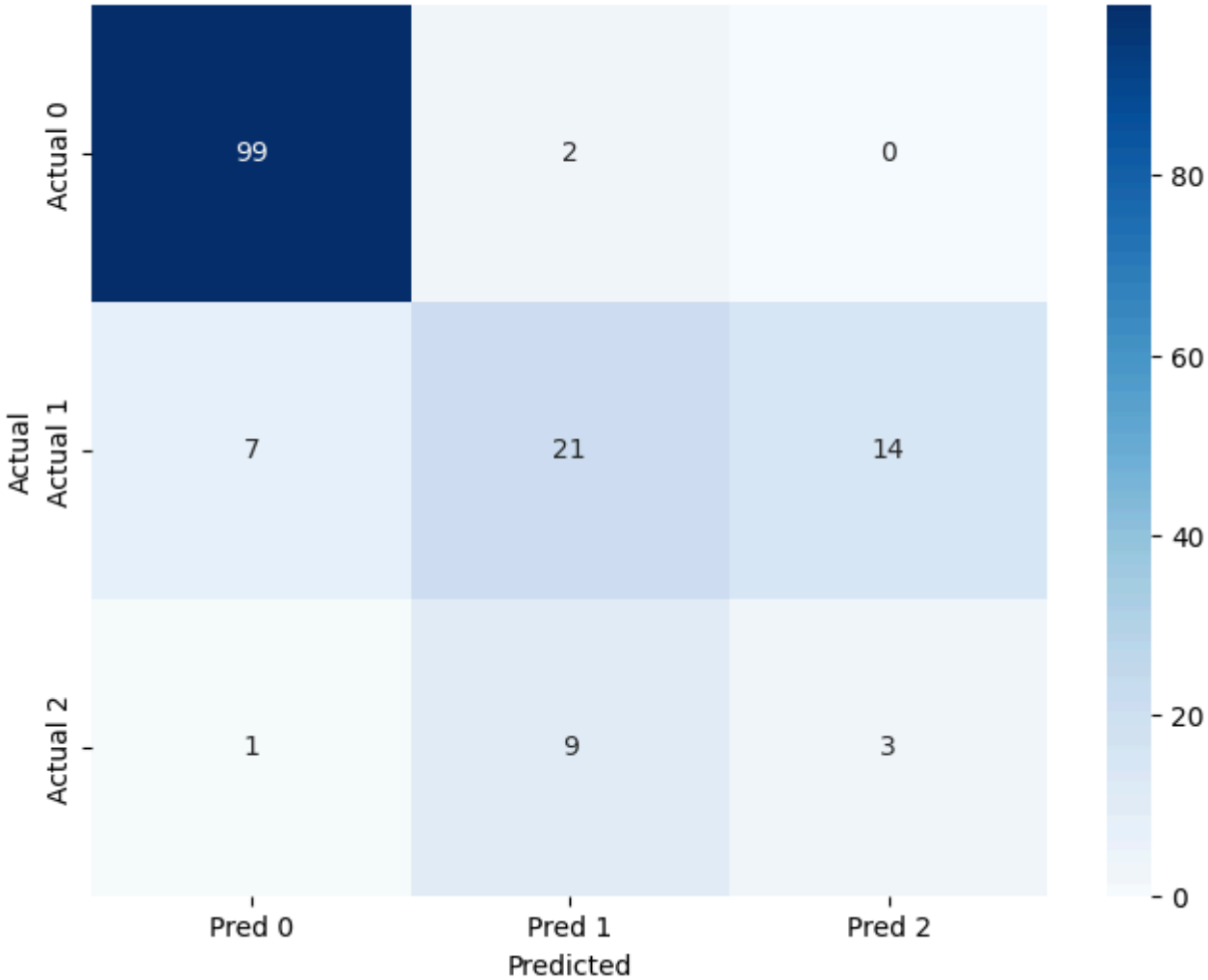
Trainable params: 13,731 (53.64 KB)

Non-trainable params: 416 (1.62 KB)

```
Epoch 1/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 3s 53ms/step - accuracy: 0.3153 - loss: 1.9183 - val_ac
curacy: 0.4697 - val_loss: 1.0414 - learning_rate: 0.0010
Epoch 2/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - accuracy: 0.4755 - loss: 1.4820 - val_ac
curacy: 0.6061 - val_loss: 0.9361 - learning_rate: 0.0010
Epoch 3/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.5109 - loss: 1.1965 - val_ac
curacy: 0.7273 - val_loss: 0.8334 - learning_rate: 0.0010
Epoch 4/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - accuracy: 0.6186 - loss: 1.1715 - val_ac
curacy: 0.7879 - val_loss: 0.7593 - learning_rate: 0.0010
Epoch 5/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - accuracy: 0.6493 - loss: 0.9393 - val_ac
curacy: 0.8182 - val_loss: 0.6959 - learning_rate: 0.0010
Epoch 6/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - accuracy: 0.6484 - loss: 1.0520 - val_ac
curacy: 0.8030 - val_loss: 0.6389 - learning_rate: 0.0010
Epoch 7/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - accuracy: 0.7319 - loss: 0.8820 - val_ac
curacy: 0.8030 - val_loss: 0.5858 - learning_rate: 0.0010
Epoch 8/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7068 - loss: 0.6764 - val_ac
curacy: 0.8485 - val_loss: 0.5441 - learning_rate: 0.0010
Epoch 9/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7356 - loss: 0.7469 - val_ac
curacy: 0.8333 - val_loss: 0.5064 - learning_rate: 0.0010
Epoch 10/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7943 - loss: 0.6481 - val_ac
curacy: 0.8485 - val_loss: 0.4823 - learning_rate: 0.0010
Epoch 11/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7882 - loss: 0.6210 - val_ac
curacy: 0.8485 - val_loss: 0.4602 - learning_rate: 0.0010
Epoch 12/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8056 - loss: 0.6020 - val_ac
curacy: 0.8636 - val_loss: 0.4467 - learning_rate: 0.0010
Epoch 13/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7458 - loss: 0.6222 - val_ac
curacy: 0.8636 - val_loss: 0.4289 - learning_rate: 0.0010
Epoch 14/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8081 - loss: 0.6267 - val_ac
curacy: 0.8636 - val_loss: 0.4193 - learning_rate: 0.0010
Epoch 15/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7645 - loss: 0.6723 - val_ac
curacy: 0.8333 - val_loss: 0.4223 - learning_rate: 0.0010
Epoch 16/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7996 - loss: 0.5350 - val_acc
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7996 - loss: 0.5330 - val_acc
uracy: 0.8485 - val_loss: 0.4263 - learning_rate: 0.0010
Epoch 17/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8013 - loss: 0.6036 - val_ac
curacy: 0.8030 - val_loss: 0.4342 - learning_rate: 0.0010
Epoch 18/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8048 - loss: 0.6776 - val_ac
curacy: 0.8182 - val_loss: 0.4282 - learning_rate: 0.0010
Epoch 19/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8165 - loss: 0.5393 - val_ac
curacy: 0.8182 - val_loss: 0.4153 - learning_rate: 0.0010
Epoch 20/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8095 - loss: 0.5377 - val_ac
curacy: 0.8333 - val_loss: 0.4104 - learning_rate: 0.0010
Epoch 21/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7925 - loss: 0.5503 - val_ac
curacy: 0.8182 - val_loss: 0.4118 - learning_rate: 0.0010
Epoch 22/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8420 - loss: 0.4936 - val_ac
curacy: 0.8182 - val_loss: 0.4219 - learning_rate: 0.0010
Epoch 23/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8496 - loss: 0.5224 - val_ac
curacy: 0.8182 - val_loss: 0.4246 - learning_rate: 0.0010
Epoch 24/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8317 - loss: 0.5741 - val_ac
curacy: 0.8030 - val_loss: 0.4076 - learning_rate: 0.0010
Epoch 25/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8392 - loss: 0.5988 - val_ac
curacy: 0.8333 - val_loss: 0.3991 - learning_rate: 0.0010
Epoch 26/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8585 - loss: 0.5348 - val_ac
curacy: 0.8333 - val_loss: 0.3922 - learning_rate: 0.0010
Epoch 27/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8363 - loss: 0.4083 - val_ac
curacy: 0.8485 - val_loss: 0.3902 - learning_rate: 0.0010
Epoch 28/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8404 - loss: 0.5236 - val_ac
curacy: 0.8636 - val_loss: 0.3948 - learning_rate: 0.0010
Epoch 29/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8195 - loss: 0.5491 - val_ac
curacy: 0.8333 - val_loss: 0.4000 - learning_rate: 0.0010
Epoch 30/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8250 - loss: 0.5365 - val_ac
curacy: 0.8333 - val_loss: 0.4006 - learning_rate: 0.0010
Epoch 31/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8398 - loss: 0.4554 - val_ac
curacy: 0.8182 - val_loss: 0.3937 - learning_rate: 0.0010
Epoch 32/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.8276 - loss: 0.5307 - val_ac
curacy: 0.8030 - val_loss: 0.3977 - learning_rate: 0.0010
Epoch 33/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8283 - loss: 0.5070 - val_ac
curacy: 0.8030 - val_loss: 0.3953 - learning_rate: 5.0000e-04
Epoch 34/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8483 - loss: 0.4914 - val_ac
curacy: 0.8030 - val_loss: 0.3972 - learning_rate: 5.0000e-04
Epoch 35/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8499 - loss: 0.4276 - val_ac
curacy: 0.8030 - val_loss: 0.3974 - learning_rate: 5.0000e-04
Epoch 36/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8758 - loss: 0.4143 - val_ac
curacy: 0.8030 - val_loss: 0.3917 - learning_rate: 5.0000e-04
Epoch 37/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8744 - loss: 0.4864 - val_ac
curacy: 0.8030 - val_loss: 0.3967 - learning_rate: 5.0000e-04
Model 3 Test Loss (after removing Severity 3): 0.4022
Model 3 Test Accuracy (after removing Severity 3): 0.7885
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
```

Confusion Matrix - Model 3 (Severity) excluding Class 3




```
              precision    recall  f1-score   support

     Class 0     0.9252    0.9802    0.9519       101
     Class 1     0.6562    0.5000    0.5676        42
     Class 2     0.1765    0.2308    0.2000        13

    accuracy                         0.7885       156
   macro avg     0.5860    0.5703    0.5732       156
weighted avg     0.7904    0.7885    0.7858       156
```

	Metric	Score
0	Accuracy	0.788462
1	Precision (weighted)	0.790418
2	Recall (weighted)	0.788462