part2

December 5, 2022

1 Part 2: Illness Detection

1.0.1 This part determines level of illness within sheep based on sitting/standing ground truth data

[ ]:
```
# Imports
import pandas as pd
import numpy as np
import statistics
from sklearn.preprocessing import StandardScaler
```

[ ]:
```
# read in the data first
# note that D is missing several days
files = ['A.csv', 'B.csv', 'U.csv', 'V.csv', 'W.csv', 'D.csv']
sheep_raw = []
data_dir = 'data/'
for file in files:
    file = data_dir + file
    df = pd.read_csv(file)
    # drop unnecessary columns
    df = df.drop(columns=df.columns[3:])
    df = df.drop(columns=df.columns[0:2])

    # rename column
    df.columns = ['Standing']

    # change accelerometer value to 1 if standing, 0 if sitting
    # THERE'S A SPECIAL CASE FOR SHEEP V and F, THE ACCELEROMETER WAS UPSIDE␣
↪DOWN
    if file == 'data/V.csv' or file == 'data/F.csv':
        df['Standing'] = np.where(df['Standing'] > 0.49, 1, 0)
    else:
        df['Standing'] = np.where(df['Standing'] < -0.4, 1, 0)
    sheep_raw.append(df)
```

[ ]:
```
# version of reading in ALL data
```

```
all_files = ['A.csv', 'B.csv', 'C.csv', 'D.csv', 'E.csv', 'F.csv', 'G.csv', 'H.
↪csv', 'I.csv', 'J.csv', 'K.csv', 'L.csv', 'M.csv', 'N.csv', 'O_0.csv', 'P.
↪csv', 'Q.csv', 'R_0.csv', 'S.csv', 'T.csv', 'U.csv', 'V.csv', 'W.csv']

sheep_raw_full = []
data_dir = 'data/'
for file in all_files:
    file = data_dir + file
    df = pd.read_csv(file)
    # drop unnecessary columns
    df = df.drop(columns=df.columns[3:])
    df = df.drop(columns=df.columns[0:2])

    # rename column
    df.columns = ['Standing']

    # change accelerometer value to 1 if standing, 0 if sitting
    # THERE'S A SPECIAL CASE FOR SHEEP V and F, THE ACCELEROMETER WAS UPSIDE␣
↪DOWN
    if file == 'data/V.csv' or file == 'data/F.csv':
        df['Standing'] = np.where(df['Standing'] > 0.49, 1, 0)
    else:
        df['Standing'] = np.where(df['Standing'] < -0.4, 1, 0)
    sheep_raw_full.append(df)
```

[ ]:
```
for sheep, file in zip(sheep_raw_full, all_files):
    temp = sheep['Standing'].to_list()
    total_sit = 0
    total_stand = 0
    for instance in temp:
        total_stand += instance
        if instance == 0:
            total_sit += 1
    if total_sit/(total_sit+total_stand) > 0.9:
        print(f'{file}: {total_stand}')
    print(f'Sit percentage: {round(total_sit/(total_sit+total_stand), 4) *␣
↪100}%\nTotal Size: {total_stand+total_sit}\n')
```

```
Sit percentage: 63.01%
Total Size: 21686

Sit percentage: 62.67%
Total Size: 21686

Sit percentage: 63.6%
Total Size: 21686
```

```
Sit percentage: 57.4%
Total Size: 5831

Sit percentage: 64.57000000000001%
Total Size: 21686

Sit percentage: 79.56%
Total Size: 21686

Sit percentage: 61.82%
Total Size: 21686

Sit percentage: 65.99000000000001%
Total Size: 21686

Sit percentage: 52.92%
Total Size: 21686

Sit percentage: 56.64%
Total Size: 21686

Sit percentage: 49.27%
Total Size: 21686

Sit percentage: 70.37%
Total Size: 21686

Sit percentage: 72.84%
Total Size: 21686

Sit percentage: 63.93%
Total Size: 21686

Sit percentage: 63.849999999999994%
Total Size: 21686

Sit percentage: 72.22%
Total Size: 2977

Sit percentage: 53.38%
Total Size: 21686

Sit percentage: 65.38000000000001%
Total Size: 21686

Sit percentage: 49.25%
Total Size: 21686
```

```
Sit percentage: 69.17999999999999%
Total Size: 21686

Sit percentage: 64.3%
Total Size: 21686

Sit percentage: 64.92%
Total Size: 21686

Sit percentage: 63.7%
Total Size: 21686
```

[ ]:
```
print(sheep_raw[0].shape)
print(sheep_raw[1].shape)
print(sheep_raw[2].shape)
print(sheep_raw[3].shape)
print(sheep_raw[4].shape)
print(sheep_raw[5].shape)
```

```
(21686, 1)
(21686, 1)
(21686, 1)
(21686, 1)
(21686, 1)
(5831, 1)
```

1.0.2 Windowing

[ ]:
```
# Function to window our samples
def splitSamples(df: pd.DataFrame, samples_per_split: int) -> list:
    all_samples = []
    start_index = 0
    end_index = samples_per_split
    while end_index < len(df):
        temp_df = df[start_index:end_index]
        all_samples.append(temp_df)
        start_index += samples_per_split // 2
        end_index += samples_per_split // 2

    return all_samples
```

[ ]:
```
# Call the function with each dataframe to get sample lists

# The accelerometer records 2880 times a day, so let's try a window of that size
SAMPLES_PER_SPLIT = 2880
for i in range(len(sheep_raw)):
```

```
    sheep_raw[i] = splitSamples(sheep_raw[i], SAMPLES_PER_SPLIT)

for i in range(len(sheep_raw_full)):
    sheep_raw_full[i] = splitSamples(sheep_raw_full[i], SAMPLES_PER_SPLIT)
```

[ ]:
```
# Convert dataframes into just a list of sit/stand
for i in range(len(sheep_raw)):
    for j in range(len(sheep_raw[i])):
        sheep_raw[i][j] = sheep_raw[i][j]['Standing'].tolist()

# Convert dataframes into just a list of sit/stand
for i in range(len(sheep_raw_full)):
    for j in range(len(sheep_raw_full[i])):
        sheep_raw_full[i][j] = sheep_raw_full[i][j]['Standing'].tolist()
```

1.0.3 Feature Engineering

[ ]:
```
def extractFeatures(windows: list) -> list:
    """
    There are a few features we should extract here
    1. total sits
    2. total stands
    3. changes (how many times did it go from sit to stand or vice versa)
    4. mean (essentially what percentage of the time was it standing)
    5. mode (was it standing or sitting more)
    6. longest sit
    7. longest stand
    8. Average length of sitting bouts
    9. Average length of standing bouts
    10. Std dev of sitting bouts
    11. Standard deviation of standing bouts
    """
    X = []

    for window in windows:
        temp = []
        total_sit = 0
        total_stand = 0
        total_changes = 0
        longest_sit = 0
        longest_stand = 0
        average_sit = []
        average_stand = []

        current_sit_streak = 0
        current_stand_streak = 0
```

```
    # mean and median can be calculated without looping
    mean = np.mean(window)
    mode = statistics.mode(window)

    prev_state = window[0]

    for sample in window:
        if sample == 0:
            total_sit += 1
            current_sit_streak += 1
            if current_sit_streak > longest_sit:
                longest_sit = current_sit_streak
            if prev_state != sample:
                total_changes += 1
                average_stand.append(current_stand_streak)
                current_stand_streak = 0
        else:
            total_stand += 1
            current_stand_streak += 1
            if current_stand_streak > longest_stand:
                longest_stand = current_stand_streak
            if prev_state != sample:
                total_changes += 1
                average_sit.append(current_sit_streak)
                current_sit_streak = 0

        # add on any leftover sits or stands
        if current_sit_streak != 0:
            average_sit.append(current_sit_streak)
        else:
            average_stand.append(current_stand_streak)

        # can't have empty lists
        if not average_sit:
            average_sit.append(0)
        if not average_stand:
            average_stand.append(0)
        prev_state = sample

    temp.append(total_sit)
    temp.append(total_stand)
    temp.append(total_changes)
    temp.append(mean)
    temp.append(mode)
    temp.append(longest_sit)
    temp.append(longest_stand)
    temp.append(np.mean(average_sit))
```

```
        temp.append(np.mean(average_stand))
        temp.append(np.std(average_sit))
        temp.append(np.std(average_stand))

        X.append(temp.copy())

    return X
```

[ ]:
```
sheep = []
for i in range(len(sheep_raw)):
    sheep.append(extractFeatures(sheep_raw[i]))

sheep_full = []
for i in range(len(sheep_raw_full)):
    sheep_full.append(extractFeatures(sheep_raw_full[i]))
```

[ ]:
```
print(sheep[0][0])
print(sheep[0][1])

print(sheep_full[0][0])
print(sheep_full[0][1])
```

```
[1999, 881, 118, 0.3059027777777778, 0, 175, 116, 62.21137026239067,
22.103081827842722, 42.75723646511831, 24.541816253464393]
[1885, 995, 114, 0.3454861111111111, 0, 198, 116, 58.02368692070031,
25.64577397910731, 43.31277351234201, 26.867492965424038]
[1999, 881, 118, 0.3059027777777778, 0, 175, 116, 62.21137026239067,
22.103081827842722, 42.75723646511831, 24.541816253464393]
[1885, 995, 114, 0.3454861111111111, 0, 198, 116, 58.02368692070031,
25.64577397910731, 43.31277351234201, 26.867492965424038]
```

1.0.4 Create X and Y to prepare model

[ ]:
```
X = []
X_RAW = []
y = []

# Sheep have 5 levels of illness. We group these into 3 levels:
#   1 = not sick
#   2,3 = moderately sick
#   4,5 = very sick
# Sheep A, B, U, V, W, and D have sickness levels of 4, 3, 5, 1, 2, and 5␣
↪respectively
SHEEP_ILLNESS_LEVEL = [2, 1, 2, 0, 1, 2]
for s, sr, ill in zip(sheep, sheep_raw, SHEEP_ILLNESS_LEVEL):
    for window, window_raw in zip(s, sr):
        X.append(window)
```

```python
        X_RAW.append(window_raw)
        y.append(ill)

X_FULL = []
X_RAW_FULL = []
y_full = []
# FULL_SHEEP_ILLNESS_LEVEL = [4, 3, 1, 5, 2, 1, 3, 5, 4, 1, 4, 4, 2, 2, 3, 3,␣
 ↪5, 3, 4, 1, 5, 1, 2]
FULL_SHEEP_ILLNESS_LEVEL = [2, 1, 0, 2, 1, 0, 1, 2, 2, 0, 2, 2, 1, 1, 1, 1, 2,␣
 ↪1, 2, 0, 2, 0, 1]
for s, sr, ill in zip(sheep_full, sheep_raw_full, FULL_SHEEP_ILLNESS_LEVEL):
    for window, window_raw in zip(s, sr):
        X_FULL.append(window)
        X_RAW_FULL.append(window_raw)
        y_full.append(ill)

y_full_binary = []
# FULL_SHEEP_BINARY_ILLNESS_LEVEL = [4, 3, 1, 5, 2, 1, 3, 5, 4, 1, 4, 4, 2, 2,␣
 ↪3, 3, 5, 3, 4, 1, 5, 1, 2]
FULL_SHEEP_BINARY_ILLNESS_LEVEL = [1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1,␣
 ↪1, 1, 1, 1, 0, 1, 0, 0]
for s, ill in zip(sheep_full, FULL_SHEEP_BINARY_ILLNESS_LEVEL):
    for window in s:
        y_full_binary.append(ill)

# normalize X
sc = StandardScaler()
X = sc.fit_transform(X)
sc = StandardScaler()
X_FULL = sc.fit_transform(X_FULL)
```

1.0.5 Model Creation

[ ]:
```python
# Train and evaluate the model
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import numpy as np

def train_and_evaluate_model(X, y, model, binary):
    # Do the K-fold cross validation
    cv = StratifiedKFold(n_splits=10)
    scores = cross_val_score(model, X, y, cv=cv, scoring='accuracy')

    # This function sets an empty array of size len(y)
```

```
    # It then does the predictions for each fold (retraining the model each␣
↪time)
    # and fills in the prediction in the correct spot in the array. By the end␣
↪of it,
    # you will have your y array and a new array (predictions) of guesses for␣
↪each sample
    # this is because Kfold uses each sample once for test data.
    predictions = cross_val_predict(model, X, y, cv=cv)

    # we can use this to create a confusion matrix
    if binary == 0:
        new_labels = ['Not Sick', 'Moderately Sick', 'Very Sick']
        old_labels = [0, 1, 2]
    else:
        new_labels = ['Healthy', 'Sick']
        old_labels = [0, 1]

    cm = confusion_matrix(y, predictions)
    plt.matshow(cm, cmap=plt.cm.Blues)
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            plt.text(x=j, y=i,s=cm[i, j], va='center', ha='center',␣
↪size='xx-large')
    plt.xticks(ticks=old_labels, labels=new_labels)
    plt.yticks(ticks=old_labels, labels=new_labels)
    plt.xlabel('Predicted Labels')
    plt.ylabel('True Label')

    print(f'All accuracy scores from 10 folds:')
    for score in scores:
        print(f'{round(score * 100, 2)}%', end=' ')
    print()
    print(f'Mean Accuracy from 10 Fold Cross-Validation: {round(np.
↪mean(scores)*100, 2)}%\n')
```

1.0.6 Testing Models

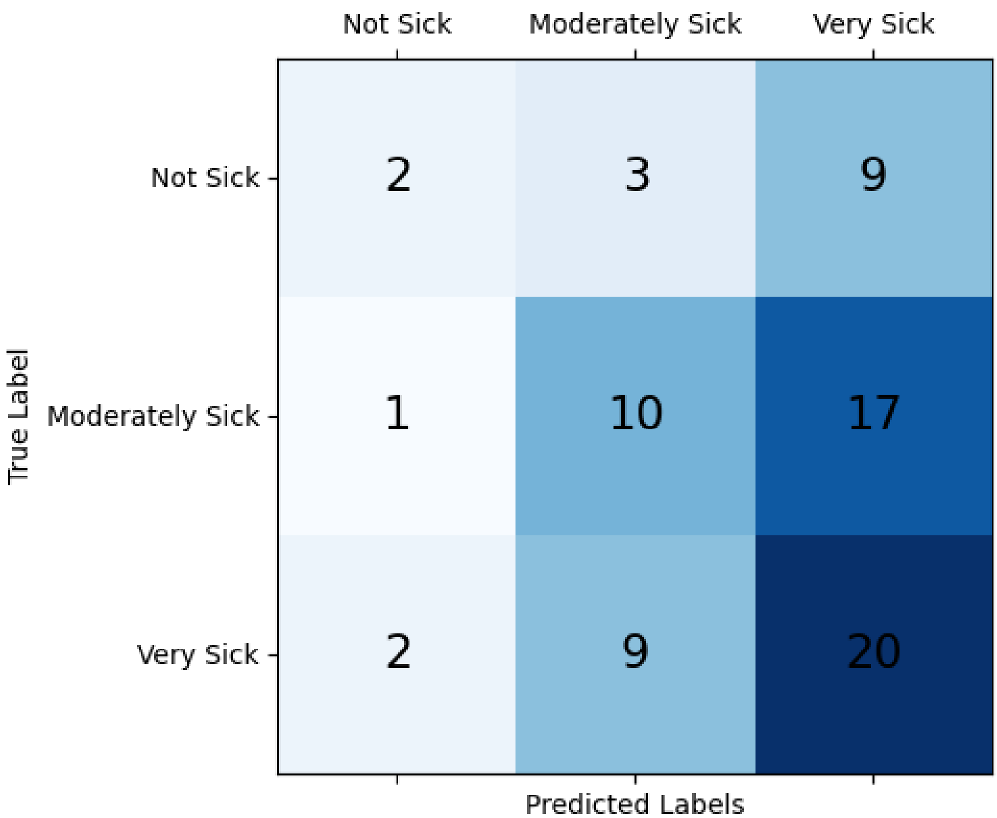
[ ]:
```
from sklearn.ensemble import RandomForestClassifier

print('USING RAW DATA')
model = RandomForestClassifier(n_estimators=100, random_state=42)
train_and_evaluate_model(X_RAW, y, model, 0)

print('')
print('USING FEATURE ENGINEERED DATA')
model = RandomForestClassifier(n_estimators=100, random_state=42)
```

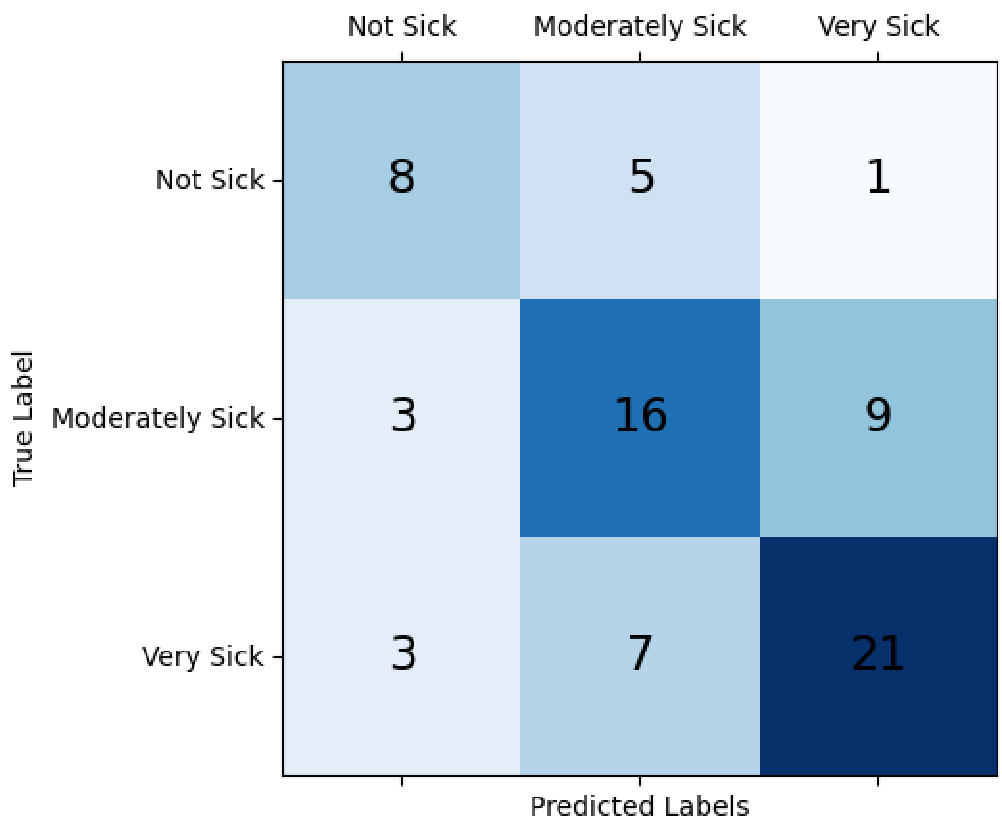
```python
train_and_evaluate_model(X, y, model, 0)

print('********************')
print('USING RAW DATA WITH ALL SHEEP')
model = RandomForestClassifier(n_estimators=100, random_state=42)
train_and_evaluate_model(X_RAW_FULL, y_full, model, 0)

print('')
print('USING FEATURE ENGINEERED DATA WITH ALL SHEEP')
model = RandomForestClassifier(n_estimators=100, random_state=42)
train_and_evaluate_model(X_FULL, y_full, model, 0)

print('********************')
print('USING RAW DATA WITH ALL SHEEP WITH BINARY OUTPUT')
model = RandomForestClassifier(n_estimators=100, random_state=42)
train_and_evaluate_model(X_RAW_FULL, y_full_binary, model, 1)

print('')
print('USING FEATURE ENGINEERED DATA WITH ALL SHEEP WITH BINARY OUTPUT')
model = RandomForestClassifier(n_estimators=100, random_state=42)
train_and_evaluate_model(X_FULL, y_full_binary, model, 1)
```

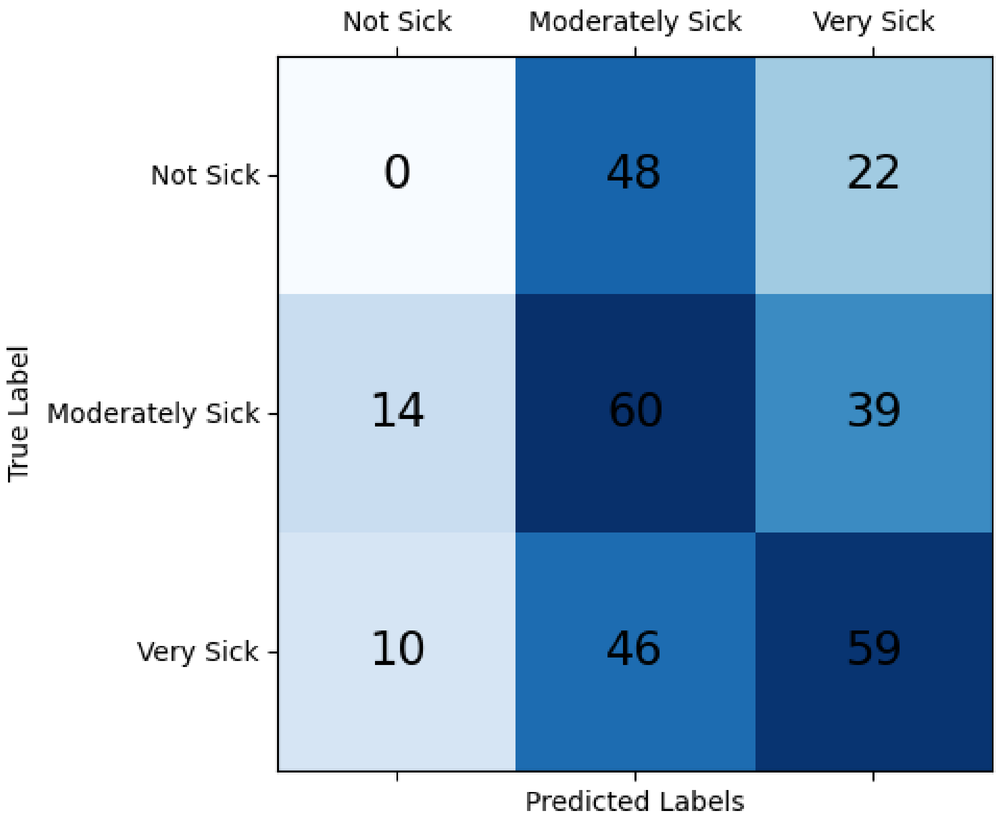
```
USING RAW DATA
All accuracy scores from 10 folds:
50.0% 37.5% 50.0% 71.43% 42.86% 42.86% 42.86% 57.14% 42.86% 0.0%
Mean Accuracy from 10 Fold Cross-Validation: 43.75%


USING FEATURE ENGINEERED DATA
All accuracy scores from 10 folds:
37.5% 12.5% 37.5% 71.43% 85.71% 71.43% 71.43% 71.43% 100.0% 71.43%
Mean Accuracy from 10 Fold Cross-Validation: 63.04%

********************
USING RAW DATA WITH ALL SHEEP
All accuracy scores from 10 folds:
40.0% 33.33% 33.33% 53.33% 36.67% 30.0% 46.67% 50.0% 41.38% 34.48%
Mean Accuracy from 10 Fold Cross-Validation: 39.92%


USING FEATURE ENGINEERED DATA WITH ALL SHEEP
All accuracy scores from 10 folds:
36.67% 46.67% 36.67% 36.67% 43.33% 60.0% 53.33% 40.0% 44.83% 37.93%
Mean Accuracy from 10 Fold Cross-Validation: 43.61%

********************
USING RAW DATA WITH ALL SHEEP WITH BINARY OUTPUT
```

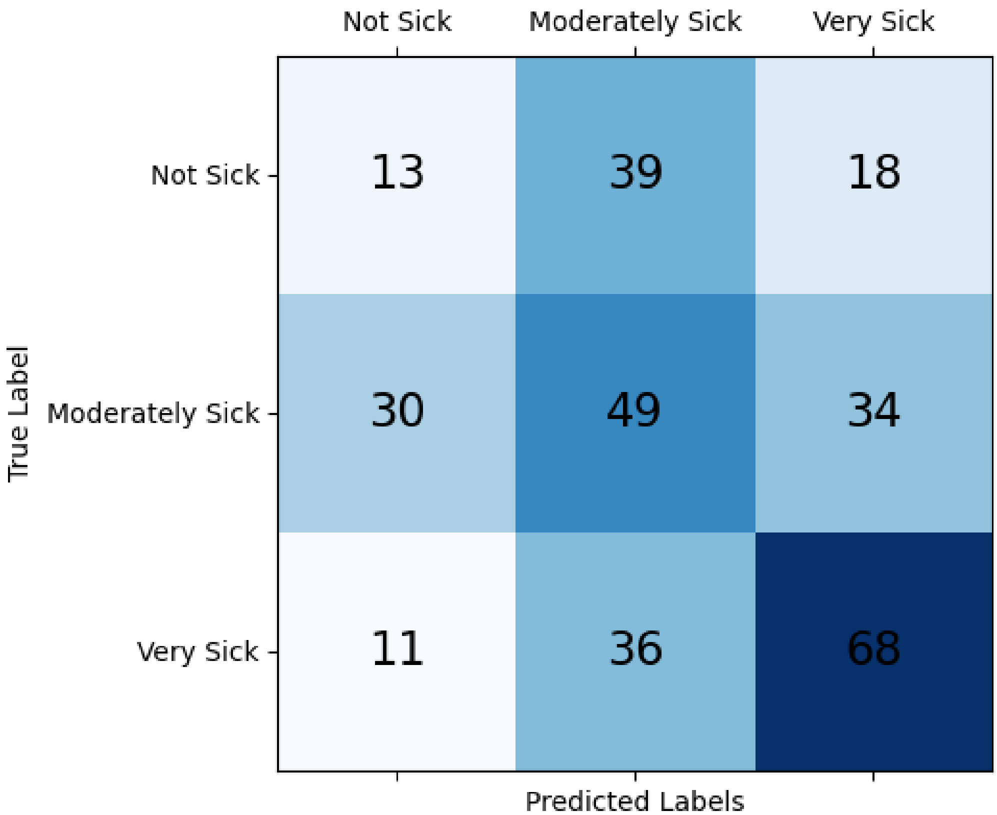
```
All accuracy scores from 10 folds:
60.0% 66.67% 50.0% 56.67% 63.33% 53.33% 73.33% 43.33% 51.72% 62.07%
Mean Accuracy from 10 Fold Cross-Validation: 58.05%
```

```
USING FEATURE ENGINEERED DATA WITH ALL SHEEP WITH BINARY OUTPUT
All accuracy scores from 10 folds:
70.0% 76.67% 76.67% 60.0% 40.0% 53.33% 66.67% 76.67% 48.28% 75.86%
Mean Accuracy from 10 Fold Cross-Validation: 64.41%
```

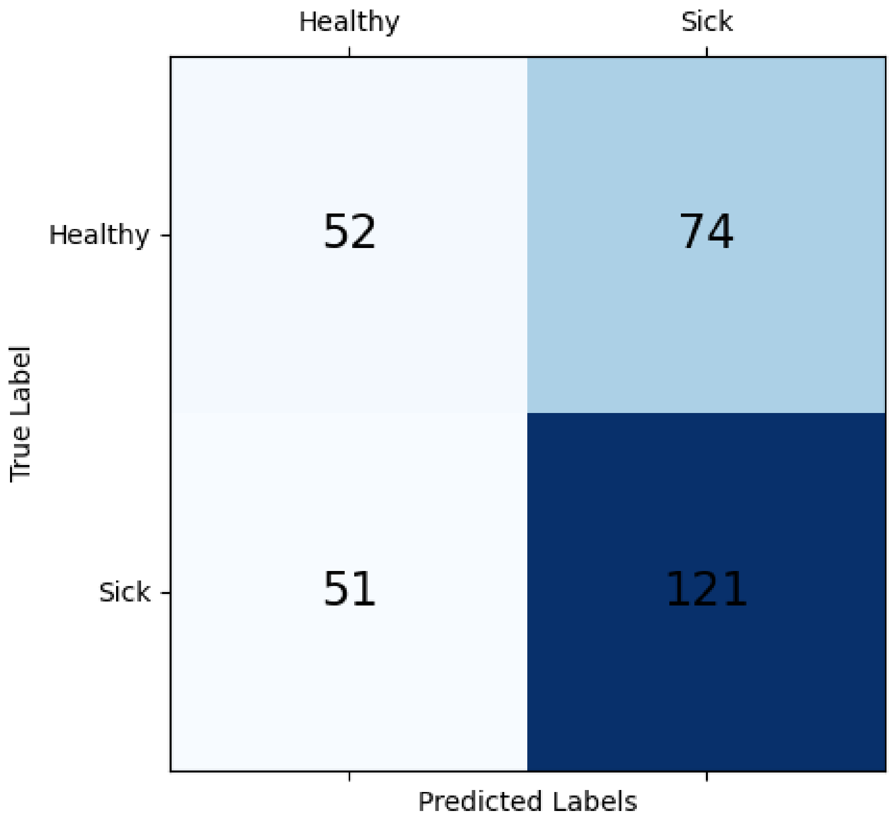


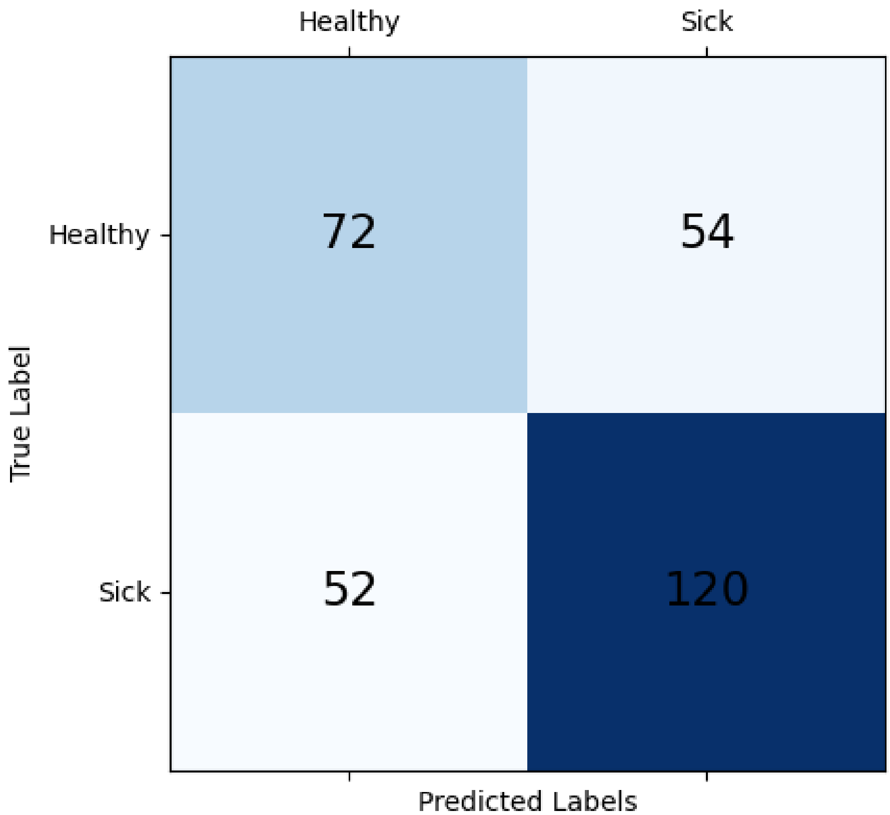

Not Sick
Moderately Sick
Very Sick
True Label
Not Sick
Moderately Sick
Very Sick
8
5
1
3
16
9
3
7
21
Predicted Labels


Not Sick
Moderately Sick
Very Sick
Not Sick
Moderately Sick
Very Sick
True Label
Predicted Labels
0
48
22
14
60
39
10
46
59



Not Sick
Moderately Sick
Very Sick
Not Sick
Moderately Sick
Very Sick
True Label
13
39
18
30
49
34
11
36
68
Predicted Labels

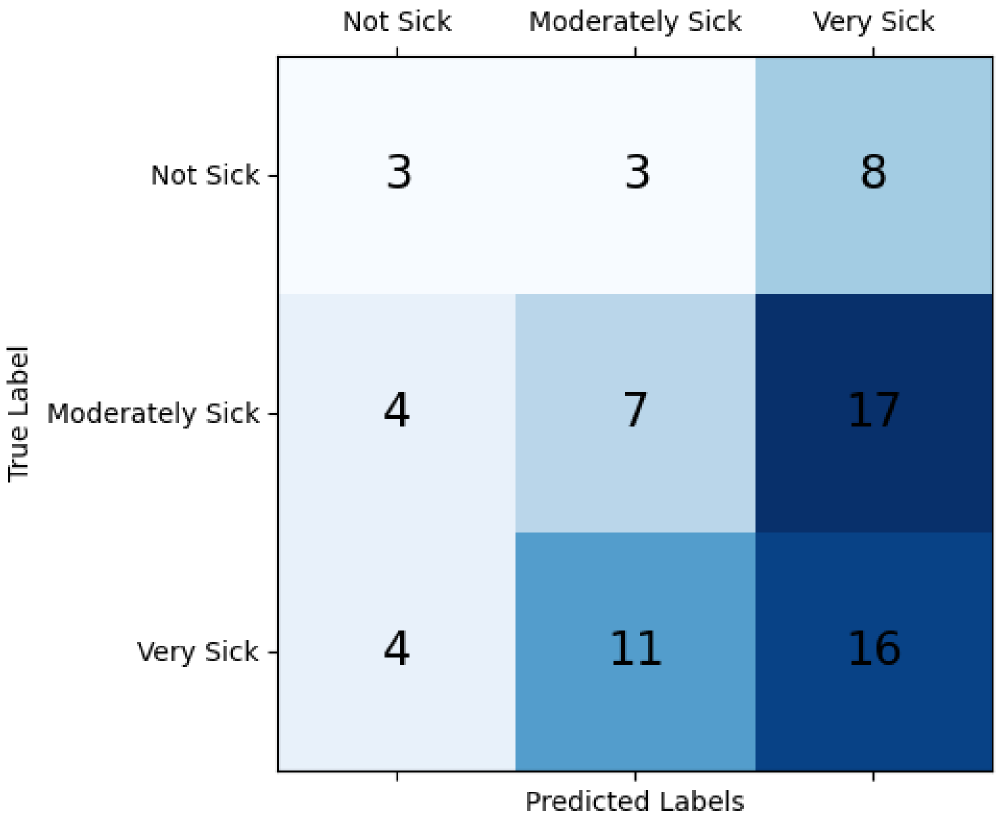

Healthy
Sick
Healthy
Sick
52
74
51
121
True Label
Predicted Labels

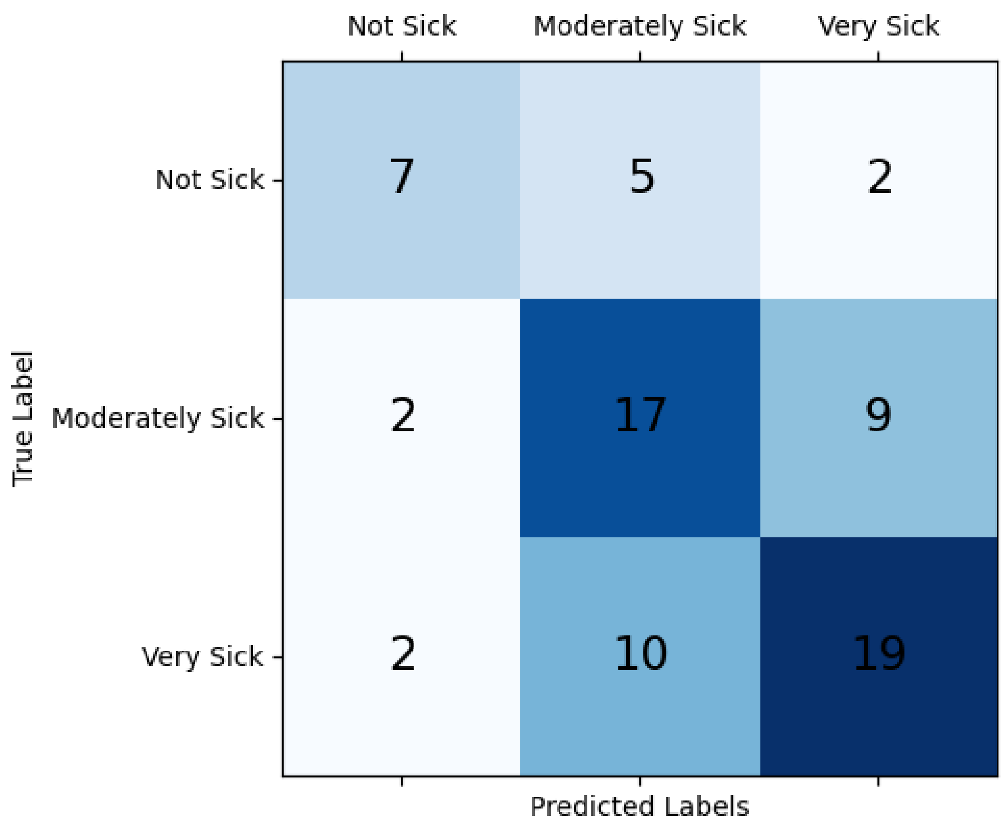


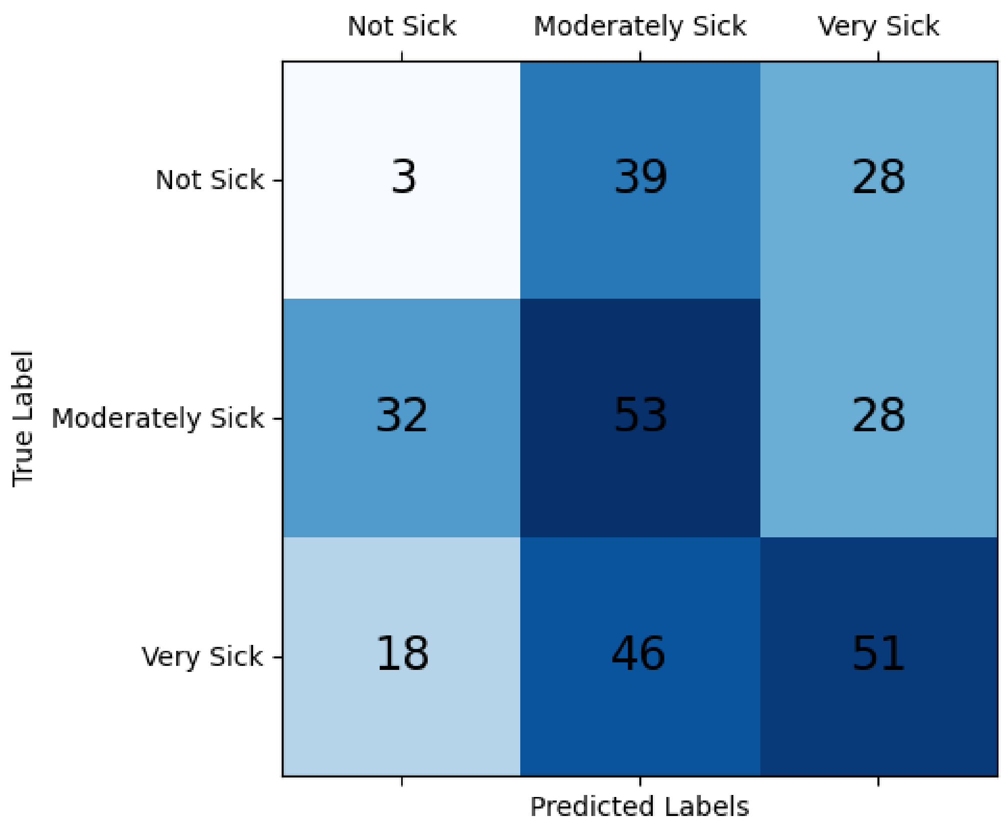
[ ]:
```
from xgboost import XGBClassifier

print('USING RAW DATA')
model = XGBClassifier()
train_and_evaluate_model(X_RAW, y, model, 0)

print('')
print('USING FEATURE ENGINEERED DATA')
model = XGBClassifier()
train_and_evaluate_model(X, y, model, 0)

print('********************')
print('USING RAW DATA WITH ALL SHEEP')
model = XGBClassifier()
train_and_evaluate_model(X_RAW_FULL, y_full, model, 0)

print('')
print('USING FEATURE ENGINEERED DATA WITH ALL SHEEP')
model = XGBClassifier()
train_and_evaluate_model(X_FULL, y_full, model, 0)
```

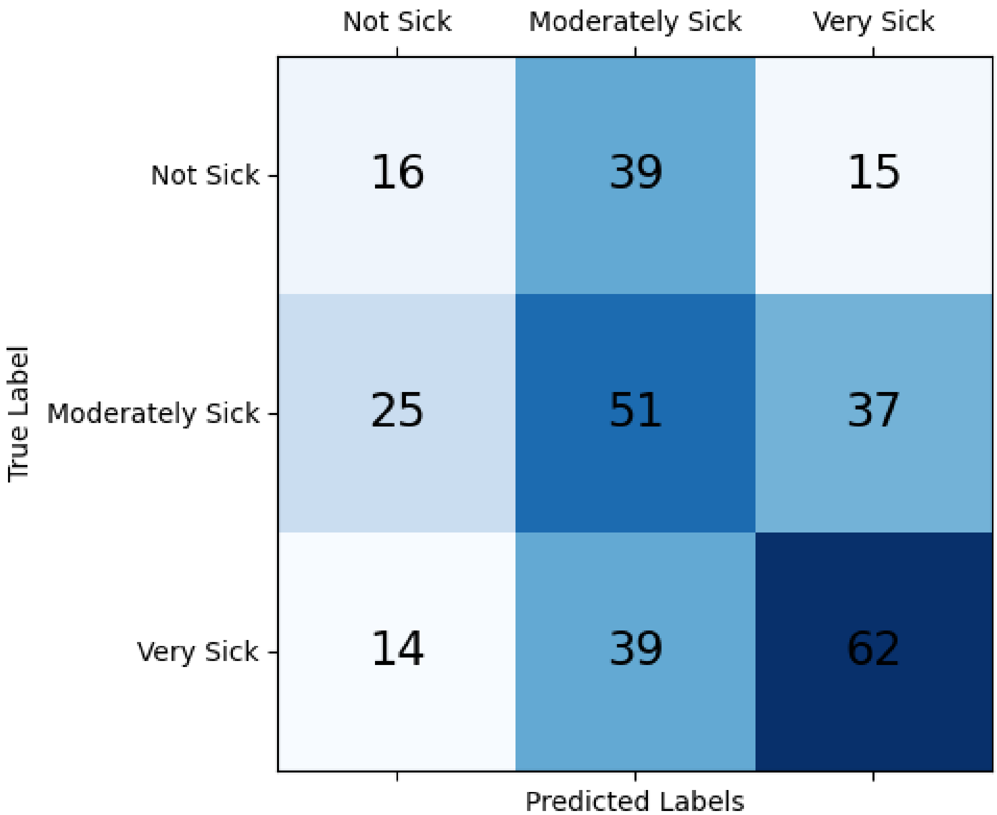
```python
print('*********************')
print('USING RAW DATA WITH ALL SHEEP WITH BINARY OUTPUT')
model = XGBClassifier()
train_and_evaluate_model(X_RAW_FULL, y_full_binary, model, 1)

print('')
print('USING FEATURE ENGINEERED DATA WITH ALL SHEEP WITH BINARY OUTPUT')
model = XGBClassifier()
train_and_evaluate_model(X_FULL, y_full_binary, model, 1)
```

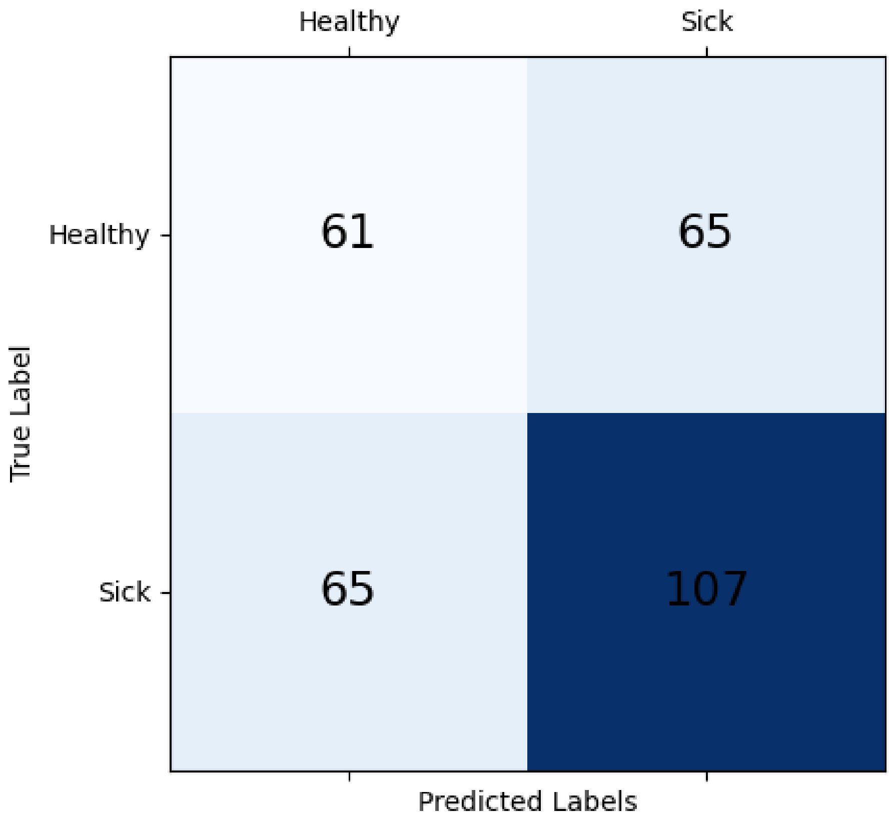
```
USING RAW DATA
All accuracy scores from 10 folds:
25.0% 50.0% 50.0% 28.57% 14.29% 42.86% 28.57% 42.86% 42.86% 28.57%
Mean Accuracy from 10 Fold Cross-Validation: 35.36%


USING FEATURE ENGINEERED DATA
All accuracy scores from 10 folds:
50.0% 12.5% 37.5% 28.57% 85.71% 85.71% 85.71% 57.14% 85.71% 71.43%
Mean Accuracy from 10 Fold Cross-Validation: 60.0%

*********************
USING RAW DATA WITH ALL SHEEP
All accuracy scores from 10 folds:
50.0% 30.0% 30.0% 30.0% 50.0% 30.0% 50.0% 26.67% 37.93% 24.14%
Mean Accuracy from 10 Fold Cross-Validation: 35.87%


USING FEATURE ENGINEERED DATA WITH ALL SHEEP
All accuracy scores from 10 folds:
30.0% 43.33% 40.0% 46.67% 40.0% 70.0% 50.0% 33.33% 51.72% 27.59%
Mean Accuracy from 10 Fold Cross-Validation: 43.26%

*********************
USING RAW DATA WITH ALL SHEEP WITH BINARY OUTPUT
All accuracy scores from 10 folds:
50.0% 53.33% 50.0% 60.0% 63.33% 53.33% 70.0% 60.0% 48.28% 55.17%
Mean Accuracy from 10 Fold Cross-Validation: 56.34%


USING FEATURE ENGINEERED DATA WITH ALL SHEEP WITH BINARY OUTPUT
All accuracy scores from 10 folds:
63.33% 60.0% 70.0% 53.33% 43.33% 50.0% 70.0% 70.0% 51.72% 75.86%
Mean Accuracy from 10 Fold Cross-Validation: 60.76%
```

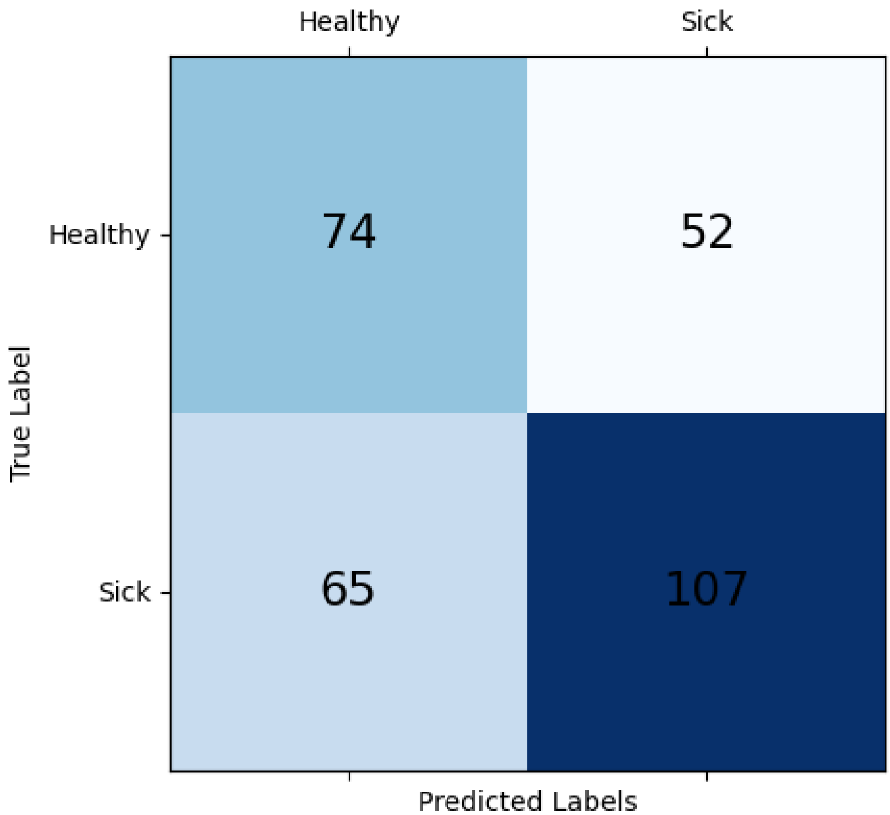

Not Sick
Moderately Sick
Very Sick
Not Sick
Moderately Sick
Very Sick
True Label
Predicted Labels
3
3
8
4
7
17
4
11
16


Not Sick
Moderately Sick
Very Sick
True Label
Not Sick
Moderately Sick
Very Sick
7
5
2
2
17
9
2
10
19
Predicted Labels


Not Sick
Moderately Sick
Very Sick
Not Sick
Moderately Sick
Very Sick
True Label
3
39
28
32
53
28
18
46
51
Predicted Labels


Not Sick
Moderately Sick
Very Sick
True Label
Not Sick
Moderately Sick
Very Sick
16
39
15
25
51
37
14
39
62
Predicted Labels

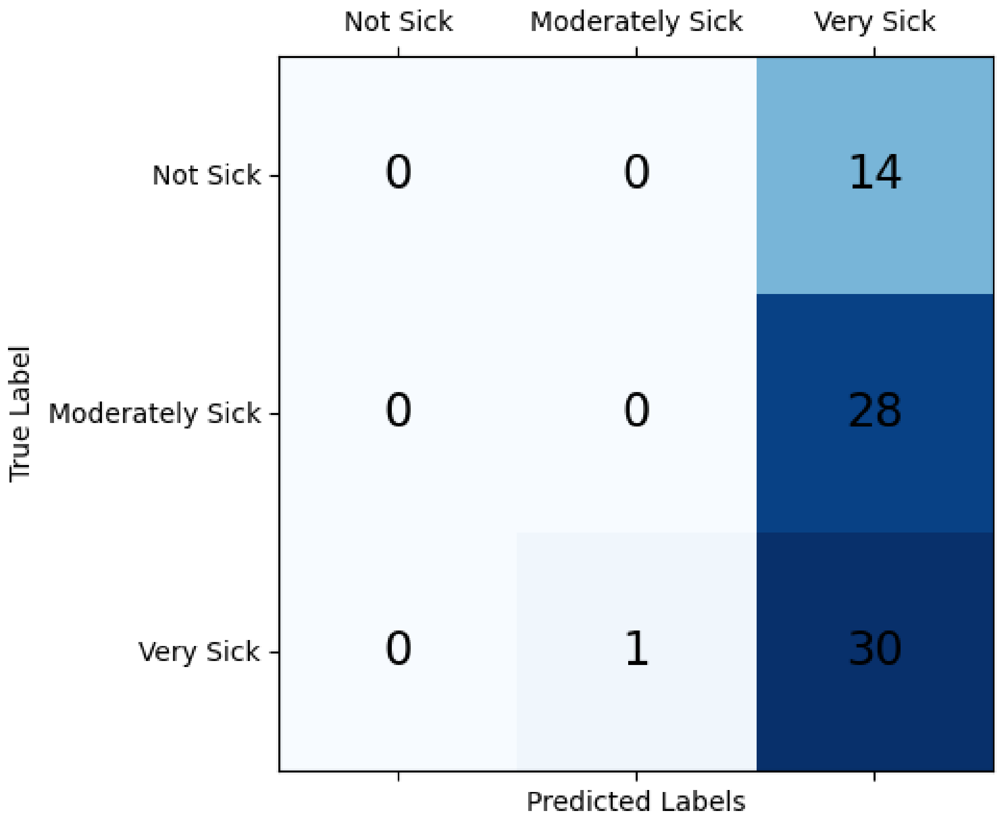

Healthy
Sick
Healthy
Sick
61
65
65
107
True Label
Predicted Labels

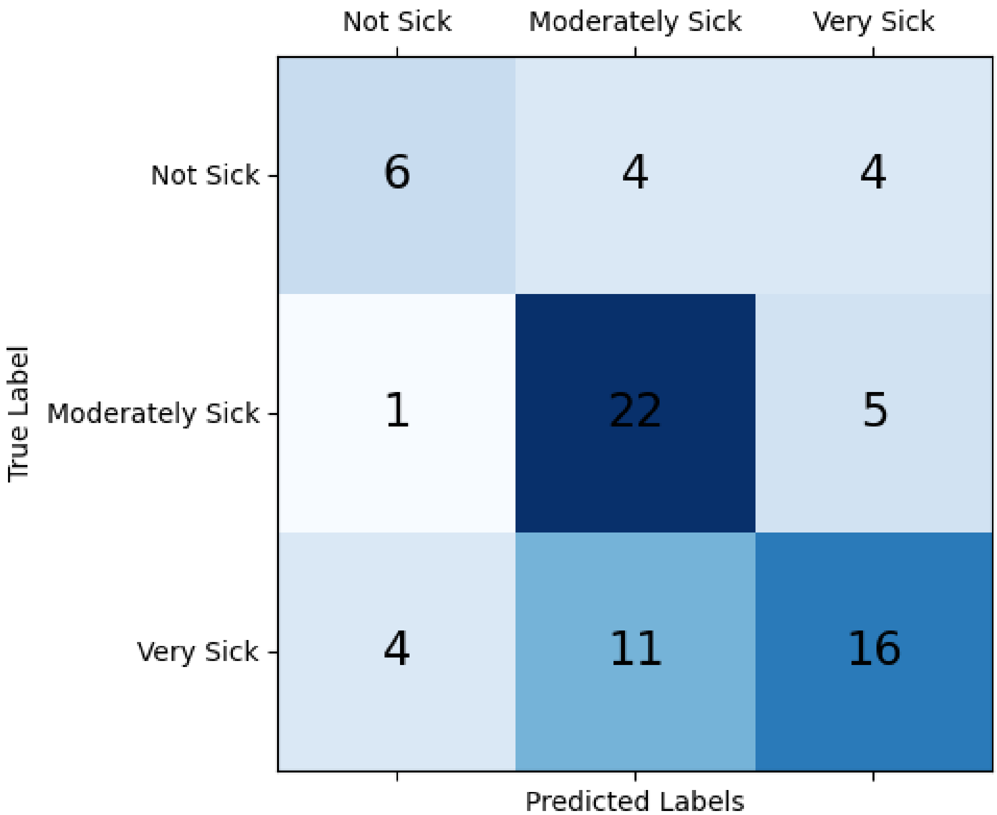


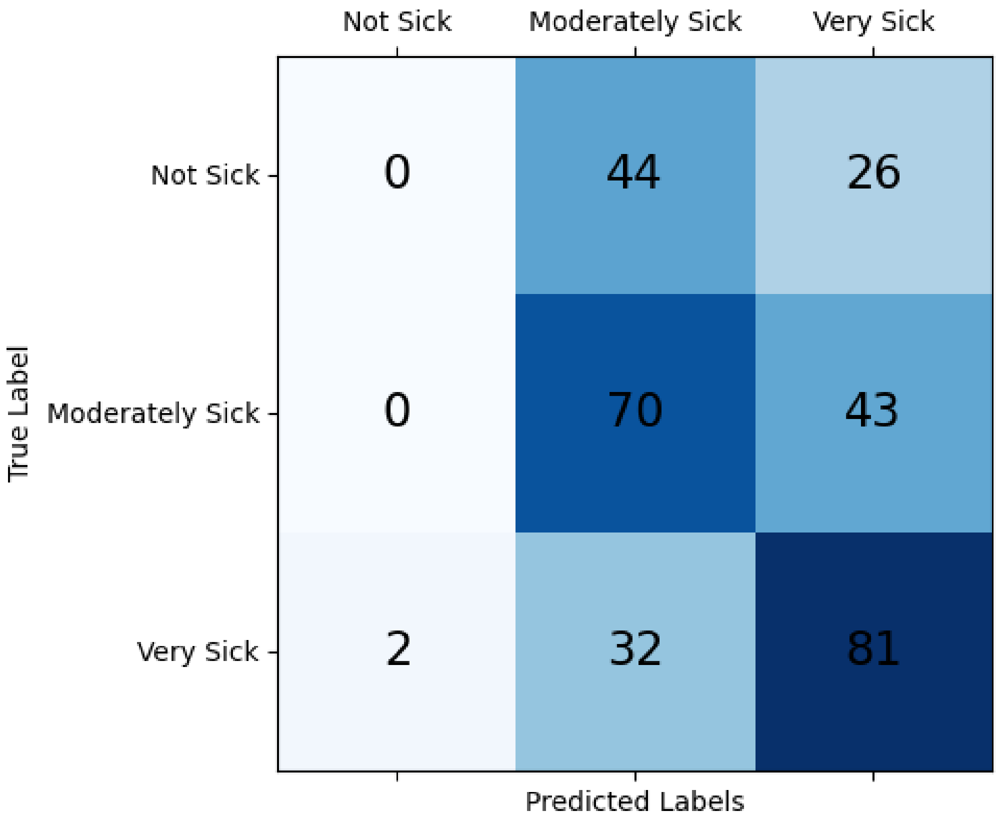
[ ]:
```
from sklearn.svm import SVC

print('USING RAW DATA')
model = SVC()
train_and_evaluate_model(X_RAW, y, model, 0)

print('')
print('USING FEATURE ENGINEERED DATA')
model = SVC()
train_and_evaluate_model(X, y, model, 0)

print('********************')
print('USING RAW DATA WITH ALL SHEEP')
model = SVC()
train_and_evaluate_model(X_RAW_FULL, y_full, model, 0)

print('')
print('USING FEATURE ENGINEERED DATA WITH ALL SHEEP')
model = SVC()
train_and_evaluate_model(X_FULL, y_full, model, 0)
```

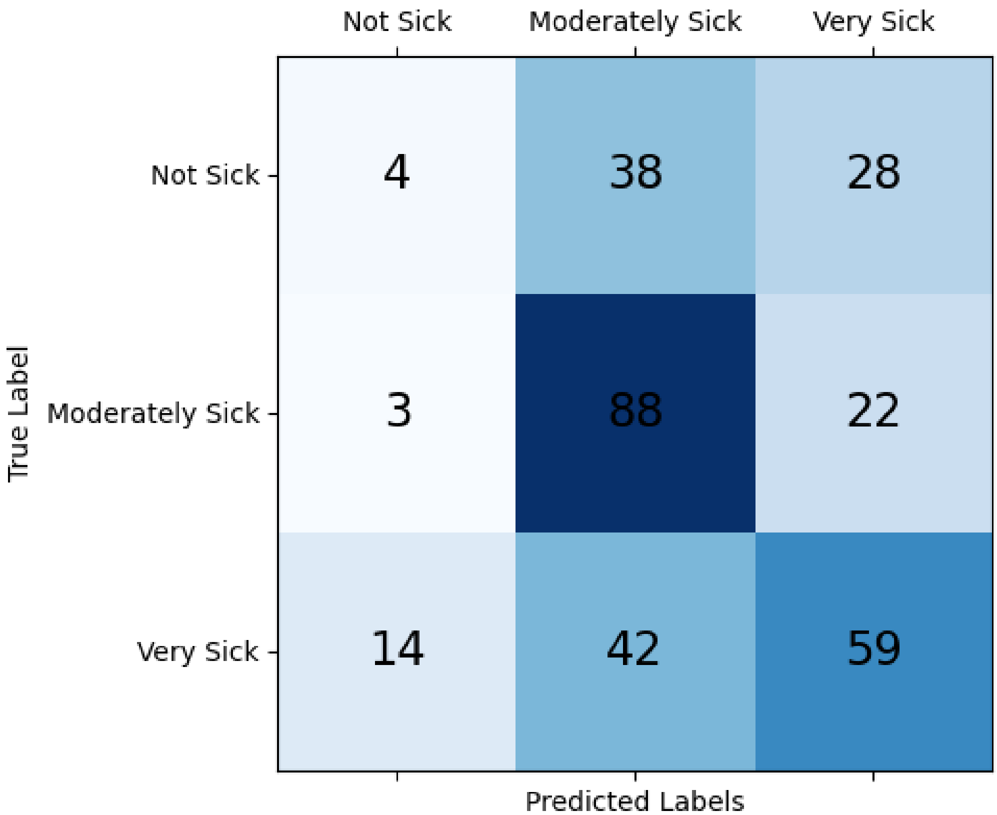
```python
print('*********************')
print('USING RAW DATA WITH ALL SHEEP WITH BINARY OUTPUT')
model = SVC()
train_and_evaluate_model(X_RAW_FULL, y_full_binary, model, 1)

print('')
print('USING FEATURE ENGINEERED DATA WITH ALL SHEEP WITH BINARY OUTPUT')
model = SVC()
train_and_evaluate_model(X_FULL, y_full_binary, model, 1)
```

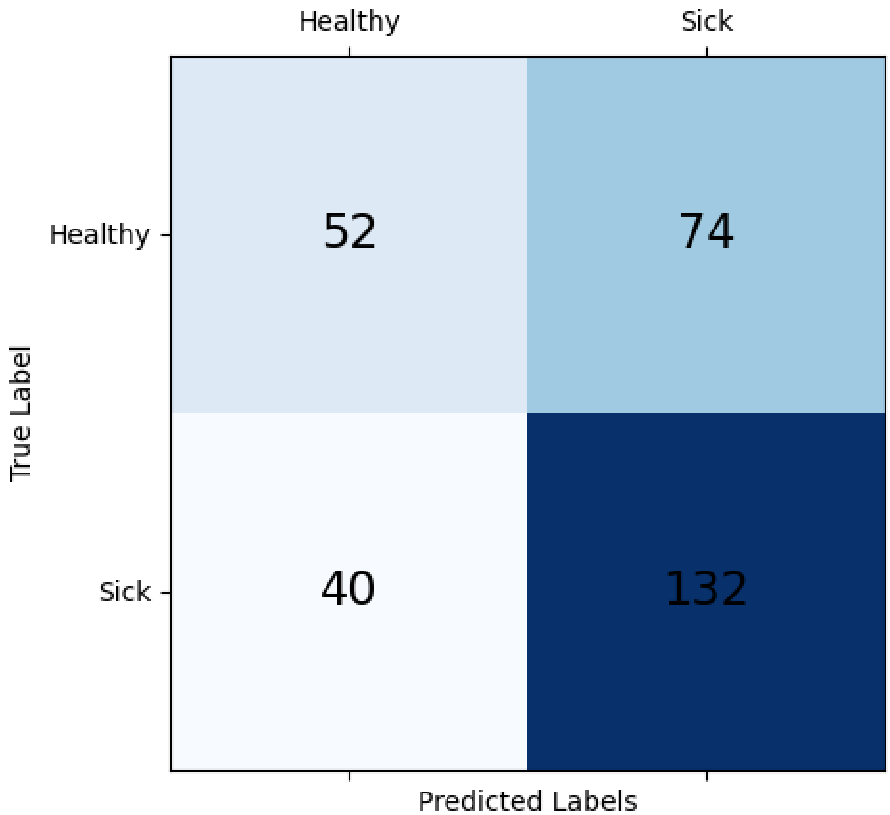
```
USING RAW DATA
All accuracy scores from 10 folds:
37.5% 37.5% 37.5% 42.86% 42.86% 42.86% 42.86% 42.86% 42.86% 42.86%
Mean Accuracy from 10 Fold Cross-Validation: 41.25%


USING FEATURE ENGINEERED DATA
All accuracy scores from 10 folds:
50.0% 25.0% 37.5% 85.71% 57.14% 57.14% 57.14% 71.43% 100.0% 71.43%
Mean Accuracy from 10 Fold Cross-Validation: 61.25%

*********************
USING RAW DATA WITH ALL SHEEP
All accuracy scores from 10 folds:
46.67% 46.67% 43.33% 56.67% 66.67% 40.0% 56.67% 53.33% 51.72% 44.83%
Mean Accuracy from 10 Fold Cross-Validation: 50.66%


USING FEATURE ENGINEERED DATA WITH ALL SHEEP
All accuracy scores from 10 folds:
36.67% 50.0% 36.67% 50.0% 63.33% 43.33% 63.33% 56.67% 65.52% 41.38%
Mean Accuracy from 10 Fold Cross-Validation: 50.69%

*********************
USING RAW DATA WITH ALL SHEEP WITH BINARY OUTPUT
All accuracy scores from 10 folds:
63.33% 76.67% 50.0% 60.0% 63.33% 56.67% 80.0% 60.0% 48.28% 58.62%
Mean Accuracy from 10 Fold Cross-Validation: 61.69%


USING FEATURE ENGINEERED DATA WITH ALL SHEEP WITH BINARY OUTPUT
All accuracy scores from 10 folds:
60.0% 76.67% 63.33% 50.0% 50.0% 46.67% 80.0% 76.67% 48.28% 79.31%
Mean Accuracy from 10 Fold Cross-Validation: 63.09%
```

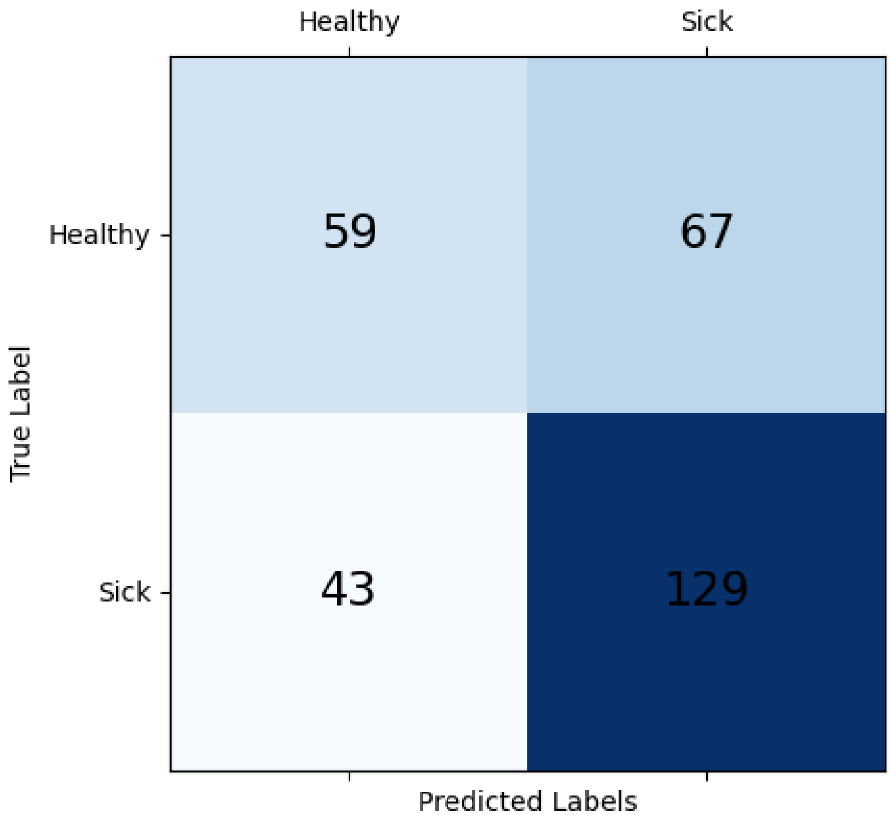

Not Sick
Moderately Sick
Very Sick
True Label
Not Sick
Moderately Sick
Very Sick
0
0
14
0
0
28
0
1
30
Predicted Labels


Not Sick
Moderately Sick
Very Sick
True Label
Not Sick
Moderately Sick
Very Sick
6
4
4
1
22
5
4
11
16
Predicted Labels


Not Sick
Moderately Sick
Very Sick
True Label
Not Sick
Moderately Sick
Very Sick
0
44
26
0
70
43
2
32
81
Predicted Labels


Not Sick
Moderately Sick
Very Sick
True Label
Not Sick
Moderately Sick
Very Sick
4
38
28
3
88
22
14
42
59
Predicted Labels

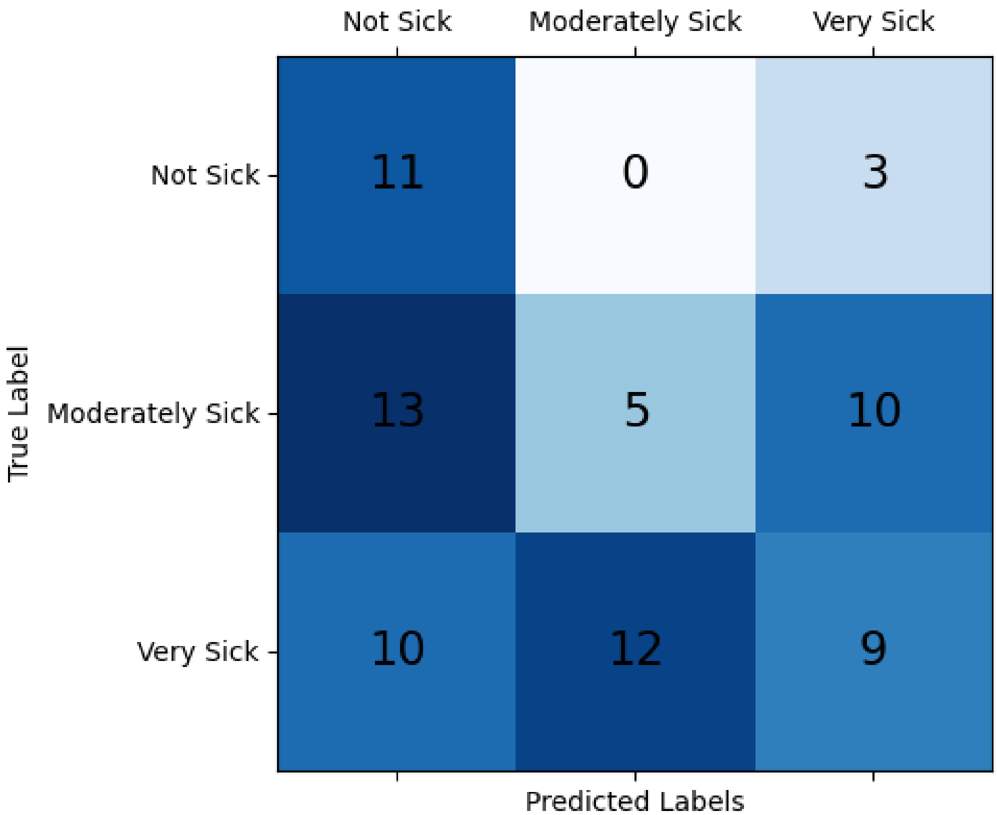

Healthy
Sick
Healthy
Sick
True Label
52
74
40
132
Predicted Labels

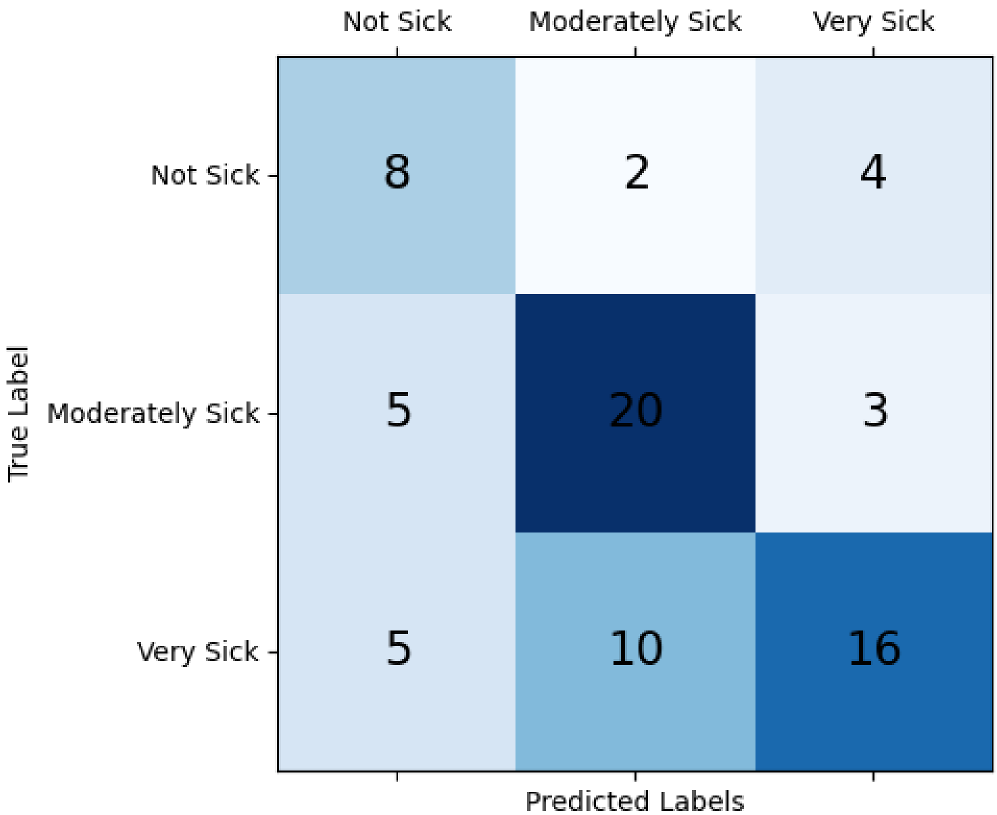


[ ]:
```
from sklearn.neighbors import KNeighborsClassifier

print('USING RAW DATA')
model = KNeighborsClassifier()
train_and_evaluate_model(X_RAW, y, model, 0)

print('')
print('USING FEATURE ENGINEERED DATA')
model = KNeighborsClassifier()
train_and_evaluate_model(X, y, model, 0)

print('*********************')
print('USING RAW DATA WITH ALL SHEEP')
model = KNeighborsClassifier()
train_and_evaluate_model(X_RAW_FULL, y_full, model, 0)

print('')
print('USING FEATURE ENGINEERED DATA WITH ALL SHEEP')
model = KNeighborsClassifier()
train_and_evaluate_model(X_FULL, y_full, model, 0)
```

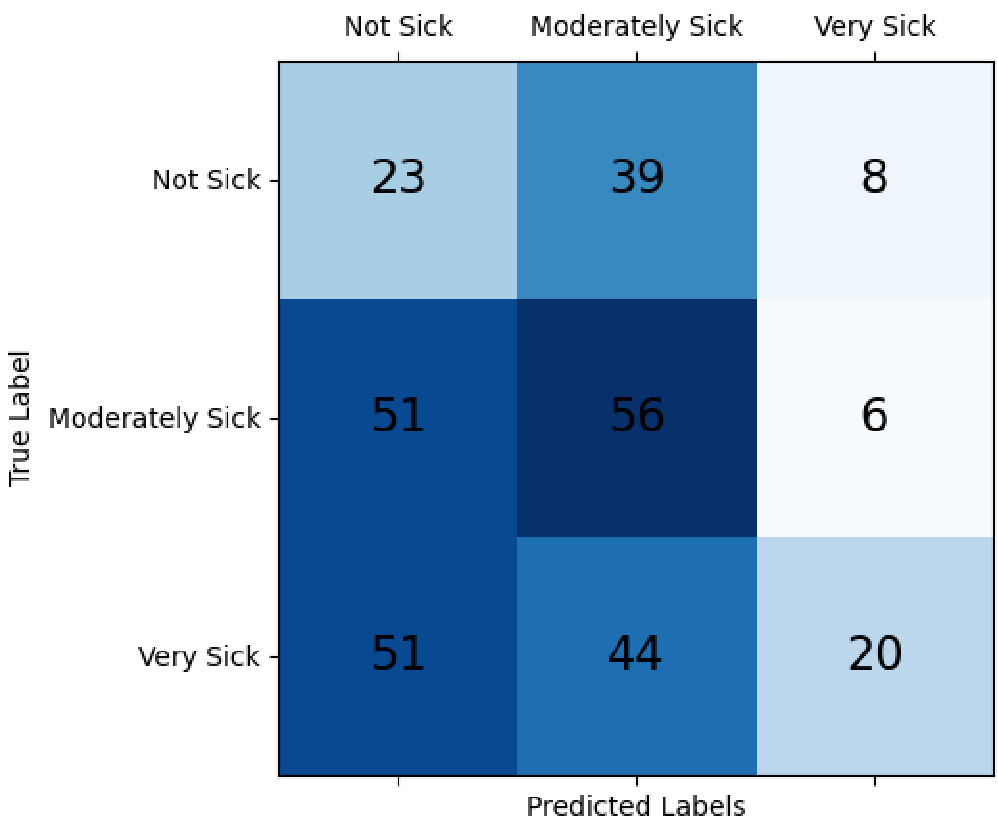
```python
print('********************')
print('USING RAW DATA WITH ALL SHEEP WITH BINARY OUTPUT')
model = KNeighborsClassifier()
train_and_evaluate_model(X_RAW_FULL, y_full_binary, model, 1)

print('')
print('USING FEATURE ENGINEERED DATA WITH ALL SHEEP WITH BINARY OUTPUT')
model = KNeighborsClassifier()
train_and_evaluate_model(X_FULL, y_full_binary, model, 1)
```

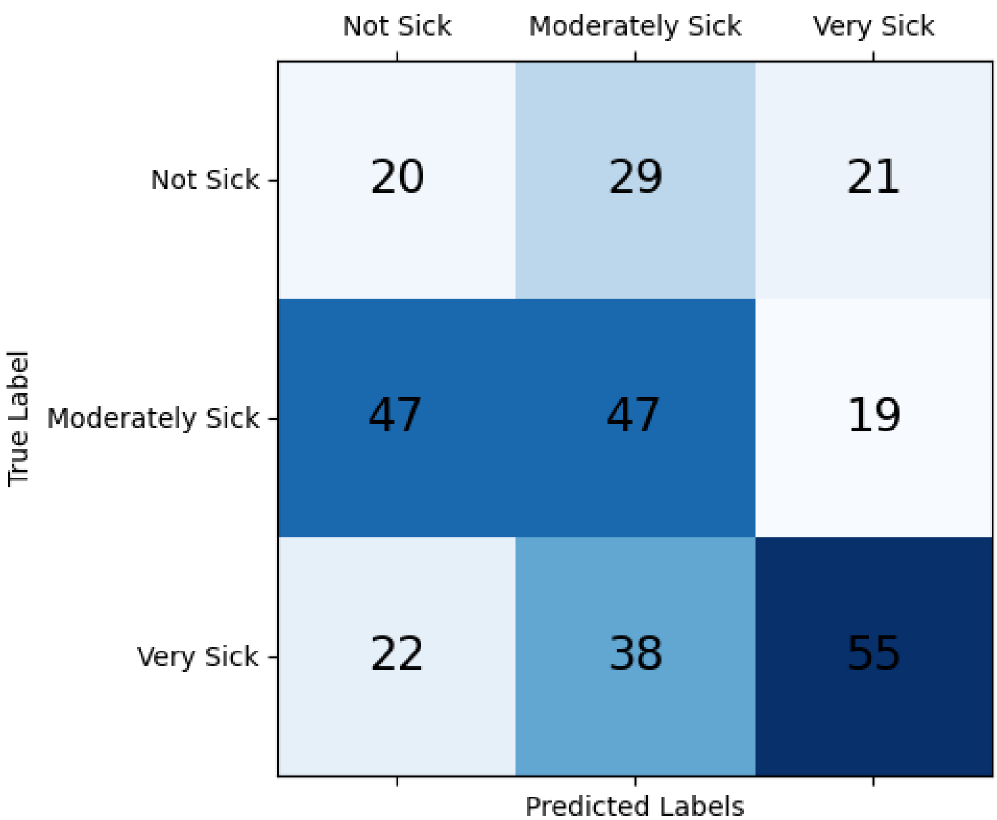
```
USING RAW DATA
All accuracy scores from 10 folds:
50.0% 37.5% 75.0% 57.14% 14.29% 28.57% 14.29% 28.57% 0.0% 28.57%
Mean Accuracy from 10 Fold Cross-Validation: 33.39%


USING FEATURE ENGINEERED DATA
All accuracy scores from 10 folds:
50.0% 25.0% 50.0% 57.14% 57.14% 57.14% 57.14% 71.43% 100.0% 85.71%
Mean Accuracy from 10 Fold Cross-Validation: 61.07%

********************
USING RAW DATA WITH ALL SHEEP
All accuracy scores from 10 folds:
16.67% 33.33% 40.0% 26.67% 36.67% 33.33% 43.33% 30.0% 41.38% 31.03%
Mean Accuracy from 10 Fold Cross-Validation: 33.24%


USING FEATURE ENGINEERED DATA WITH ALL SHEEP
All accuracy scores from 10 folds:
23.33% 40.0% 43.33% 60.0% 50.0% 30.0% 46.67% 43.33% 41.38% 31.03%
Mean Accuracy from 10 Fold Cross-Validation: 40.91%

********************
USING RAW DATA WITH ALL SHEEP WITH BINARY OUTPUT
All accuracy scores from 10 folds:
50.0% 50.0% 43.33% 46.67% 56.67% 53.33% 43.33% 26.67% 24.14% 65.52%
Mean Accuracy from 10 Fold Cross-Validation: 45.97%


USING FEATURE ENGINEERED DATA WITH ALL SHEEP WITH BINARY OUTPUT
All accuracy scores from 10 folds:
70.0% 60.0% 70.0% 50.0% 56.67% 40.0% 60.0% 60.0% 44.83% 65.52%
Mean Accuracy from 10 Fold Cross-Validation: 57.7%
```

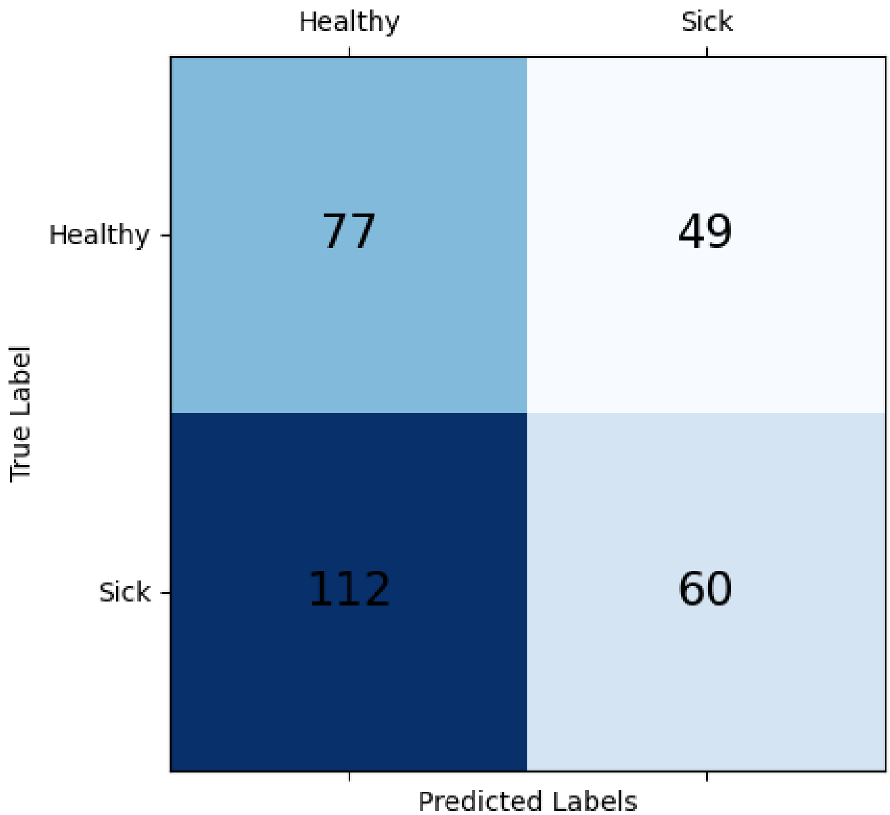

Not Sick
Moderately Sick
Very Sick
True Label
Predicted Labels
11
0
3
13
5
10
10
12
9

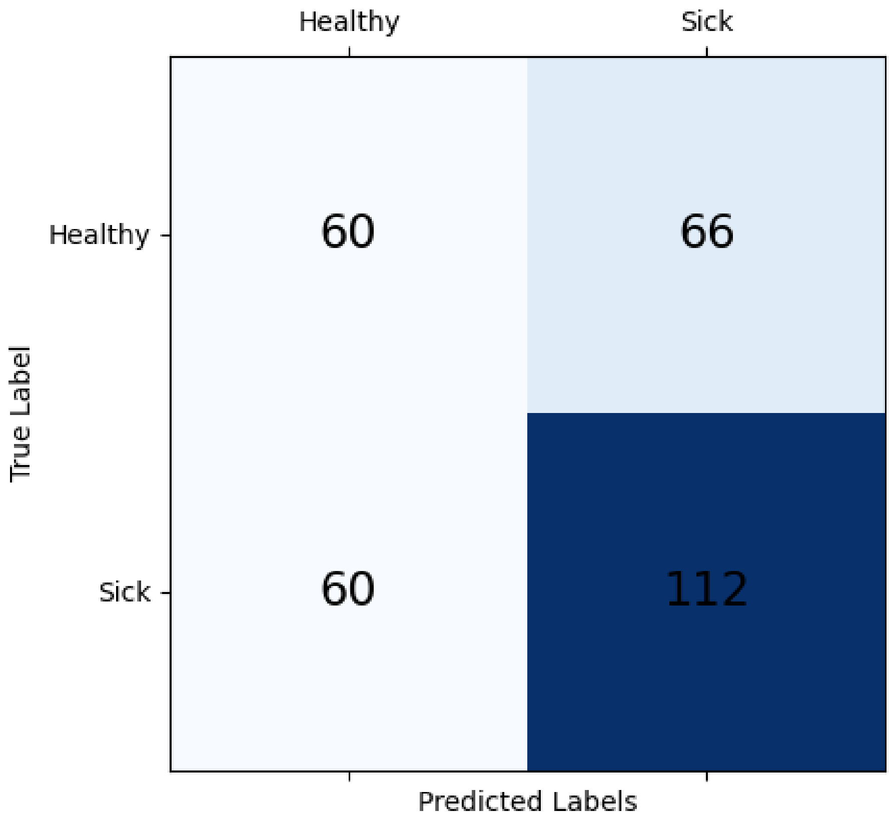


Not Sick
Moderately Sick
Very Sick
True Label
Not Sick
Moderately Sick
Very Sick
8
2
4
5
20
3
5
10
16
Predicted Labels


Not Sick
Moderately Sick
Very Sick
Not Sick
Moderately Sick
Very Sick
True Label
23
39
8
51
56
6
51
44
20
Predicted Labels


Not Sick
Moderately Sick
Very Sick
Not Sick
Moderately Sick
Very Sick
True Label
Predicted Labels
20
29
21
47
47
19
22
38
55


Healthy
Sick
Healthy
Sick
True Label
77
49
112
60
Predicted Labels


Healthy
Sick
Healthy
Sick
60
66
60
112
True Label
Predicted Labels