Open in Colab

Disease Prediction

Setup and initialization

```
from google.colab import drive
drive.mount('/content/drive')
```

```
     Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mou
```

```
import tensorflow as tf

from tensorflow.keras import layers
from tensorflow.keras import regularizers
from sklearn.model_selection import train_test_split

print(tf.__version__)

# let's set the random seed to make the results reproducible
tf.random.set_seed(74)
```

```
     2.9.2
```

```
!pip install git+https://github.com/tensorflow/docs

import tensorflow_docs as tfdocs
import tensorflow_docs.modeling
import tensorflow_docs.plots
```

```
     Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
     Collecting git+https://github.com/tensorflow/docs
       Cloning https://github.com/tensorflow/docs to /tmp/pip-req-build-ttbhhumv
       Running command git clone -q https://github.com/tensorflow/docs /tmp/pip-req-build-tt
     Requirement already satisfied: astor in /usr/local/lib/python3.7/dist-packages (from te
     Requirement already satisfied: absl-py in /usr/local/lib/python3.7/dist-packages (from 
     Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-packages (from t
     Requirement already satisfied: nbformat in /usr/local/lib/python3.7/dist-packages (from
     Requirement already satisfied: protobuf<3.20,>=3.12.0 in /usr/local/lib/python3.7/dist-
     Requirement already satisfied: pyyaml in /usr/local/lib/python3.7/dist-packages (from t
     Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-packag
     Requirement already satisfied: traitlets>=5.1 in /usr/local/lib/python3.7/dist-packages
     Requirement already satisfied: jupyter-core in /usr/local/lib/python3.7/dist-packages (
     Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.7/dist-package
```

```
Requirement already satisfied: fastjsonschema in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: importlib-metadata>=3.6 in /usr/local/lib/python3.7/dist
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/python3.7/dis
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /usr/lo
Requirement already satisfied: importlib-resources>=1.4.0 in /usr/local/lib/python3.7/d
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.7/dist-packages
Building wheels for collected packages: tensorflow-docs
  Building wheel for tensorflow-docs (setup.py) ... done
  Created wheel for tensorflow-docs: filename=tensorflow_docs-0.0.0.dev0-py3-none-any.w
  Stored in directory: /tmp/pip-ephem-wheel-cache-82qymla4/wheels/cc/c4/d8/5341e93b6376
Successfully built tensorflow-docs
Installing collected packages: tensorflow-docs
Successfully installed tensorflow-docs-0.0.0.dev0
```

```
from  IPython import display
from matplotlib import pyplot as plt

import numpy as np

import pathlib
import shutil
import tempfile


# currentdir
import os

logdir = os.path.join(os.getcwd(), "tensorboard_logs")
shutil.rmtree(logdir, ignore_errors=True)
```

1. Dataset Preparation

```
import pandas as pd

disease_training = pd.read_csv('/content/drive/MyDrive/projects/oman-gulf-college-project/dat
disease_testing = pd.read_csv('/content/drive/MyDrive/projects/oman-gulf-college-project/data
disease_training.head()
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	jc
0	1	1	1	0	0	0	
1	0	1	1	0	0	0	
2	1	0	1	0	0	0	

3 Remove last column

```
disease_training.isna().sum()
```

```
itching                    0
skin_rash                  0
nodal_skin_eruptions       0
continuous_sneezing        0
shivering                  0
                        ...
blister                    0
red_sore_around_nose       0
yellow_crust_ooze          0
prognosis                  0
Unnamed: 133            4920
Length: 134, dtype: int64
```

Double-click (or enter) to edit

```
disease_training.drop('Unnamed: 133', inplace=True, axis=1)
```

```
disease_training.isna().sum()
```

```
itching                 0
skin_rash               0
nodal_skin_eruptions    0
continuous_sneezing     0
shivering               0
                       ..
inflammatory_nails      0
blister                 0
red_sore_around_nose    0
yellow_crust_ooze       0
prognosis               0
Length: 133, dtype: int64
```

```
#disease_training.head()
```

Convert category to numeric values

```
#get class labels

class_names = np.unique(disease_training.prognosis)
disease_training.prognosis = pd.Categorical(disease_training.prognosis)
disease_testing.prognosis = pd.Categorical(disease_testing.prognosis)
class_names.shape
```

```
     (41,)
```

```
#disease_training.prognosis.cat.codes
#disease_training
#disease_testing.head()
```

Separate Features and Label - Training

Training Set

```
X = disease_training.drop('prognosis', axis=1)
y = disease_training.prognosis.cat.codes
np.unique(y)
```

```
     array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
            17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
            34, 35, 36, 37, 38, 39, 40], dtype=int8)
```

Unseen Test Set

```
X_unseen = disease_testing.drop('prognosis', axis=1)
y_unseen = disease_testing.prognosis.cat.codes
np.unique(X_unseen)
```

```
     array([0, 1])
```

Split into Training & Validation Test

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=74)
```

```
#print(X_train.shape)
#print(y_train.shape)
#print(X_test.shape)
#print(y_test.shape)
```

```
X_test
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills
4007	0	0	0	0	0	0
1938	0	0	0	0	0	0
4462	0	0	0	0	0	0
3227	0	0	0	0	0	0
3889	0	0	0	0	0	0
...	...	...	...	...	...	...
111	0	0	0	0	0	0
3741	0	0	0	0	0	0
3280	1	1	1	0	0	0
3687	0	0	0	0	0	0
656	0	0	0	0	0	0

984 rows × 132 columns

```
# Number of features
FEATURES = 132
FEATURES
```

```
132
```

2. Model Training

Training configuration

```
FEATURES=X_train.shape[1]
N_VALIDATION = X_train.shape[0] *.2 #int(1e3)
```

```
N_TRAIN = X_train.shape[0]*.8 #int(1e4)
BUFFER_SIZE = int(100)
BATCH_SIZE = 50
STEPS_PER_EPOCH = N_TRAIN//BATCH_SIZE

[FEATURES, N_VALIDATION, N_TRAIN, BUFFER_SIZE, BATCH_SIZE, STEPS_PER_EPOCH]
```

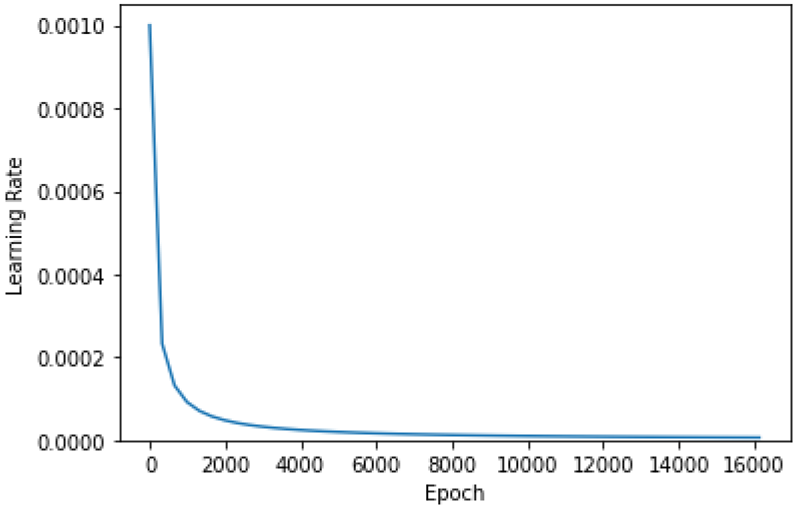
```
     [132, 787.2, 3148.8, 100, 50, 62.0]
```

Create Model

Find the ideal learning rate

```
lr_schedule = tf.keras.optimizers.schedules.InverseTimeDecay(
  0.001,
  decay_steps=STEPS_PER_EPOCH*100,
  decay_rate=1,
  staircase=False)

def get_optimizer():
  return tf.keras.optimizers.Adam(lr_schedule)
```

```
step = np.linspace(0,1000000)
lr = lr_schedule(step)
plt.figure(figsize = (6,4))
plt.plot(step/STEPS_PER_EPOCH, lr)
plt.ylim([0,max(plt.ylim())])
plt.xlabel('Epoch')
_ = plt.ylabel('Learning Rate')
```



```
#metrics = [
#    tfma.metrics.ExampleCount(name='example_count'),
#    tf.keras.metrics.SparseCategoricalCrossentropy(
#        name='sparse_categorical_crossentropy'),
#    tf.keras.metrics.SparseCategoricalAccuracy(name='accuracy'),
#    tf.keras.metrics.Precision(name='precision', top_k=1),
#    tf.keras.metrics.Precision(name='precision', top_k=3),
#    tf.keras.metrics.Recall(name='recall', top_k=1),
#    tf.keras.metrics.Recall(name='recall', top_k=3),
#    tfma.metrics.MultiClassConfusionMatrixPlot(
#        name='multi_class_confusion_matrix_plot'),
#]

METRICS = 'accuracy'
LOSS = tf.keras.losses.SparseCategoricalCrossentropy()
```

Settings for automation

```
def get_callbacks(name):
  return [
    tfdocs.modeling.EpochDots(),
    tf.keras.callbacks.EarlyStopping(monitor='acc', patience=100),
    tf.keras.callbacks.TensorBoard(os.path.join(logdir,name)),
  ]


def compile_and_fit(model, name, loss=None, optimizer=None, metrics = None, max_epochs=10000)
  if optimizer is None:
    optimizer = get_optimizer()

  if loss is None:
    loss = LOSS
  if metrics is None:
    metrics = [METRICS]

  model.compile(
      optimizer=optimizer,
      loss=loss,
      metrics=metrics
  )

  model.summary()

  history = model.fit(
    X_train,
    y_train,
    steps_per_epoch = STEPS_PER_EPOCH,
    epochs=max_epochs,
    validation_split=0.1,
    #validation_data=[X_test, y_test],
    callbacks=get_callbacks(name),
    verbose=0)
  return history
```

Models

```
size_histories = {}
```

Model 1

Simple model with 3 layers

```
model1 = tf.keras.Sequential([
    layers.Dense(4, activation='elu', input_shape=(FEATURES,)),
    layers.Dense(41, activation=tf.keras.activations.softmax)
])
```

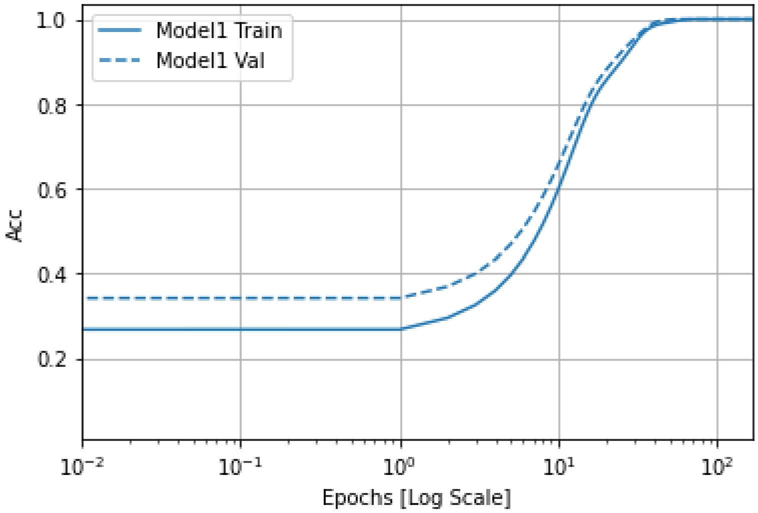
```
model1_history = compile_and_fit(
    model1,
    'models/model1',
    loss=LOSS,
    metrics=['acc']
)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 4)                 532       
                                                                 
 dense_1 (Dense)             (None, 41)                205       
                                                                 
=================================================================
Total params: 737
Trainable params: 737
Non-trainable params: 0
_________________________________________________________________

Epoch: 0, acc:0.0678,  loss:3.6389,  val_acc:0.1574,  val_loss:3.5249,  
..........................................................................................
Epoch: 100, acc:1.0000,  loss:0.0443,  val_acc:1.0000,  val_loss:0.0440,  
.....................................................................
```

```
size_histories['model1'] = model1_history
```

```
plotter = tfdocs.plots.HistoryPlotter(metric = 'acc', smoothing_std=10)
plotter.plot(size_histories)
a = plt.xscale('log')

plt.xlim([.01, max(plt.xlim())])
plt.ylim([.01, max(plt.ylim())])
plt.xlabel("Epochs [Log Scale]")
```

```
Text(0.5, 0, 'Epochs [Log Scale]')
```



```
loss, acc = model1.evaluate(X_test, y_test)
print(f"Model Loss (Test Set) : {loss}")
print(f"Model Accuracy (Test Set): {acc}")
```

```
31/31 [==============================] - 0s 1ms/step - loss: 0.0093 - acc: 1.0000
Model Loss (Test Set) : 0.009301541373133 66
Model Accuracy (Test Set): 1.0
```

```
#lrs = 1e-4 * (10 ** (tf.range(BATCH_SIZE)/20))
#plt.figure(figsize=(6,4))
#plt.semilogx(lrs, size_histories['models/model1'].history['loss'])
#plt.xlabel("Learning Rate")
#plt.ylabel("Loss")
#plt.title("Learning Rate vs Loss")
```

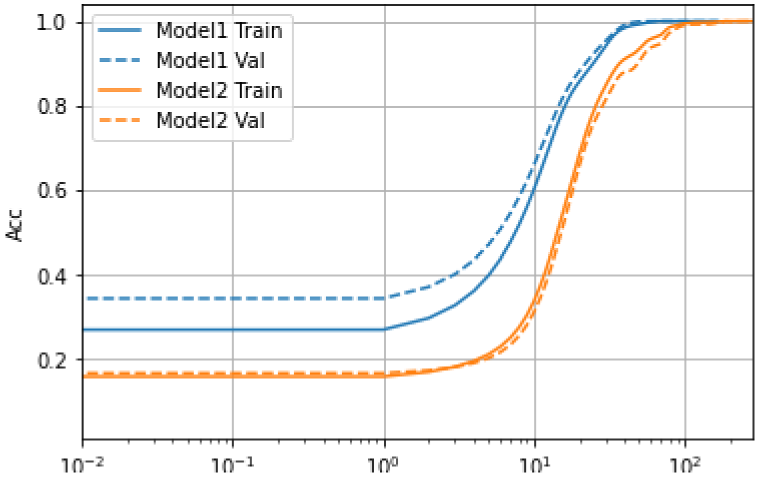
Model 2

```
model2 = tf.keras.Sequential([
    layers.Dense(4, activation='elu', input_shape=(FEATURES,)),
    layers.Dense(4, activation='elu', input_shape=(FEATURES,)),
    layers.Dense(41, activation=tf.keras.activations.softmax)
])

#model.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(),
#                 optimizer = tf.keras.optimizers.Adam(),
#                 #metrics=['MultiClassConfusionMatrixPlot'])
#                 metrics=["accuracy"])

#scheduler = tf.keras.callbacks.LearningRateScheduler(lambda epoch: 1e-4 * 10 **(epoch/20))

#history = model.fit(X_train, y_train, epochs=40, callbacks=[scheduler])


model2_history = compile_and_fit(
    model2,
    'models/model2',
    loss=LOSS,
    metrics=['acc']
)
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_2 (Dense)             (None, 4)                 532

 dense_3 (Dense)             (None, 4)                 20

 dense_4 (Dense)             (None, 41)                205

=================================================================
Total params: 757
Trainable params: 757
Non-trainable params: 0
_________________________________________________________________

Epoch: 0, acc:0.0810,  loss:3.6838,  val_acc:0.1269,  val_loss:3.6242,
.......................................................................................
Epoch: 100, acc:0.9944,  loss:0.1163,  val_acc:0.9898,  val_loss:0.1270,
.......................................................................................
Epoch: 200, acc:0.9994,  loss:0.0206,  val_acc:1.0000,  val_loss:0.0243,
..................................................................................
```

```
size_histories['model2'] = model2_history


plotter = tfdocs.plots.HistoryPlotter(metric = 'acc', smoothing_std=10)
plotter.plot(size_histories)
a = plt.xscale('log')

plt.xlim([.01, max(plt.xlim())])
plt.ylim([.01, max(plt.ylim())])
plt.xlabel("Epochs [Log Scale]")
```

```
Text(0.5, 0, 'Epochs [Log Scale]')
```



Model 3

```
model3 = tf.keras.Sequential([
    layers.Dense(64, activation='elu', input_shape=(FEATURES,)),
    layers.Dense(64, activation='elu', input_shape=(FEATURES,)),
    layers.Dense(64, activation='elu', input_shape=(FEATURES,)),
    layers.Dense(41, activation=tf.keras.activations.softmax)
])
```

```
model3_history = compile_and_fit(
    model3,
    'models/model3',
    loss=LOSS,
    metrics=['acc']
)
```

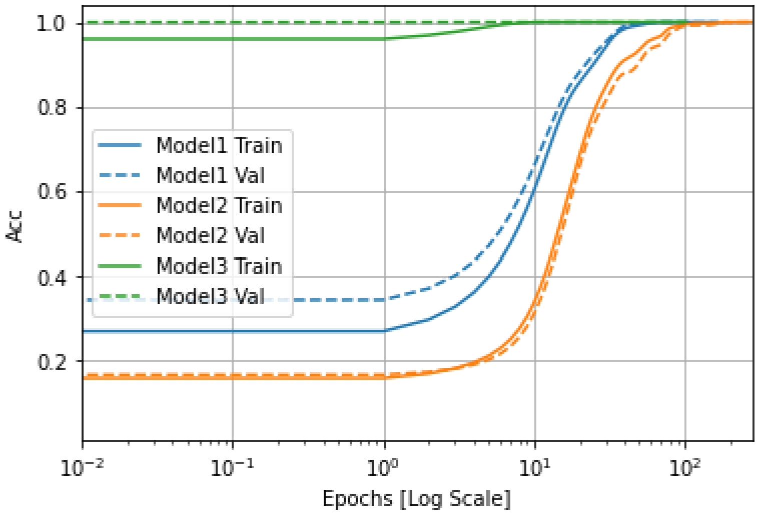
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_5 (Dense)             (None, 64)                8512

 dense_6 (Dense)             (None, 64)                4160

 dense_7 (Dense)             (None, 64)                4160

 dense_8 (Dense)             (None, 41)                2665

=================================================================
Total params: 19,497
Trainable params: 19,497
Non-trainable params: 0
_________________________________________________________________

Epoch: 0, acc:0.7555,  loss:2.3684,  val_acc:0.9975,  val_loss:0.8737,
....................................................................................................
```

```
Epoch: 100, acc:1.0000,  loss:0.0000,  val_acc:1.0000,  val_loss:0.0000,
...
```

```
size_histories['model3'] = model3_history
```

```
plotter = tfdocs.plots.HistoryPlotter(metric = 'acc', smoothing_std=10)
plotter.plot(size_histories)
a = plt.xscale('log')

plt.xlim([.01, max(plt.xlim())])
plt.ylim([.01, max(plt.ylim())])
plt.xlabel("Epochs [Log Scale]")
```

```
Text(0.5, 0, 'Epochs [Log Scale]')
```



Model 4

```
model4 = tf.keras.Sequential([
    layers.Dense(512, activation='elu', input_shape=(FEATURES,)),
    layers.Dense(512, activation='elu', input_shape=(FEATURES,)),
    layers.Dense(512, activation='elu', input_shape=(FEATURES,)),
    layers.Dense(512, activation='elu', input_shape=(FEATURES,)),
    layers.Dense(41, activation=tf.keras.activations.softmax)
])
```

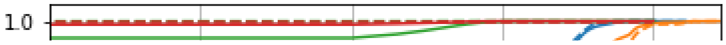
```
model4_history = compile_and_fit(
    model4,
    'models/model4',
    loss=LOSS,
    metrics=['acc']
)
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_9 (Dense)             (None, 512)               68096     
                                                                 
 dense_10 (Dense)            (None, 512)               262656    
                                                                 
 dense_11 (Dense)            (None, 512)               262656    
                                                                 
 dense_12 (Dense)            (None, 512)               262656    
                                                                 
 dense_13 (Dense)            (None, 41)                21033     
                                                                 
=================================================================
Total params: 877,097
Trainable params: 877,097
Non-trainable params: 0
_________________________________________________________________

Epoch: 0, acc:0.9500,  loss:0.3242,  val_acc:1.0000,  val_loss:0.0005,  
..........................................................................................
Epoch: 100, acc:1.0000,  loss:0.0000,  val_acc:1.0000,  val_loss:0.0000,  
..
```

```
size_histories['model4'] = model4_history
```

```
plotter = tfdocs.plots.HistoryPlotter(metric = 'acc', smoothing_std=10)
plotter.plot(size_histories)
a = plt.xscale('log')

plt.xlim([.01, max(plt.xlim())])
plt.ylim([.01, max(plt.ylim())])
plt.xlabel("Epochs [Log Scale]")
```

```
Text(0.5, 0, 'Epochs [Log Scale]')
```



Model 5

3. Evaluate Model

Evaluate with test data

```
loss, acc = model1.evaluate(X_test, y_test)
print(f"Model Loss (Test Set) : {loss}")
print(f"Model Accuracy (Test Set): {acc}")
```

```
31/31 [==============================] - 0s 1ms/step - loss: 0.0093 - acc: 1.0000
Model Loss (Test Set) : 0.00930154137313366
Model Accuracy (Test Set): 1.0
```

Evaluate with unseen data (Loss vs Accuracy)

```
loss, acc = model1.evaluate(X_unseen, y_unseen)
print("Model 1:")
print(f"Model Loss: {loss}")
print(f"Model Accuracy: {acc}")

loss, acc = model2.evaluate(X_unseen, y_unseen)
print("Model 2:")
print(f"Model Loss: {loss}")
print(f"Model Accuracy: {acc}")

loss, acc = model3.evaluate(X_unseen, y_unseen)
print("Model 3:")
print(f"Model Loss: {loss}")
print(f"Model Accuracy: {acc}")

loss, acc = model4.evaluate(X_unseen, y_unseen)
print("Model 4:")
print(f"Model Loss: {loss}")
print(f"Model Accuracy: {acc}")
```

```
2/2 [==============================] - 0s 6ms/step - loss: 0.3150 - acc: 0.9762
Model 1:
Model Loss: 0.3149896264076233
Model Accuracy: 0.976190447807312
2/2 [==============================] - 0s 5ms/step - loss: 0.0231 - acc: 0.9762
Model 2:
Model Loss: 0.02305091917514801
Model Accuracy: 0.976190447807312
2/2 [==============================] - 0s 6ms/step - loss: 0.2116 - acc: 0.9762
Model 3:
Model Loss: 0.21163904666900635
Model Accuracy: 0.976190447807312
2/2 [==============================] - 0s 7ms/step - loss: 0.0106 - acc: 1.0000
Model 4:
Model Loss: 0.010562791489064693
Model Accuracy: 1.0
```

3.

Test Set

```
### Test set
predictions = model1.predict(X_test)
```

```
31/31 [==============================] - 0s 1ms/step
```

```
predicted=tf.argmax(predictions, axis=1)
res= pd.DataFrame({'Test':y_test, 'B':predicted})
```

```
# print side by side
summary = pd.DataFrame({'Test Set':y_test, 'Predicted':predicted})
summary
```

	Test Set	Predicted
4007	39	39
4462	31	31

Unseen Test Data

```
### Unseen set
predictions1 = model1.predict(X_unseen)
```

```
     2/2 [==============================] - 0s 4ms/step
```

```
X_unseen.shape
```

```
     (42, 132)
```

```
result1=tf.argmax(predictions1, axis=1)
res1= pd.DataFrame({'Unseem Set':y_unseen, 'B':result1})
res1
```

```
10   23  23
11   30  30
12    7   7
13   32  32
14   28  28
15   29  29
16    8   8
17   11  11
18   37  37
19   40  40
20   19  19
21   20  20
22   21  21
23   22  22
24    3   3
25   36  36
26   10  10
27   34  34
28   13  13
29   18  18
30   39  39
31   26  26
32   24  24
33   25  25
34   31  31
35    5   5
```

Confusion Matrix

```
37    2   2
```

```
#!pip install tensorflow_addons

from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
```

```
y_pred = model1.predict(X_test)

import tensorflow_addons as tfa

metric = tfa.metrics.MultiLabelConfusionMatrix(num_classes=41)
rr=np.argmax(y_pred, axis=1)
print(len(y_test))
print(len(rr))
```

```
31/31 [==============================] - 0s 2ms/step
984
984
1.0
```

Accuracy Score

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, rr)
```

```
1.0
```

Multilabel confusion matrix

```
print("Actual \n", y_test)
print("\nPredicted \n",rr)
```

```
Actual 
 4007    39
1938    18
4462    31
3227    18
3889     5
        ..
111     30
3741    23
3280    15
3687    38
656      3
Length: 984, dtype: int8

Predicted 
 [39 18 31 18  5 15 27 13 24 29  2  0 24 35 25 38  2 13 22 27 27 18 21  3
 13 27 14  0 20 27 40 23 34 14 26 10  3 25 22  5 11 20 24 20 29 33 31 16
  8 40 24  1 36 11  4 18 18 15 21 33 24 34 11 23 23 33 22 12 28 27 18 28
 21  8 29  0 20 27 30 23 30 40 33 30  6  6 33 18 27 19 20  4 39 16 16  8
 34 25 39 15  2 29 31 26 39 11 30  6  1 40  6 31  0 35 28 31 11 16 27  8
```

```
      37 34 11 19 29  7 29 11 31 34 32  4 19  5 19 21 35 20 26 15 29 40 25 22
       5 11 39  3  6 38 28  8 11 39 20 19 38 34  7 36 12  8  7 26 14 37  6 35
      18 13 28  4  3 30 12 20 25 32 16 38 40 18 33 28 25 26 25 13  2 12  5 38
      13  1  2  8 30  9 21 39 20 17 16 34 10 18 39  7 25 14  8 12 23 20 37 10
       6 22  2 25 14  8 28  3 34 39  8 12 18 35 18  5  0 28 17 14 29 31  0  6
      30  4 21  0  6 19  9 20 37 30 14 23 34 14 33 35 26 28 28 14 31 33 26 23
      13  6  6 22 17 33 35  9  9 34 19  9 23 24 27 26  8 34  6 31 16  3 25  3
      27  2 13  6 24  0 28 26 34 39 10 14 29  1 36 22 26 35 32 39  4 33 38 40
      20  3 18  4 36 38  8 32  9 15  8  1 25  8 35 13 26  7 27 31  2 22 10 20
      21  7 34 11 21 34 12 36 32  2 34  6 30 33 21 17 28  7 21 30 19  0  6 26
      32  3 28 35 10 16 35  7 13 24  6 31  0 28 12  1 34 27 36  0 33  0  5 35
      33 26 38  6 40 26 18 10 40 28 34 36 30 23 37  6 29 17 39 34 20 18 12 20
      23 38 17 29 19 15 37 32  2 23 25 26 29 38 40 16 34 40 33 21  4  2 17 36
      32 19  1 16  7  2 34 35 30  1 23 34 18 36 27 22 33 18  7  1 17 18 26 15
       4 34 10 24 27 31 28  0 18 35 23 24  9 38 24 13 10 33 23 10  0 30  4 39
       6 39 10 10 34 12 16 35  3  7 40 27 13  8 22 22 37 14 10 29 38 24 20 32
      37 39 10  5 16  8 36  4 17 15  4 35 16  3 25 25 13 30 16 36 12 35 12 15
       1  6  6 39 22 34 37  1  9 22 32 39 25 13 13  6 23  3 17 12 32  6  9 24
      29 36  7  2  4 24 38  0  1 20  6  2 22 10 19 37 23 15 39 14 17 12 12 30
      11 39  8 21 23 34  7 22  6 20 11  3 14 12 15 24 36 12 22 34  3 36 17 22
      14 13  4 23 30 14  1 25 19  7 33 25 25 30 30 30 17 13  1 18 23 25 30  0
      17 34  5 27  8 20  5 18  4 16 15 15 39 10 30 21 23 15 16 39 39 13 21  5
      26 20 22 40 35  9  9 29  9 16 16 33 39 19  5 15 14 18 28 32 18 16  0  6
       4 11 16 18 40 14 33 39 29 37 13 17 32 19 28 22  6 32 10 30 37  4 29 38
      22 25 36 34  9 14  1  7 22 31 22 13 20 24 20 36  2 29 18  3 23  6 13 16
       4  8  8 33  1  4  1 36  9 22  9  2 13 14 24 19 37  1 27 19  1 11 38  9
      33 28  8  0 19 10 19 11  6 12 26 19 13 10 24  6 26 19  5  4 32 11  8 32
       0  3 19 24  2 27 26 13 40 10 26 32 34 21 27 13 22 13 31 12 34 15 13 29
       6 34 29 28  1 28 22  5 24 23 22  1 30 18 36 13 32  2 26  2 34  4 35 36
       8 28 38 17 33 19  8 31 36  8 18 31  7  0 26 37  3 16 40 18 13  7 24 21
      14 26 26 28 21 28 30 30 26 17 33  3  0 29 34 37  7  3 21  3 13 14  6 10
      15 30 35  9 24  7 21 19 17 10 13  2 36 10  8  8 25 28  2 28  4 39 32  2
      12 22 22 16 16 23 22 35  3 26 24 40 10 31  2 35  8  6 18 15 14 24  4  5
      34 20  9 28  2  2 12 19  3  5 11 31  4 13  5 14 17 34 12 10 19 33 39  4
      25  5  7 21 38  4 30 18 25 10  1 18 24 35 32 11 38 26  3  1  6 11 29 21
      23  9 35 11 11 22  3 31 39 26 35  9 19  4 18 37 40 13 25 30 23 15 38  3]
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, rr)
```

```
     array([[21,  0,  0, ...,  0,  0,  0],
            [ 0, 23,  0, ...,  0,  0,  0],
            [ 0,  0, 25, ...,  0,  0,  0],
            ...,
            [ 0,  0,  0, ..., 19,  0,  0],
            [ 0,  0,  0, ...,  0, 27,  0],
            [ 0,  0,  0, ...,  0,  0, 18]])
```

```
from sklearn.metrics import classification_report

label_names = ['label A', 'label B', 'label C', 'label D']

print(classification_report(y_unseen, result1,target_names=class_names))
```

```
                                         precision    recall  f1-score   support

(vertigo) Paroymsal  Positional Vertigo       1.00      1.00      1.00         1
                                   AIDS       1.00      1.00      1.00         1
                                   Acne       1.00      1.00      1.00         1
                    Alcoholic hepatitis       1.00      1.00      1.00         1
                                Allergy       1.00      1.00      1.00         1
                              Arthritis       1.00      1.00      1.00         1
                       Bronchial Asthma       1.00      1.00      1.00         1
                   Cervical spondylosis       1.00      1.00      1.00         1
                            Chicken pox       1.00      1.00      1.00         1
                    Chronic cholestasis       1.00      1.00      1.00         1
                            Common Cold       1.00      1.00      1.00         1
                                 Dengue       1.00      1.00      1.00         1
                              Diabetes        1.00      1.00      1.00         1
           Dimorphic hemmorhoids(piles)       1.00      1.00      1.00         1
                          Drug Reaction       1.00      1.00      1.00         1
                       Fungal infection       1.00      0.50      0.67         2
                                   GERD       0.50      1.00      0.67         1
                        Gastroenteritis       1.00      1.00      1.00         1
                           Heart attack       1.00      1.00      1.00         1
                            Hepatitis B       1.00      1.00      1.00         1
                            Hepatitis C       1.00      1.00      1.00         1
                            Hepatitis D       1.00      1.00      1.00         1
                            Hepatitis E       1.00      1.00      1.00         1
                          Hypertension        1.00      1.00      1.00         1
                        Hyperthyroidism       1.00      1.00      1.00         1
                           Hypoglycemia       1.00      1.00      1.00         1
                         Hypothyroidism       1.00      1.00      1.00         1
                               Impetigo       1.00      1.00      1.00         1
                               Jaundice       1.00      1.00      1.00         1
                                Malaria       1.00      1.00      1.00         1
                               Migraine       1.00      1.00      1.00         1
                        Osteoarthristis       1.00      1.00      1.00         1
           Paralysis (brain hemorrhage)       1.00      1.00      1.00         1
                    Peptic ulcer diseae       1.00      1.00      1.00         1
                              Pneumonia       1.00      1.00      1.00         1
                              Psoriasis       1.00      1.00      1.00         1
                           Tuberculosis       1.00      1.00      1.00         1
                                Typhoid       1.00      1.00      1.00         1
                Urinary tract infection       1.00      1.00      1.00         1
                         Varicose veins       1.00      1.00      1.00         1
                            hepatitis A       1.00      1.00      1.00         1

                               accuracy                           0.98        42
                              macro avg       0.99      0.99      0.98        42
                           weighted avg       0.99      0.98      0.98        42
```