```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a v
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session


# Importing libraries
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from sklearn.preprocessing import LabelEncoder
import warnings
warnings.filterwarnings("ignore")

pd.set_option('display.max_columns', None)


# Reading data
data = pd.read_csv("Training.csv")
data.head()
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills
0	1	1	1	0	0	0
1	0	1	1	0	0	0
2	1	0	1	0	0	0
3	1	1	0	0	0	0
4	1	1	1	0	0	0

```
# Checking statistical summary
data.describe(include='all')
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shiveri
count	4920.000000	4920.000000	4920.000000	4920.000000	4920.0000
unique	NaN	NaN	NaN	NaN	N
top	NaN	NaN	NaN	NaN	N
freq	NaN	NaN	NaN	NaN	N
mean	0.137805	0.159756	0.021951	0.045122	0.0219
std	0.344730	0.366417	0.146539	0.207593	0.1465
min	0.000000	0.000000	0.000000	0.000000	0.0000
25%	0.000000	0.000000	0.000000	0.000000	0.0000
50%	0.000000	0.000000	0.000000	0.000000	0.0000
75%	0.000000	0.000000	0.000000	0.000000	0.0000
max	1.000000	1.000000	1.000000	1.000000	1.0000

```
# Removing redundant column
data.drop("Unnamed: 133", axis=1, inplace=True)
data.head()
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills
0	1	1	1	0	0	0
1	0	1	1	0	0	0
2	1	0	1	0	0	0
3	1	1	0	0	0	0
4	1	1	1	0	0	0

```
# Checking null values
print(data.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
```

```
# Calculating number of classes
len(data["prognosis"].unique())
```

```
41
```

```
# Division of dataset
X_train = data.drop("prognosis", axis=1)
Y_train = data["prognosis"]
```

```
# Encoding the target's labels
le = LabelEncoder()
le.fit(Y_train)
Y_train = le.transform(Y_train)
Y_train = tf.keras.utils.to_categorical(Y_train)
Y_train
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

```
# Checking X_train shape
X_train.shape
```

```
(4920, 132)
```

```
# Defining the model
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation="relu", input_shape=(None, X_train.shape[0], X_train.shape[1])),
    tf.keras.layers.Dense(128, activation="relu"),
    tf.keras.layers.Dense(41, activation="softmax")
])
# Compiling the model
model.compile(
        loss = tf.keras.losses.CategoricalCrossentropy(),
        optimizer = tf.keras.optimizers.Adam(),
        metrics = "accuracy"
)
```

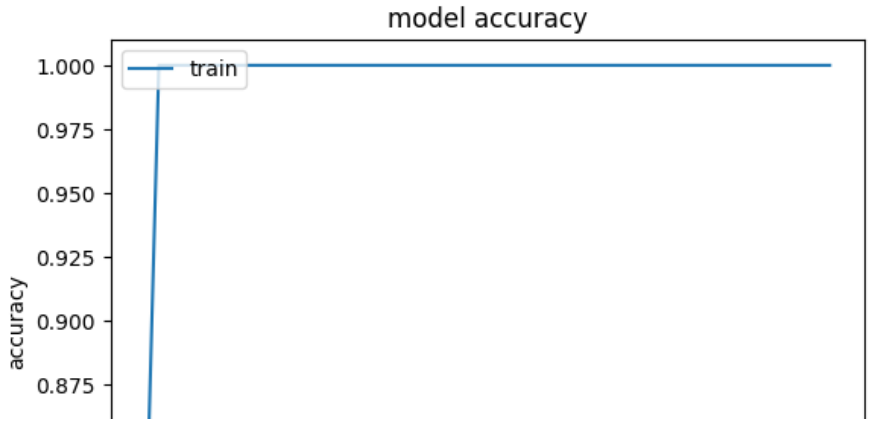
```
# Training the model
history = model.fit(
   X_train,
   Y_train,
   epochs=50
)
```

```
Epoch 1/50
154/154 [==============================] - 2s 4ms/step - loss: 1.6243 - accuracy: 0.8004
Epoch 2/50
154/154 [==============================] - 1s 7ms/step - loss: 0.0393 - accuracy: 1.0000
Epoch 3/50
154/154 [==============================] - 1s 6ms/step - loss: 0.0098 - accuracy: 1.0000
Epoch 4/50
154/154 [==============================] - 1s 7ms/step - loss: 0.0047 - accuracy: 1.0000
Epoch 5/50
154/154 [==============================] - 1s 5ms/step - loss: 0.0027 - accuracy: 1.0000
Epoch 6/50
154/154 [==============================] - 1s 6ms/step - loss: 0.0018 - accuracy: 1.0000
Epoch 7/50
154/154 [==============================] - 1s 7ms/step - loss: 0.0013 - accuracy: 1.0000
Epoch 8/50
154/154 [==============================] - 1s 4ms/step - loss: 9.3653e-04 - accuracy: 1.0000
Epoch 9/50
154/154 [==============================] - 1s 6ms/step - loss: 7.2222e-04 - accuracy: 1.0000
Epoch 10/50
154/154 [==============================] - 1s 4ms/step - loss: 5.6972e-04 - accuracy: 1.0000
Epoch 11/50
154/154 [==============================] - 1s 4ms/step - loss: 4.5715e-04 - accuracy: 1.0000
Epoch 12/50
154/154 [==============================] - 1s 5ms/step - loss: 3.7359e-04 - accuracy: 1.0000
Epoch 13/50
154/154 [==============================] - 1s 4ms/step - loss: 3.1075e-04 - accuracy: 1.0000
Epoch 14/50
154/154 [==============================] - 1s 5ms/step - loss: 2.5991e-04 - accuracy: 1.0000
Epoch 15/50
154/154 [==============================] - 1s 6ms/step - loss: 2.2159e-04 - accuracy: 1.0000
Epoch 16/50
154/154 [==============================] - 1s 4ms/step - loss: 1.8930e-04 - accuracy: 1.0000
Epoch 17/50
154/154 [==============================] - 1s 4ms/step - loss: 1.6235e-04 - accuracy: 1.0000
```

```
Epoch 18/50
154/154 [==============================] - 1s 6ms/step - loss: 1.4104e-04 - accuracy: 1.0000
Epoch 19/50
154/154 [==============================] - 1s 9ms/step - loss: 1.2396e-04 - accuracy: 1.0000
Epoch 20/50
154/154 [==============================] - 1s 6ms/step - loss: 1.0776e-04 - accuracy: 1.0000
Epoch 21/50
154/154 [==============================] - 1s 4ms/step - loss: 9.4779e-05 - accuracy: 1.0000
Epoch 22/50
154/154 [==============================] - 0s 2ms/step - loss: 8.3773e-05 - accuracy: 1.0000
Epoch 23/50
154/154 [==============================] - 0s 2ms/step - loss: 7.4145e-05 - accuracy: 1.0000
Epoch 24/50
154/154 [==============================] - 0s 2ms/step - loss: 6.6002e-05 - accuracy: 1.0000
Epoch 25/50
154/154 [==============================] - 0s 2ms/step - loss: 5.8629e-05 - accuracy: 1.0000
Epoch 26/50
154/154 [==============================] - 0s 2ms/step - loss: 5.2376e-05 - accuracy: 1.0000
Epoch 27/50
154/154 [==============================] - 0s 2ms/step - loss: 4.6888e-05 - accuracy: 1.0000
Epoch 28/50
154/154 [==============================] - 0s 2ms/step - loss: 4.1979e-05 - accuracy: 1.0000
Epoch 29/50
154/154 [==============================] - 0s 2ms/step - loss: 3.7825e-05 - accuracy: 1.0000
```

```
# Summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# Summarize history for loss
plt.plot(history.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train'], loc='upper left')
plt.show()
```




```
# Testing the model
test_data = pd.read_csv("Testing.csv")
test_data.head()
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills
0	1	1	1	0	0	0
1	0	0	0	1	1	1
2	0	0	0	0	0	0
3	1	0	0	0	0	0
4	1	1	0	0	0	0

s

```
# Splitting the test data
X_test = test_data.drop("prognosis", axis=1).values
Y_test = test_data["prognosis"].values

# Encoding Y_test
Y_test = le.fit_transform(Y_test)
Y_test = tf.keras.utils.to_categorical(Y_test)
Y_test
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

```
# Testing the model
test_loss, test_acc = model.evaluate(X_test,  Y_test, verbose=2)
```

```
2/2 - 1s - loss: 0.1126 - accuracy: 0.9762 - 555ms/epoch - 278ms/step
```