Multiclassclassification

April 28, 2024

[ ]:
```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```

[ ]:
```python
columns = ["lettr", "x-box", "y-box", "width", "height", "onpix", "x-bar",
"y-bar", "x2bar", "y2bar", "xybar", "x2ybr", "xy2br", "x-ege", "xegvy",
"y-ege", "yegvx"]
```

[ ]:
```python
df = pd.read_csv('/content/drive/MyDrive/DL/2_letter_recognition.data',␣
↪names=columns)
```

[ ]:
```python
df
```

[ ]:
```
      lettr  x-box  y-box  width  height  onpix  x-bar  y-bar  x2bar  y2bar  \
0         T      2      8      3       5      1      8     13      0      6
1         I      5     12      3       7      2     10      5      5      4
2         D      4     11      6       8      6     10      6      2      6
3         N      7     11      6       6      3      5      9      4      6
4         G      2      1      3       1      1      8      6      6      6
…       …    …    …    …    …    …    …    …    …    …
19995     D      2      2      3       3      2      7      7      7      6
19996     C      7     10      8       8      4      4      8      6      9
19997     T      6      9      6       7      5      6     11      3      7
19998     S      2      3      4       2      1      8      7      2      6
19999     A      4      9      6       6      2      9      5      3      1

       xybar  x2ybr  xy2br  x-ege  xegvy  y-ege  yegvx
0          6     10      8      0      8      0      8
1         13      3      9      2      8      4     10
2         10      3      7      3      7      3      9
3          4      4     10      6     10      2      8
4          6      5      9      1      7      5     10
…       …    …    …    …    …    …    …
19995      6      6      4      2      8      3      7
19996     12      9     13      2      9      3      7
19997     11      9      5      2     12      2      4
19998     10      6      8      1      9      5      8
```

```
19999      8      1      8      2      7      2      8

[20000 rows x 17 columns]
```

[ ]:
```
x = df.drop("lettr", axis=1).values
y = df["lettr"].values
```

[ ]:
```
x.shape
```

[ ]:
```
(20000, 16)
```

[ ]:
```
y.shape
```

[ ]:
```
(20000,)
```

[ ]:
```
np.unique(y)
```

[ ]:
```
array(['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',
       'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'],
      dtype=object)
```

[ ]:
```
x_train, x_test,y_train,y_test = train_test_split(x, y,test_size=0.2)
```

[ ]:
```
def shape():
    print("Train Shape:",x_train.shape)
    print("Test Shape:",x_test.shape)
    print("y_train shape:",y_train.shape)
    print("y_test shape:",y_test.shape)
shape()
```

```
Train Shape: (16000, 16)
Test Shape: (4000, 16)
y_train shape: (16000,)
y_test shape: (4000,)
```

[ ]:
```
x_train[0]
```

[ ]:
```
array([ 6, 11,  6,  8,  3,  7,  7, 15,  2,  4,  6,  8,  6,  8,  0,  8])
```

[ ]:
```
y_train[0]
```

[ ]:
```
'N'
```

[ ]:
```
class_names=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',
'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
```

[ ]:
```
x_test[10]
```

[ ]: array([4, 7, 6, 5, 2, 8, 3, 3, 3, 9, 1, 8, 2, 7, 3, 7])

[ ]:
```python
y_test[10]
```

[ ]: 'A'

[ ]:
```python
x_train = x_train/255
x_test = x_test/255
```

[ ]:
```python
from sklearn.preprocessing import LabelEncoder
```

[ ]:
```python
encoder = LabelEncoder()
y_train = encoder.fit_transform(y_train)
y_test = encoder.fit_transform(y_test)
```

[ ]:
```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
```

[ ]:
```python
model=Sequential()
model.add(Dense(512, activation='relu', input_shape=(16,)))
model.add(Dropout(0.2))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(26, activation='softmax'))
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               8704

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 26)                6682

=================================================================
Total params: 146714 (573.10 KB)
Trainable params: 146714 (573.10 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:
```
model.fit(x_train, y_train,epochs=50, batch_size=128, verbose=1,
validation_data=(x_test, y_test))
```

```
Epoch 1/50
125/125 [==============================] - 4s 16ms/step - loss: 3.1396 -
accuracy: 0.1478 - val_loss: 2.7767 - val_accuracy: 0.2720
Epoch 2/50
125/125 [==============================] - 1s 10ms/step - loss: 2.3879 -
accuracy: 0.3046 - val_loss: 2.0701 - val_accuracy: 0.4170
Epoch 3/50
125/125 [==============================] - 1s 9ms/step - loss: 1.9884 -
accuracy: 0.4021 - val_loss: 1.7956 - val_accuracy: 0.4870
Epoch 4/50
125/125 [==============================] - 1s 9ms/step - loss: 1.7631 -
accuracy: 0.4600 - val_loss: 1.6269 - val_accuracy: 0.5275
Epoch 5/50
125/125 [==============================] - 1s 9ms/step - loss: 1.6385 -
accuracy: 0.5010 - val_loss: 1.5284 - val_accuracy: 0.5575
Epoch 6/50
125/125 [==============================] - 1s 9ms/step - loss: 1.5386 -
accuracy: 0.5331 - val_loss: 1.4413 - val_accuracy: 0.5765
Epoch 7/50
125/125 [==============================] - 1s 8ms/step - loss: 1.4677 -
accuracy: 0.5602 - val_loss: 1.3720 - val_accuracy: 0.6152
Epoch 8/50
125/125 [==============================] - 1s 8ms/step - loss: 1.4035 -
accuracy: 0.5796 - val_loss: 1.3134 - val_accuracy: 0.6267
Epoch 9/50
125/125 [==============================] - 1s 9ms/step - loss: 1.3469 -
accuracy: 0.5957 - val_loss: 1.2731 - val_accuracy: 0.6310
Epoch 10/50
125/125 [==============================] - 1s 9ms/step - loss: 1.2974 -
accuracy: 0.6119 - val_loss: 1.2136 - val_accuracy: 0.6570
Epoch 11/50
125/125 [==============================] - 2s 13ms/step - loss: 1.2472 -
accuracy: 0.6320 - val_loss: 1.1663 - val_accuracy: 0.6572
Epoch 12/50
125/125 [==============================] - 2s 14ms/step - loss: 1.2012 -
accuracy: 0.6423 - val_loss: 1.1188 - val_accuracy: 0.6795
Epoch 13/50
125/125 [==============================] - 1s 11ms/step - loss: 1.1577 -
accuracy: 0.6542 - val_loss: 1.0791 - val_accuracy: 0.6950
Epoch 14/50
125/125 [==============================] - 1s 9ms/step - loss: 1.1155 -
accuracy: 0.6651 - val_loss: 1.0424 - val_accuracy: 0.6995
Epoch 15/50
125/125 [==============================] - 1s 9ms/step - loss: 1.0695 -
```

```
accuracy: 0.6801 - val_loss: 0.9925 - val_accuracy: 0.7117
Epoch 16/50
125/125 [==============================] - 1s 8ms/step - loss: 1.0438 -
accuracy: 0.6850 - val_loss: 0.9562 - val_accuracy: 0.7247
Epoch 17/50
125/125 [==============================] - 1s 8ms/step - loss: 1.0059 -
accuracy: 0.6966 - val_loss: 0.9246 - val_accuracy: 0.7315
Epoch 18/50
125/125 [==============================] - 1s 8ms/step - loss: 0.9795 -
accuracy: 0.7040 - val_loss: 0.8963 - val_accuracy: 0.7465
Epoch 19/50
125/125 [==============================] - 1s 9ms/step - loss: 0.9584 -
accuracy: 0.7112 - val_loss: 0.8993 - val_accuracy: 0.7368
Epoch 20/50
125/125 [==============================] - 1s 9ms/step - loss: 0.9289 -
accuracy: 0.7216 - val_loss: 0.8455 - val_accuracy: 0.7545
Epoch 21/50
125/125 [==============================] - 1s 9ms/step - loss: 0.9006 -
accuracy: 0.7287 - val_loss: 0.8292 - val_accuracy: 0.7605
Epoch 22/50
125/125 [==============================] - 1s 10ms/step - loss: 0.8817 -
accuracy: 0.7319 - val_loss: 0.8027 - val_accuracy: 0.7640
Epoch 23/50
125/125 [==============================] - 2s 14ms/step - loss: 0.8607 -
accuracy: 0.7343 - val_loss: 0.7837 - val_accuracy: 0.7732
Epoch 24/50
125/125 [==============================] - 2s 13ms/step - loss: 0.8464 -
accuracy: 0.7423 - val_loss: 0.7611 - val_accuracy: 0.7730
Epoch 25/50
125/125 [==============================] - 1s 10ms/step - loss: 0.8221 -
accuracy: 0.7511 - val_loss: 0.7510 - val_accuracy: 0.7765
Epoch 26/50
125/125 [==============================] - 1s 9ms/step - loss: 0.8012 -
accuracy: 0.7569 - val_loss: 0.7257 - val_accuracy: 0.7870
Epoch 27/50
125/125 [==============================] - 1s 9ms/step - loss: 0.7802 -
accuracy: 0.7600 - val_loss: 0.7274 - val_accuracy: 0.7818
Epoch 28/50
125/125 [==============================] - 1s 9ms/step - loss: 0.7764 -
accuracy: 0.7627 - val_loss: 0.7032 - val_accuracy: 0.7903
Epoch 29/50
125/125 [==============================] - 1s 8ms/step - loss: 0.7592 -
accuracy: 0.7659 - val_loss: 0.6874 - val_accuracy: 0.7960
Epoch 30/50
125/125 [==============================] - 1s 8ms/step - loss: 0.7429 -
accuracy: 0.7742 - val_loss: 0.6756 - val_accuracy: 0.7980
Epoch 31/50
125/125 [==============================] - 1s 9ms/step - loss: 0.7325 -
```

```
accuracy: 0.7756 - val_loss: 0.6547 - val_accuracy: 0.8075
Epoch 32/50
125/125 [==============================] - 1s 9ms/step - loss: 0.7134 -
accuracy: 0.7814 - val_loss: 0.6417 - val_accuracy: 0.8095
Epoch 33/50
125/125 [==============================] - 1s 9ms/step - loss: 0.7040 -
accuracy: 0.7817 - val_loss: 0.6386 - val_accuracy: 0.8083
Epoch 34/50
125/125 [==============================] - 2s 12ms/step - loss: 0.6937 -
accuracy: 0.7850 - val_loss: 0.6250 - val_accuracy: 0.8130
Epoch 35/50
125/125 [==============================] - 3s 21ms/step - loss: 0.6759 -
accuracy: 0.7904 - val_loss: 0.6069 - val_accuracy: 0.8205
Epoch 36/50
125/125 [==============================] - 1s 10ms/step - loss: 0.6657 -
accuracy: 0.7900 - val_loss: 0.6056 - val_accuracy: 0.8177
Epoch 37/50
125/125 [==============================] - 1s 8ms/step - loss: 0.6505 -
accuracy: 0.7979 - val_loss: 0.5928 - val_accuracy: 0.8205
Epoch 38/50
125/125 [==============================] - 1s 9ms/step - loss: 0.6450 -
accuracy: 0.8005 - val_loss: 0.5729 - val_accuracy: 0.8263
Epoch 39/50
125/125 [==============================] - 1s 8ms/step - loss: 0.6252 -
accuracy: 0.8079 - val_loss: 0.5614 - val_accuracy: 0.8307
Epoch 40/50
125/125 [==============================] - 1s 9ms/step - loss: 0.6208 -
accuracy: 0.8082 - val_loss: 0.5512 - val_accuracy: 0.8360
Epoch 41/50
125/125 [==============================] - 1s 9ms/step - loss: 0.6062 -
accuracy: 0.8123 - val_loss: 0.5427 - val_accuracy: 0.8367
Epoch 42/50
125/125 [==============================] - 1s 9ms/step - loss: 0.6023 -
accuracy: 0.8129 - val_loss: 0.5400 - val_accuracy: 0.8363
Epoch 43/50
125/125 [==============================] - 1s 8ms/step - loss: 0.5909 -
accuracy: 0.8163 - val_loss: 0.5363 - val_accuracy: 0.8375
Epoch 44/50
125/125 [==============================] - 1s 9ms/step - loss: 0.5773 -
accuracy: 0.8186 - val_loss: 0.5193 - val_accuracy: 0.8432
Epoch 45/50
125/125 [==============================] - 1s 12ms/step - loss: 0.5646 -
accuracy: 0.8256 - val_loss: 0.5075 - val_accuracy: 0.8462
Epoch 46/50
125/125 [==============================] - 2s 13ms/step - loss: 0.5571 -
accuracy: 0.8273 - val_loss: 0.5001 - val_accuracy: 0.8505
Epoch 47/50
125/125 [==============================] - 2s 13ms/step - loss: 0.5504 -
```

```
accuracy: 0.8303 - val_loss: 0.5037 - val_accuracy: 0.8435
Epoch 48/50
125/125 [==============================] - 1s 9ms/step - loss: 0.5376 -
accuracy: 0.8326 - val_loss: 0.5066 - val_accuracy: 0.8465
Epoch 49/50
125/125 [==============================] - 1s 9ms/step - loss: 0.5327 -
accuracy: 0.8365 - val_loss: 0.4740 - val_accuracy: 0.8577
Epoch 50/50
125/125 [==============================] - 1s 9ms/step - loss: 0.5280 -
accuracy: 0.8335 - val_loss: 0.4611 - val_accuracy: 0.8635
```

[ ]: <keras.src.callbacks.History at 0x7ee4d9761ab0>

[ ]:
```
predictions = model.predict(x_test)
```

```
125/125 [==============================] - 0s 2ms/step
```

[ ]:
```
index=10
print(predictions[index])
final_value=np.argmax(predictions[index])
print("Actual label:",y_test[index])
print("Predicted label:",final_value)
print("Class (A-Z):",class_names[final_value])
```

```
[9.8096299e-01 1.9918522e-09 2.3191149e-09 2.1174830e-05 1.2993358e-09
 1.2054219e-09 2.3324210e-05 8.5040911e-06 1.2225636e-03 1.0371854e-02
 1.1481132e-05 4.4757978e-04 1.9393451e-06 3.2419814e-07 3.3192793e-04
 4.9842871e-08 1.1108665e-04 4.9027985e-06 5.8424729e-03 3.6999367e-10
 4.3506113e-10 2.2929499e-12 4.7252440e-15 6.2943972e-04 5.4896348e-11
 8.2712686e-06]
Actual label: 0
Predicted label: 0
Class (A-Z): A
```