neuralnet

December 9, 2023

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```
data=pd.read_csv('dataset2.csv')
```

[ ]:
```
#check for missing values
#get rid of missing values
data=data.dropna(axis=0)
```

[ ]:
```
#get rid of peopleID
data=data.drop(['PeopleId'],axis=1)
```

[ ]:
```
#encode activityID
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
data['activityID']=le.fit_transform(data['activityID'])
```

[ ]:
```
y=data["activityID"]
X=data.drop(["activityID"],axis=1)
#split the data into training, validation and testing sets
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.
 ↪2,stratify=y,random_state=42)
```

[ ]:
```
import tensorflow as tf
from tensorflow.keras import layers, models
```

[ ]:
```
# Define the neural network model
model = models.Sequential()
model.add(layers.Dense(128, activation='relu', input_shape=(31,)))
model.add(layers.Dropout(0.5))  # Adding dropout for regularization
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dropout(0.5))
```

```
model.add(layers.Dense(13, activation='softmax'))  # 13 classes, softmax for␣
 ↪multi-class classification

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',␣
 ↪metrics=['accuracy'])

model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_31 (Dense)            (None, 128)               4096

 dropout_3 (Dropout)         (None, 128)               0

 dense_32 (Dense)            (None, 64)                8256

 dropout_4 (Dropout)         (None, 64)                0

 dense_33 (Dense)            (None, 32)                2080

 dropout_5 (Dropout)         (None, 32)                0

 dense_34 (Dense)            (None, 13)                429

=================================================================
Total params: 14,861
Trainable params: 14,861
Non-trainable params: 0
_________________________________________________________________
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_31 (Dense)            (None, 128)               4096

 dropout_3 (Dropout)         (None, 128)               0

 dense_32 (Dense)            (None, 64)                8256

 dropout_4 (Dropout)         (None, 64)                0

 dense_33 (Dense)            (None, 32)                2080

 dropout_5 (Dropout)         (None, 32)                0
```

```
 dense_34 (Dense)             (None, 13)                429

=================================================================
Total params: 14,861
Trainable params: 14,861
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2)
```

```
Epoch 1/10
57281/57281 [==============================] - 48s 837us/step - loss: 1.2819 -
accuracy: 0.5990 - val_loss: 0.8136 - val_accuracy: 0.7545
Epoch 2/10
57281/57281 [==============================] - 49s 849us/step - loss: 1.0712 -
accuracy: 0.6736 - val_loss: 0.7696 - val_accuracy: 0.7673
Epoch 3/10
57281/57281 [==============================] - 48s 836us/step - loss: 1.0454 -
accuracy: 0.6848 - val_loss: 0.7462 - val_accuracy: 0.7736
Epoch 4/10
57281/57281 [==============================] - 48s 837us/step - loss: 1.0347 -
accuracy: 0.6907 - val_loss: 0.7391 - val_accuracy: 0.7754
Epoch 5/10
57281/57281 [==============================] - 48s 842us/step - loss: 1.0234 -
accuracy: 0.6965 - val_loss: 0.7321 - val_accuracy: 0.7838
Epoch 6/10
57281/57281 [==============================] - 48s 837us/step - loss: 1.0201 -
accuracy: 0.6984 - val_loss: 0.7023 - val_accuracy: 0.7918
Epoch 7/10
57281/57281 [==============================] - 48s 845us/step - loss: 1.0149 -
accuracy: 0.7018 - val_loss: 0.7268 - val_accuracy: 0.7868
Epoch 8/10
57281/57281 [==============================] - 50s 866us/step - loss: 1.0094 -
accuracy: 0.7040 - val_loss: 0.7253 - val_accuracy: 0.7865
Epoch 9/10
57281/57281 [==============================] - 50s 880us/step - loss: 1.0074 -
accuracy: 0.7046 - val_loss: 0.6981 - val_accuracy: 0.7904
Epoch 10/10
57281/57281 [==============================] - 49s 863us/step - loss: 1.0058 -
accuracy: 0.7059 - val_loss: 0.6829 - val_accuracy: 0.7973
```

[ ]:
```
<keras.callbacks.History at 0x2f9d9a250>
```

[ ]:
```
#examine using test dataset
test_loss, test_acc = model.evaluate(X_test, y_test)
print('test_acc: ',test_acc)
print('test_loss: ',test_loss)
```

```
17901/17901 [==============================] - 7s 387us/step - loss: 0.6845 -
accuracy: 0.7968
test_acc:  0.7967988848686218
test_loss:  0.6844893097877502
```

[ ]:
```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2864010 entries, 0 to 2864055
Data columns (total 33 columns):
 #   Column                      Dtype
---  ------                      -----
 0   activityID                  object
 1   heart_rate                  float64
 2   hand temperature (°C)       float64
 3   hand acceleration X ±16g    float64
 4   hand acceleration Y ±16g    float64
 5   hand acceleration Z ±16g    float64
 6   hand gyroscope X            float64
 7   hand gyroscope Y            float64
 8   hand gyroscope Z            float64
 9   hand magnetometer X         float64
 10  hand magnetometer Y         float64
 11  hand magnetometer Z         float64
 12  chest temperature (°C)      float64
 13  chest acceleration X ±16g   float64
 14  chest acceleration Y ±16g   float64
 15  chest acceleration Z ±16g   float64
 16  chest gyroscope X           float64
 17  chest gyroscope Y           float64
 18  chest gyroscope Z           float64
 19  chest magnetometer X        float64
 20  chest magnetometer Y        float64
 21  chest magnetometer Z        float64
 22  ankle temperature (°C)      float64
 23  ankle acceleration X ±16g   float64
 24  ankle acceleration Y ±16g   float64
 25  ankle acceleration Z ±16g   float64
 26  ankle gyroscope X           float64
 27  ankle gyroscope Y           float64
 28  ankle gyroscope Z           float64
 29  ankle magnetometer X        float64
 30  ankle magnetometer Y        float64
 31  ankle magnetometer Z        float64
 32  PeopleId                    int64
dtypes: float64(31), int64(1), object(1)
memory usage: 742.9+ MB
None
```

[ ]: