Regression-NeuralNetworks

April 9, 2022

[1]:
```
import pandas as pd

data = pd.read_csv('C:\\Users\\MSI␣
↪Stealth\\Downloads\\BMEN415Project\\regression\\Volumetric_features.csv')

print(data.head())
```

```
   S.No  Left-Lateral-Ventricle  Left-Inf-Lat-Vent  \
0     1                 22916.9              982.7
1     2                 22953.2              984.5
2     3                 23320.4             1062.1
3     4                 24360.0             1000.5
4     5                 25769.4             1124.4

   Left-Cerebellum-White-Matter  Left-Cerebellum-Cortex  Left-Thalamus  \
0                       15196.7                 55796.4         6855.5
1                       15289.7                 55778.6         6835.1
2                       15382.1                 55551.2         7566.0
3                       14805.4                 54041.8         8004.6
4                       16331.1                 54108.6         6677.4

   Left-Caudate  Left-Putamen  Left-Pallidum  3rd-Ventricle  …  \
0        2956.4        4240.7         2223.9         2034.4  …
1        3064.2        4498.6         2354.1         1927.1  …
2        3231.7        4456.2         1995.4         2064.7  …
3        3137.3        4262.2         1983.4         2017.7  …
4        2964.4        4204.6         2409.7         2251.8  …

   rh_supramarginal_thickness  rh_frontalpole_thickness  \
0                       2.408                     2.629
1                       2.417                     2.640
2                       2.374                     2.601
3                       2.366                     2.639
4                       2.381                     2.555

   rh_temporalpole_thickness  rh_transversetemporal_thickness  \
0                      3.519                            2.009
1                      3.488                            2.111
```

```
2                      3.342                             2.146
3                      3.361                             2.056
4                      3.450                             2.052

   rh_insula_thickness  rh_MeanThickness_thickness  BrainSegVolNotVent.2  \
0                2.825                     2.33635               1093846
1                2.720                     2.34202               1099876
2                2.684                     2.31982               1097999
3                2.700                     2.29215               1070117
4                2.574                     2.30397               1075926

        eTIV.1  Age  dataset
0  1619602.965   85        1
1  1624755.130   85        1
2  1622609.518   86        1
3  1583854.236   87        1
4  1617375.362   89        1

[5 rows x 141 columns]
```

[2]:
```
# Separate Target Variable and Predictor Variables
X=data['Age'].values
y=data[['Left-Lateral-Ventricle', 'Left-Inf-Lat-Vent',
             'Left-Cerebellum-White-Matter', 'Left-Cerebellum-Cortex',
             'Left-Thalamus', 'Left-Caudate', 'Left-Putamen',
            'Left-Pallidum','3rd-Ventricle','4th-Ventricle',
            'Brain-Stem','Left-Hippocampus','Left-Amygdala',
            'CSF','Left-Accumbens-area','Left-VentralDC',
            'Left-vessel','Left-choroid-plexus','Right-Lateral-Ventricle',
           ␣
 ↪'Right-Inf-Lat-Vent','Right-Cerebellum-White-Matter','Right-Cerebellum-Cortex',
            'Right-Thalamus','Right-Caudate','Right-Putamen',
           ␣
 ↪'Right-Pallidum','Right-Hippocampus','Right-Amygdala','Right-Accumbens-area',
            'Right-VentralDC','Right-vessel','Right-choroid-plexus',
            '5th-Ventricle','WM-hypointensities','Left-WM-hypointensities',
           ␣
 ↪'Right-WM-hypointensities','non-WM-hypointensities','Left-non-WM-hypointensities',
           ␣
 ↪'Right-non-WM-hypointensities','Optic-Chiasm','CC_Posterior','CC_Mid_Posterior',
           ␣
 ↪'CC_Central','CC_Mid_Anterior','CC_Anterior','BrainSegVol','BrainSegVolNotVent']].
 ↪values
```

[3]:
```
from keras.callbacks import ModelCheckpoint
from keras.models import Sequential
from keras.layers import Dense
```

```
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score
from sklearn.metrics import r2_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
import math
```

[4]:
```
train = data[:4000]
test = data[4000:]
val_X = X[4000:]
val_y = y[4000:]
```

[5]:
```
NN_model = Sequential()

# The Input Layer :
NN_model.add(Dense(128, kernel_initializer='normal',input_dim = train.shape[1],␣
↪activation='relu'))

# The Hidden Layers :
NN_model.add(Dense(256, kernel_initializer='normal',activation='relu'))
NN_model.add(Dense(256, kernel_initializer='normal',activation='relu'))
NN_model.add(Dense(256, kernel_initializer='normal',activation='relu'))

# The Output Layer :
NN_model.add(Dense(1, kernel_initializer='normal',activation='linear'))

# Compile the network :
NN_model.compile(loss='mean_absolute_error', optimizer='adam',␣
↪metrics=['mean_absolute_error'])
NN_model.summary()
```

```
Model: "sequential"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               18176

 dense_1 (Dense)             (None, 256)               33024

 dense_2 (Dense)             (None, 256)               65792

 dense_3 (Dense)             (None, 256)               65792

 dense_4 (Dense)             (None, 1)                 257
```

```
=================================================================
Total params: 183,041
Trainable params: 183,041
Non-trainable params: 0
_________________________________________________________________
```

[6]:
```
checkpoint_name = 'Weights-{epoch:03d}--{val_loss:.5f}.hdf5'
checkpoint = ModelCheckpoint(checkpoint_name, monitor='val_loss', verbose = 1,␣
↪save_best_only = True, mode ='auto')
callbacks_list = [checkpoint]
```

[7]:
```
NN_model.fit(train, X, epochs=10, batch_size=32, validation_split = 0.2,␣
↪callbacks=callbacks_list, verbose=0)
```

```
Epoch 1: val_loss improved from inf to 92.89859, saving model to Weights-001--
92.89859.hdf5

Epoch 2: val_loss did not improve from 92.89859

Epoch 3: val_loss improved from 92.89859 to 76.61890, saving model to
Weights-003--76.61890.hdf5

Epoch 4: val_loss improved from 76.61890 to 21.99273, saving model to
Weights-004--21.99273.hdf5

Epoch 5: val_loss improved from 21.99273 to 12.28270, saving model to
Weights-005--12.28270.hdf5

Epoch 6: val_loss did not improve from 12.28270

Epoch 7: val_loss did not improve from 12.28270

Epoch 8: val_loss did not improve from 12.28270

Epoch 9: val_loss improved from 12.28270 to 11.59715, saving model to
Weights-009--11.59715.hdf5

Epoch 10: val_loss improved from 11.59715 to 11.50078, saving model to
Weights-010--11.50078.hdf5
```

[7]:
```
<keras.callbacks.History at 0x2202bdc82e0>
```

[10]:
```
# Load weights file of the best model :
weights_file = 'Weights-084--6.98658.hdf5' # choose the best checkpoint
NN_model.load_weights(weights_file) # load it
NN_model.compile(loss='mean_absolute_error', optimizer='adam',␣
↪metrics=['mean_absolute_error'])
```

[11]:

```
from keras import backend as K

predictions = NN_model.predict(test)
MSE = mean_squared_error(val_X, predictions)
Rsquared = r2_score(val_X, predictions)

total_cases = len(val_X) # size of validation set
avg = 0.0
SSres = 0.0
SStot = 0.0

for i in range(total_cases):
    value = val_X[i]
    predict = predictions[i]
    avg = (avg + value)/2
    SSres = SSres + (value - predict)**2
    SStot = SStot + (value - avg)**2
    #print(value, '--------- ' , predict)

Rsquared_cal = 1 - SStot/SSres
rmse = math.sqrt(MSE)

print ('NN R squared', Rsquared_cal)
print('NN RMSE = ', rmse)
```

```
NN R squared [0.86868256]
NN RMSE =  7.971411031326514
```

[ ]:

```
```