Neural Networks for Health Technology Applications

15.1.2020, Class exercise 2 - Learning to use metrics

The aim of this exercise is to learn to use the additional metrics found from scikit-learn library.

1. As in previous class exercise, create a new empty Jupyter Notebook and for the first code cell write the following codes:

```
%pylab inline
import pandas as pd
from sklearn.metrics import *
from sklearn.model_selection import train_test_split
```

These lines read all the metrics methods in sklearn.metrics library and the train_test_split method to used for splitting the datasets.

2. Read in the same dataset as in first class exercise and clean out all missing values with the following code:

```
filename = r'https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data'
df = pd.read_csv(filename,
                  index_col = None,
                 header = None,
                 na_values = '?')
df = df.dropna()
df.head()
```

3. Split the dataframe into (input) data and (output) labels. The first 12 columns contain the data and the last column contains the output value. The code simplifies the output values so that only the information “healthy” (=0) or “disease” (=1) is kept.

```
data = df.loc[:, 0:12]
labels = 1.0*(df.loc[:, 13] > 0)
print(data.shape, labels.shape)
```

```
(297, 13) (297,)
```

4. Make a simple statistics classifier, for example using the age (first column), to predict if the patient has “disease” (=1) or if the patient is “healthy” (=0)

```
predicted_output = 1.0*(data[0] > 60.0)
```

5. Calculate the confusion matrix for our simple statistics classifier by comparing the predicted output values to the true diagnostic labels

```
print(confusion_matrix(labels, predicted_output))
```

```
[[125  35]
 [ 93  44]]
```

Notice that the rows and columns are vise versa compared to the classical medical test confusion matrices. If you want transpose the matrix, do the following

```
print(confusion_matrix(labels, predicted_output).T)
```

```
[[125  93]
 [ 35  44]]
```

6. Calculate the classification report for the predicted output

```
print(classification_report(labels, predicted_output))
```

```
              precision    recall  f1-score   support

         0.0       0.57      0.78      0.66       160
         1.0       0.56      0.32      0.41       137

    accuracy                           0.57       297
   macro avg       0.57      0.55      0.53       297
weighted avg       0.57      0.57      0.54       297
```

To do next:

- Try other metrics and scoring methods found in the scikit-learn metrics.
- Try other variables found in this dataset to find out how well they could predict the cardiovascular diseases in this dataset.
- Try some linear models found from the scikit-learn to predict the outcome. Below is an example how to use LinearRegression model. Are they working better or worse than a simple statistical classifier?

```
from sklearn.linear_model import LinearRegression

reg = LinearRegression().fit(data, labels)
predicted_output = reg.predict(data) > 0.5
cm = confusion_matrix(labels, predicted_output)
print(cm)
```

- Build a simple neural network to predict the outcome and use scikit-learn metrics to see how the results are improving.

Remember to save your exercise!