Assignment 6 – Machine Learning Applications

This assignment will count towards your grade .

Summary

In this assignment, you will use and evaluate existing machine learning algorithms for automatic diagnosis tasks with imaging datasets.

Introduction

We will be performing a classification task using the scikit-learn and logistic regression. All the machine learning models implemented in scikit-learn have a very similar interface. Once you learn to work with one model you will be able to switch to another one seamlessly.
We will use two feature set derived from a fundus retina images dataset of 1,200 images.
http://www.adcis.net/en/Download-Third-Party/Messidor.html
The task will be the detection of the presence of diabetic retinopathy. The first manifestations of diabetic retinopathy are small lesions called microaneurysms (swollen capillaries), then at later stages we see exudates and hemorrhages.

In the data directory, you can find the 3 CSV files. One describing the ground truth, one containing a 256-dimensional feature set representing the vasculature as a whole, and another one a 3-dimensional feature set representing microaneurysms. If you are interested in how they are computed, you can read more in these papers:
https://link.springer.com/chapter/10.1007/978-3-319-67561-9_28 (vasculature)
http://ieeexplore.ieee.org/abstract/document/6091562/ (microaneurysms).

The following assignment needs to be presented as a Jupyter notebook. However, it is suggested to write as much as possible of code as an external module and use Jupyter notebooks for visualization and code description only.

We will be using the following shorthand for the libraries:

```
import pandas as pd
import matplotlib.pyplot as plt

import numpy as np
import skimage as sk
import skimage.io as skio
import skimage.transform as sktr
import skimage.filters as skfl
import skimage.feature as skft
import skimage.color as skcol
import skimage.exposure as skexp
import skimage.morphology as skmr
```

```
import skimage.util as skut
import skimage.measure as skme

import sklearn.model_selection as le_ms
import sklearn.decomposition as le_de
import sklearn.discriminant_analysis as le_di
import sklearn.preprocessing as le_pr
import sklearn.linear_model as le_lm
import sklearn.metrics as le_me
```

Classification with vessel features (3 points)

- Load features and ground truth data from the CSV files using pandas. Then select only samples with no retinopathy or with maximum retinopathy and their relative target labels. The target labels need to be converted to 0 (no retinopathy) and 1 (maximum retinopathy). See code below:

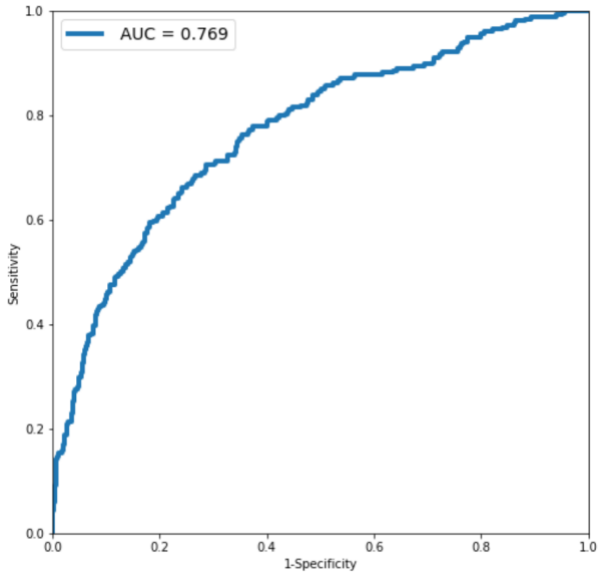
```
DATA_DIR = './data/'
#load data (originally from: http://www.adcis.net/en/Download-Third-Party/Messidor.html)
groundTruthFr = pd.read_csv( DATA_DIR + 'messidorGroundTruth.csv', index_col='ID' )
vessFeatFr = pd.read_csv( DATA_DIR + 'vesselFeatures.csv', index_col='ID' ) # vessel features DataFrame
lesionFeatFr = pd.read_csv( DATA_DIR + 'lesionFeatures.csv', index_col='ID') # lesion features DataFrame

# select only samples with no retinopathy or with maximum retinopathy
lbl = (groundTruthFr['retinopathy'] == 0) | (groundTruthFr['retinopathy'] > 2)

# Retrieve only vessel-based and lesion-based according to lbl
Xvess = vessFeatFr[lbl].values # vessel features
Xles = lesionFeatFr[lbl].values # lesion features
y = (groundTruthFr[lbl]['retinopathy'].values > 0).astype(int) # target class (0: no retinopathy, 1: retinopathy)
```

- Now, we will train and test a logistic regression classifier using the vessel-based features in the Xvess matrix. Use the code below and complete the sections starting with “TODO”:

```
# number of splits
N_SPLITS = 10
kfold = le_ms.KFold( n_splits=N_SPLITS )

# set vessel features
X = Xvess.copy()

# array containing the predictions of our classifier
#(init to -1 to make sure that probabilities have been written)
yPredictionArr = np.zeros( len(y) )
yPredictionArr[:] = -1

for train_index, test_index in kfold.split(range(len(y))):
    # train_index: list of numerical indexes identifying the training set
    # e.g. [  24   25   26 ..., 1197 1198 1199]
    # test_index: list of numerical indexes identifying the test set
    # e.g. [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23]

    #TODO:
    # split labels and feature matrix in training and testing set and store them in the following variables
    #     trainX = ?
    #     trainY = ?
    #     testX = ?
    #     testY = ?

    #TODO:
    # Scale features with their median and interquantile range
    # you can use scaler = le_pr.RobustScaler(with_centering=True, with_scaling=True, quantile_range=(25.0, 75.0))
    # remember that for a "real" test cross validation you should not use data from the test set in any way
    # even for scaling. So, you could "fit" the  scaler on trainX and then transform both trainX and testX as
    # follows:
    # scaler.fit(trainX)
    # trainX = scaler.transform(trainX)
    # testX = scaler.transform(testX)


    # TODO:
    # - Instantiate a logistic regression model with L1 regularization with a weight of 0.5.
    #   Check ( le_lm.LogisticRegression, "penalty" and "C" parameters )
    # - Train on trainX,trainY using the le_lm.LogisticRegression.fit method
    # - predict the probability for being in class 1 by using the le_lm.LogisticRegression.predict_proba method.
    #   Keep in mind that predict_proba returns a two dimensional array ( number of samples x  number of classes),
    #   so you will have to pick on
    # - store the probability in the yPredictionArr array
```

Classifier Evaluation (3 points)

- Create a function to:
  - Compute False positive ratios and true positive ratios using le_me.roc_curve
  - Compute the area under the roc curve using le_me.auc
  - Plot the ROC curve.
  - Add the AUC value in the plot and remember to label the axes!



Above is an example of what you should get. Do not expect exactly the same result!

- Manually pick a cut-off threshold and apply it to yPredictionArr. Everything above the threshold will be considered an image with retinopathy, everything below will be an image without retinopathy.
- Compute and print:
  - accuracy
  - sensitivity
  - specificity
  - confusion matrix
  - Suggestion: check the sklearn.metrics package
- **2 bonus points:** try to devise a way to automatically compute the cut-off threshold based on the ROC curve

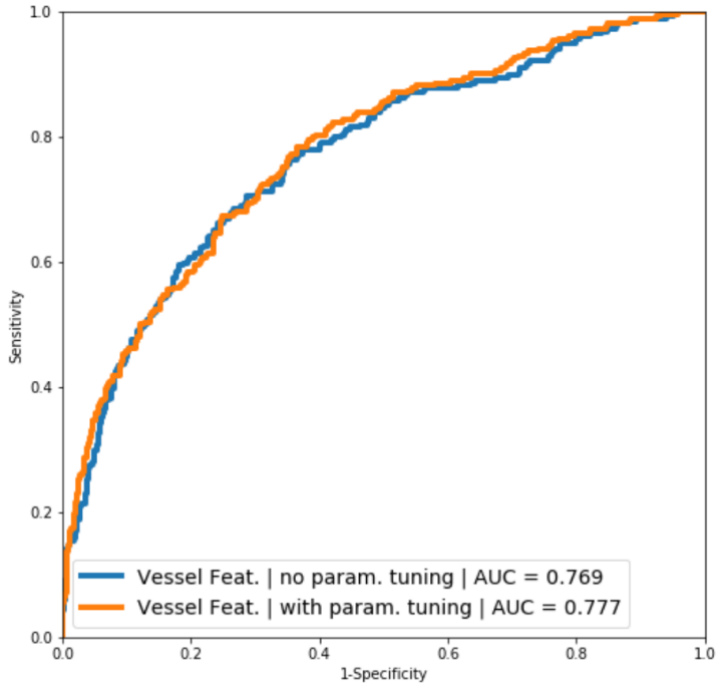
Nested Cross-Validation (2 points)

- Now, we will try to fine tune the 'C' hyperparameter of the logistic regression. This parameter represents how much it should regularize. Since, we do not have a fixed validation set, we will need to use an inner cross-validation loop in our training set to

further split it in training and validation iteratively. You can do this by leveraging the le_ms.RandomizedSearchCV function as follows:

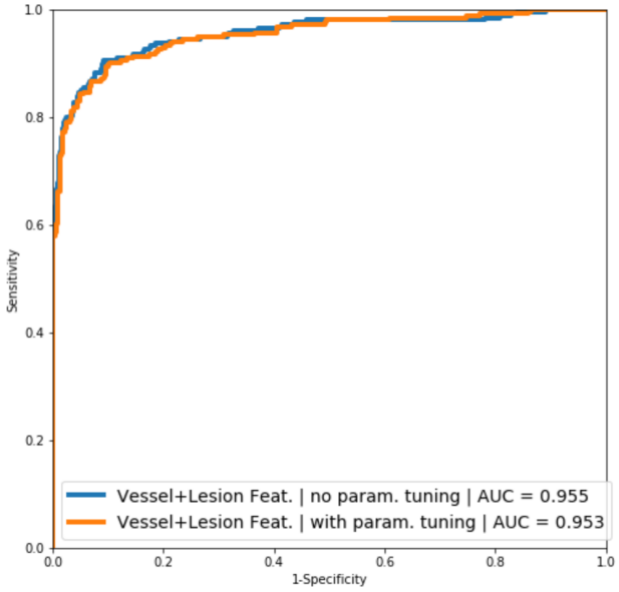
```
mod2 = #TODO: Instantiate a logistic regression model with L1 regularization with a weight of 0.5.
parameterGrid = {'C': np.linspace(0.001,1,1000) }
randSearch =le_ms.RandomizedSearchCV(mod2, parameterGrid, cv=10, n_iter=10, scoring='roc_auc', refit=True)

# find best parameters
randSearch.fit( trainX, trainY )
```

- le_ms.RandomizedSearchCV will randomly sample the parameter space train the algorithm on 9/10 of the train set, evaluate it on the remaining 1/10 (which is acting as our validation set), and store the result.
- The best set of parameters will be the ones the maximize the AUC (check the “scoring” parameter). Once the best hyperparameters have been identified, le_ms.RandomizedSearchCV will automatically retrain the model with the whole training set. The trained model is available as “randSearch.best_estimator_”.
- Now you can use randSearch.best_estimator_ to predict from the testing set as you did before (using the randSearch.best_estimator_.predict_proba function) and store the results into a new prediction array.
- Compute the ROC curve and compare it with the one obtained before (there will probably be just a very small improvement, indicating that the parameter tested does not influence that much the classification).



Adding Lesion features (2 points)

- We will try to improve the classification by adding features representing lesions.
- Combine vessel (Xvess) and lesion (Xles) features in the same matrix. Keep in mind that the resulting matrix must have the following dimensions: n_samples x n_features
- Train and test the logistic regression classifier using cross-validation with and without hyperparameter tuning, as you did before.
- Plot ROC curves:



Submission

- Now zip the folder containing your code and the notebook file as you did in assignment 01
- Upload yourname-assignment06-nb.html and yourname-assignment06-src.zip to canvas before the deadline