Mandatory Assignment 6

Web programming and data analysis (30 + 10 points/40 points)

University of Oslo - IN3110/IN4110

Fall 2019

All solutions should be stored in a directory called `assignment6` in your UiO Github repository. All functions should have docstrings that explains the types and purpose of the input, what the function does, and what it returns. Your assignment should include a `README.md` file that describes how to run your scripts.

You are expected to use Pandas for the data analysis, and Flask for the web programming.

6.0: Background (0 points)

In this assignment, you will use machine learning through scikit-learn on a classification problem with a given dataset. Additionally, you will build a web page for visualizing the data and predictions. We will be using a subset of the Pima dataset. The Pima Dataset is a publicly available dataset that contains information about 768 women of a population that is susceptible to diabetes.

The dataset, which can be found in `diabetes.csv`, contains 9 columns, where `diabetes` is the class variable that tells us if the individual tested positive (`pos`) or negative (`neg`) for diabetes. The other 8 columns are

- `pregnant`: number of times pregnant
- `glucose`: plasma glucose concentration in a glucose tolerance test
- `pressure`: diastolic blood pressure (mm Hg)
- `triceps`: triceps skin fold thickness (mm)
- `insulin`: 2-hour serum insulin (μU/ml)
- `mass`: body mass index (kg/m^2)
- `pedigree`: diabetes pedigree function
- `age`: age (years)

These will be used as features in order to predict whether or not the individual has diabetes.

6.1: Handling the data (5 points)

Build a Python script that reads the data from the file `diabetes.csv` using pandas. Note that some of the samples in the dataset are missing some values (`NaN`). Remove these samples. Then split the data into a training set and a validation set. Let the training set be 80% of the total samples. It can be a good idea to keep the proportion of positive samples the same in the training and validation sets.

To get a sense of how the data relates to the `diabetes` class, make a function that allows you to choose two dimensions of the feature space and creates a scatter plot coloring the two classes differently. Run the function with several different pairs of features. Design the script as a module such that it can be imported in the next tasks.

Name of file: `data.py`

6.2: Fitting a machine learning model (8 points)

Create a Python script containing a function, `fit`, for fitting the data using various `sklearn` functions. The function should be able to change which features used as well as switch between different classifiers. The function should return the trained classifier.

You are free to choose which `sklearn` classifiers to use, but you should be able to handle at least two different methods. Run the function with several different methods and feature subsets (that is, use fewer than all 8 features) on the training set and compare their accuracy on the validation set using `sklearn.metrics`. Optionally, you can use k-fold cross validation to estimate the accuracy.

Name of file: `fitting.py`

6.3: Visualizing the classifier (5 points)

Make a function that can visualize the classifier when the chosen feature subset has only 2 features. Create a similar scatter plot of the data points as in task 6.1, but add colored areas that represent which class the classifier predicts in that area. Color the areas in a slightly lighter color compared to the data points of the same class so that you can see them.

Name of file: `visualize.py`

6.4: Visualization through a web app (6 points)

Build a Flask app which uses your script from 6.3 to generate a plot of the classifier and displays it on the web page. Display also the classifier accuracy on the validation and training set.

Name of file: `web_visualization.py`

6.5: Interactive visualization (10 points)

Modify your solution in 6.4 so that the visitor of the web page can change the features used, classifier and optionally some hyperparameters (arguments to the classifier) of the plot using checkboxes, drop down menus, radio buttons or whatever reasonable option you prefer.

Display the accuracy of the fitted classifier on the validation and training set. If the user chose only 2 features, also generate and display the plot from 6.3/6.4.

Try to make things user friendly. You are not required to make the plot 'fancy' or dynamic in the sense of making the image zoomable, pannable or something else - it is sufficient that the user can change the parameters mentioned. The idea is to generate a static image, and then have the website load it.

6.6: Documentation and help pages (6 points)

Extend your web app with a help-page, and add a link to this on your plot page. The help page should display help for your methods in `data.py`, `fitting.py` and `visualize.py` generated automatically from docstrings using your favorite tool for this. Options here include pydoc or Sphinx.

Clarifications

- If a later exercise asks you to add functionality to a previous exercise, you do not have to submit both the old version and the updated version - it is fine to just submit the updated version.