Last updated on Monday, April 27, 2020

2020 CAB320 - Assignment Two (Machine Learning)

Key information

- Code and report submission due on **Sunday 31 May, 23.59pm**
- Use **Blackboard** to submit your work
- Group size: **up to three people per submission**

Overview

- You are provided with a dataset of tumour measurements
- The aim of this assignment is to build four different types of classifiers and evaluate their performance. The classification task is to predict whether a tumour is malignant (M) or benign (B).
- Using the *sklearn*, you will build the following classifiers
    - a *nearest neighbours* classifier
    - a *decision tree* classifier
    - a *support vector machine* classifier
    - a *neural network* classifier

  Although you can implement all classifiers with the *sklearn* library, you are allowed to use *Keras* to build the neural network classifier if you prefer.

Dataset

The records are stored in a text file named “medical_records.data”. Each row corresponds to a patient record. The diagnosis is the attribute predicted. In this dataset, the diagnosis is the second field and is either *B* (benign) or *M* (malignant). There are 32 attributes in total (ID, diagnosis, and 30 real-valued input features).

For the purpose of this assignment, you can ignore the semantic details of the attributes given in the section below. This information is only provided for the sake of completeness.

Attribute Information

1) ID number

2) Diagnosis (M = malignant, B = benign)

3-32)

Ten real-valued features are computed for each cell nucleus:

- a) radius (mean of distances from center to points on the perimeter)
- b) texture (standard deviation of gray-scale values)
- c) perimeter
- d) area
- e) smoothness (local variation in radius lengths)
- f) compactness (perimeter^2 / area - 1.0)
- g) concavity (severity of concave portions of the contour)
- h) concave points (number of concave portions of the contour)
- i) symmetry
- j) fractal dimension ("coastline approximation" - 1)

The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius. All feature values are recorded with four significant digits.

Your tasks

Preprocessing

Complete the function *prepare_dataset* that loads the data records from the text file, and converts the information to numpy arrays.

Build Classifiers

Complete the *build_*_classifier* functions. There are four functions of this type to implement in the provided python file (nearest neighbours, decision trees, neural networks and support vector machines).

These classifiers have hyperparameters that affect the capacity/complexity of the classifier, you should use *cross-validation* to estimate the best value of one of these hyperparameters for each type of classifier. In this assignment, for the sake of time, we only consider one hyperparameter per classifier. Namely,

- nearest neighbours → number of neighbours
- decision trees → maximum depth of the tree or minimum size of a leaf
- support vector machine → parameter *C*
- neural networks → number of neurons in the hidden layers

You should report the prediction errors including on your *test_data.* These errors are best reported in tables and figures. You are encouraged to use all the available functions of the *sklearn* and *tensorflow/keras* libraries.

Your experiments should be repeatable. That is, you should write and document clearly your functions to allow the markers to repeat your experiments. **A marker should only have to toggle the comment status of some commented out lines in the main function of your submitted file to repeat your experiments.**

Deliverables

You should submit via Blackboard two files

A **report in pdf format strictly limited to 4 pages in total** (be concise!)

- Explain the methodology you use for the hyperparameter search
- Describe and compare the performance of your classifiers
- Use tables and figures

Your Python file **myQuack.py**

Mark Distribution

- **Report**: 5 marks
    - Structure (sections, page numbers), grammar, no typos.
    - Clarity of explanations.
    - Figures and tables (use for explanations and to report performance).
- **Code quality**: 5 marks
    - Readability, meaningful variable names.
    - Proper use of Python constructs like dictionaries and list comprehension.
    - Proper header comments in classes and functions.
    - Function parameter documentation.
    - In-line comments.
- **Functions**: 30 marks
    - Preprocessing - 4 marks
    - Classifiers - 4 marks per classifier (in total 4*4)
    - Other functions – 10 marks

Final Remarks

- Do not underestimate the workload. Start early. You are strongly encouraged to ask questions during the zoom sessions.
- You are encourage to use Slack (channels are monitored by the tutors).

Marking Guide and Criteria Table

- **Report**: 5 marks
    - Structure (sections, page numbers), grammar, no typos.
    - Clarity of explanations.
    - Figures and tables (use for explanations and to report performance).

Levels of Achievement

5 Marks	**4 Marks**	**3 Marks**	**2 Marks**	**1 Mark**
Report written at the highest professional standard with respect to spelling, grammar, formatting, structure, and language terminology.	Report is very-well written and understandable throughout, with only a few insignificant presentation errors. Methodology, experiments and recommendations are clear.	The report is generally well-written and understandable but with a few small presentation errors that make one of two points unclear. Clear figures and tables.	Large parts of the report are poorly-written, making many parts difficult to understand. Use of sections with proper section titles. No figures or tables.	The entire report is poorly-written and/or incomplete and/or impossible to understand. The report is in pdf format.

To get “i Marks”, the report needs to satisfy <u>all the positive items and none of the negative items</u> of the columns “j Marks” for all j<i.

- **Code quality**: 5 marks
    - Readability, meaningful variable names.
    - Proper use of Python constructs like numpy arrays, dictionaries and list comprehension when applicable.
    - Proper header comments in all classes and functions.
    - Function parameter documentation.
    - In-line comments.

Levels of Achievement

5 Marks	**4 Marks**	**3 Marks**	**2 Marks**	**1 Mark**
Code is generic. Minimal changes would be needed to run same experiments on a different dataset.	Proper use of numpy array operations. Avoid unnecessary loops. Useful in-line comments. Code structured so that it is straightforward to repeat the experiments	No magic numbers (that is, all numerical constants have been assigned to variables). Appropriate use of auxiliary functions. Each function parameter documented (including type and shape of its parameters)	Header comments with instructions on how to run the code to repeat the experiments.	Code functional but looks like a high-entropy spaghetti plate

To get "i Marks", the report needs to satisfy <u>all the positive items and none of the negative items</u> of the columns "j Marks" for all j<i.

- **Functions**: **30** marks
    - *prepare_dataset* implemented correctly (4 marks)
        - Returns numpy arrays X, y
        - Data read correctly from file
        - X, y are of the correct dimensions/shape and contain values of the correct type (integers, floats, no strings)
    - Classifier functions (4 classifiers: 4 * 4 marks = 20 marks)
        - Correct selection of sklearn functions for the requested type of classifiers. [1 mark]
        - Hyperparameter selection (one hyperparameter tuning with cross validation search). [2 marks]
        - Proper model training with function returning a valid estimator/classifier object. [1 mark]
    - Functions to repeat all the experiments. [10 marks]
        - For each type of classifier, a function calls the build classifier and evaluate the classifier on a test set.
        - Proper creation and use of a test set.
        - Plotting and/or tabling of relevant training, validation and test results.