Imperial College London

Department of Mathematics
MATH96007 - MATH97019 - MATH97097
Methods for Data Science
Years 3/4/5

Coursework 1 – Supervised learning

Submission deadline: Wednesday, 23 February 2022 at 4 pm

Please read carefully the following instructions.

The goal of this coursework is to analyse two data sets using several tools and algorithms introduced in the lectures, which you have also studied in detail through the weekly Python notebooks containing the computational tasks.

You will solve the tasks in this coursework using Python. You are allowed to use Python code that you will have developed in your coding tasks. You are also allowed to use any other basic mathematical functions contained in numpy that you use to write your own code. Importantly, unless explicitly stated, you are **not** allowed to use for your solutions any model-level Python packages (e.g., sklearn, statsmodels, etc) or ready-made code found online.

Submission

The submission of your coursework will consist of two items:

- a **Jupyter notebook (file format: ipynb)** with all your tasks clearly labelled. You should use the template called *SurnameCID_CW1.ipynb,* which is provided on Blackboard.

  The notebook should contain the cells with your code and their output, plus some brief text explaining your calculations, choices, mathematical reasoning, and discussion of results. *(Note: Before submitting you must run the notebook, and the outputs of the cells printed.)* You may produce your notebook with Google Colab, but you can also develop your Jupyter notebook through the Anaconda environment (or any local Python environment) installed on your computer.

- Once you have executed all cells in your notebook and their outputs are printed, you should also save the notebook as an **html file**, which you will also submit.

Submission instructions

The submission will be done **online via Turnitin on Blackboard**.

The deadline is **Wednesday, 23 February 2022 at 4 pm**.

*You will upload **two documents** to Blackboard, wrapped into a **single zip file**:*

1) *Your Jupyter notebook as an **ipynb file**.*
2) *Your notebook exported as an **html file**.*

You are also required to comply with these specific requirements:

- *Name your **zip file** as 'SurnameCID_CW!.zip', e.g. Smith123456_CW!.zip. Do not submit multiple files.*
- *Your ipynb file must produce all plots that appear in your html file, i.e., make sure you have run all cells in the notebook before exporting the html.*
- *The notebook should have clear headings to indicate the answers to each question, e.g. 'Task 1.1'.*

Note about online submissions:

- *There are known issues with particular browsers (or settings with cookies or popup blockers) when submitting to Turnitin. If the submission 'hangs', please try another browser.*
- *You should also check that your files are not empty or corrupted after submission.*

- ***To avoid last minute problems*** *with your online submission, we recommend that you upload versions of your coursework early, before the deadline. You will be able to update your coursework until the deadline, but having these early versions provides you with some safety back up.*

Needless to say, projects must be your own work: *You may discuss the analysis with your colleagues but the code, writing, figures and analysis must be your own. The Department may use code profiling and tools such as Turnitin to check for plagiarism, as plagiarism cannot be tolerated.*

Marks

The coursework is worth **40% of your total mark for the course.**
This coursework contains a ***mastery component*** **for MSc and 4th year MSci students**.

Some general guidance about writing your solutions and marking scheme:

Coursework tasks are different from exams. Sometimes they can be more open-ended and may require going beyond what we have covered explicitly in lectures. In some parts of the tasks, initiative and creativity will be important, as is the ability to pull together the mathematical content of the course, drawing links between subjects and methods, and backing up your analysis with relevant computations that you will need to justify.

To gain the marks for each of the Tasks you are required to:

(1) complete the task as described;
(2) comment any code so that we can understand each step;
(3) provide a brief written introduction to the task explaining what you did and why you did it;
(4) provide appropriate, relevant, clearly labelled figures documenting and summarising your findings;
(5) provide an explanation of your findings in mathematical terms based on your own computations and analysis and linking the outcomes to concepts presented in class or in the literature;
(6) consider summarising your results of different methods and options with a judicious use of summary tables of figures.

The quality of presentation and communication is very important, so use good combinations of tables and figures to present your results, as needed.

Explanation and understanding of the mathematical concepts are crucial.

Marks will be reserved and allocated for: presentation; quality of code; clarity of arguments; explanation of choices made and alternatives considered; mathematical interpretation of the results obtained; as well as additional relevant work that shows initiative and understanding beyond the task stated in the coursework.

Code: Competent Python code is expected. As stated above, you are allowed to use your own code and the code developed in the coding tasks in the course. Copy-pasting code from other sources (e.g., online) is **not** allowed. You are expected to develop your own code for the specific tasks starting from your Python notebooks containing the coding tasks. You are not allowed to use Python packages like sklearn, statsmodels, etc unless explicitly stated.

Note that the mere addition of extra calculations (or ready-made 'pipelines') that are unrelated to the task without a clear explanation and justification of your rationale will not be beneficial in itself and, in fact, can also be detrimental if it reveals lack of understanding of the required task.

Coursework

In this coursework, you will work with two different data sets of high-dimensional samples:

- **a data set of chemicals evaluated for toxicity**
- **a medical data set characterising tumour malignancy in cancer**

You will perform a **regression task** with the former, and a **binary classification task** with the latter.

Task 1: Regression (45 marks)

Data set: Your first task deals with a *modified* **chemistry data set** that we have prepared based on molecular descriptors for an array of chemicals, each associated with a level of toxicity towards fish as measured in experiments with *Pimephales promelas*. (If you click the link you might understand why we thought it better to stick with the Greek name for the genus of this fish.) Each sample in the dataset (rows) corresponds to a chemical substance characterised by 10 features (molecular descriptors, columns). We will consider the toxicity level (column 'LC50') as the target variable to regress, while the other 10 variables are our predictors.

- This modified chemistry data set is made available to you on Blackboard as `chemistry_samples.csv`.
- We also provide on Blackboard a test set in the file `chemistry_test.csv`.

*Important: The test set should **not** be used in any learning, either parameter training or hyper-parameter tuning of the models. In other words, the test set should be put aside and reserved as **unseen**, and only be used a posteriori to support your conclusions and to evaluate the out-of-sample performance of your models.*

Questions:

1.1 Linear regression (10 marks)

1.1.1 - Use the data set `chemistry_samples.csv` to obtain a linear regression model to predict the *toxicity factor LC50* as your target variable using all the other features as predictors. Report the inferred values of the model parameters and the in-sample R^2 score for the data set.

1.1.2 - Apply the model to the test (unseen) data (`chemistry_test.csv`) to predict the target variable, and compute the out-of-sample R^2 score on this test set. Compare the out-of-sample and the in-sample R^2 score, and explain your findings.

1.2 Ridge regression (20 marks)

1.2.1 - Use the data set `chemistry_samples.csv` and repeat task 1.1.1 employing Ridge regression with 5-fold cross-validation to tune the penalty hyper-parameter of the model. Using the average mean squared errors (MSE) over all folds, demonstrate with plots how you scan the penalty hyper-parameter to find its optimal value. Report the optimal value for the penalty parameter and the performance of the model in terms of MSE. Using some of your computations, explain the trend of bias, variance and MSE as a function of the penalty hyper-parameter.

1.2.2 - Fix the penalty hyper-parameter to the optimal value found in 1.2.1 and retrain the model on the entire data set `chemistry_samples.csv`. Obtain the in-sample R^2 score when applied to `chemistry_samples.csv`, and compare it to the out-of-sample R^2 score on the test set `chemistry_test.csv`. Use some of your computations to discuss the differences between ridge regression and linear regression (Task 1.1).

1.3 Relaxation of Lasso regression (15 marks)

1.3.1 - In this task, you will implement a relaxation of the Lasso optimisation that can be solved using gradient descent. Consider a *smooth* version of Lasso in which the penalty term of the cost function is approximated through smooth functions L_c (Huber functions) so that it is differentiable. The cost function to be optimised is given by:

$$L_{\text{LASSO}}(\boldsymbol{\beta}) = ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||^2 + \lambda \sum_{i}^{p} L_c(\beta_i)$$

$$L_c(\beta) = \begin{cases} \frac{1}{2}\beta^2 & \text{for } |\beta| \leq c, \\ c(|\beta| - \frac{1}{2}c), & \text{otherwise.} \end{cases}$$

where p is the number of predictors, λ is the penalty hyper-parameter, and c regulates the `sharpness' of the Huber functions. Here we will fix c=0.001. A skeleton for the code of this task is provided within the *SurnameCID_CW1.ipynb* template on Blackboard.

1.3.2 - Use the data set `chemistry_samples.csv` and employ this relaxed Lasso-Huber regression with a 5-fold cross-validation (with the same folds as in 1.2.1) to conduct a grid search to find the optimal penalty hyper-parameter λ. Report the in-sample and out-of-sample performance of the model with the optimal penalty hyper-parameter, as in 1.2.2, using the R^2 score.

1.3.2 - Discuss the differences in the regression coefficients obtained through Lasso (Task 1.3) and Ridge (Task 1.2) regressions.

Task 2: Classification (55 marks)

Data set: Your second task deals with the classification of **breast tumour samples** as 'benign' or 'malignant' based on 30 features. The column 'DIAGNOSIS' corresponds to the tumour classification where 'B' stands for 'benign' and 'M' for 'malignant'. The other 30 columns correspond to the features.

- The data set is available on Blackboard under file `tumour_samples.csv`.
- The test set is in the file `tumour_test.csv`.
- We also provide a balanced data set `tumour_samples_bal.csv` which is used in Tasks 2.3.3 and 3.2.2.

*Important: The test set should **not** be used in any learning, either of parameter training or hyper-parameter tuning of the models. In other words, the test set should be reserved as **unseen**, and only be used a posteriori to support your conclusions and to evaluate the out-of-sample performance of your models.*

Questions:

2.1 kNN classifier (10 marks)

2.1.1 - Train a k-Nearest Neighbour (kNN) classifier on the data set (`tumour_samples.csv`). Demonstrate that you have used a grid search with 5-fold cross-validation to find an optimal value of the hyper-parameter *k*.

2.1.2 - As in 1.2.2., fix the optimal *k* and retrain the model on the entire data set `tumour_samples.csv`. Use *accuracy* to compare the performance of your optimised classifier on the data set `tumour_samples.csv` to the performance on the test data set `tumour_test.csv`.

2.2 Random forest (20 marks)

2.2.1 - Train a random forest classifier on the data set `tumour_samples.csv` employing ***cross-entropy*** as your information criterion for the splits in the decision trees. Use the same 5-fold cross-validation subsets as in 2.1.1 to explore and optimise over suitable ranges the following hyper-parameters: (i) number of decision trees; (ii) depth of trees. Use *accuracy* as the measure of performance for this hyper-parameter optimisation.

2.2.2 - Compare the performance of your optimal random forest classifier on the data set `tumour_samples.csv` to the performance on the test data `tumour_test.csv` using different measures computed from the *confusion matrix,* in particular commenting on accuracy, recall and F1-score.

2.3 Support vector machine (SVM) (25 marks)

2.3.1 - Train a soft margin linear SVM classifier on the data set `tumour_samples.csv` using the same 5-fold cross-validation subsets as in 2.1.1 to optimise the hardness hyper-parameter that regulates the boundary violation penalty. Use *accuracy* as a measure of performance for this hyper-parameter optimisation. Display the accuracy of the SVM classifiers as the hardness hyper-parameter is varied, and discuss the limits of low hardness and high hardness.

2.3.2 - Evaluate the performance of the SVM classifiers obtained as the hardness hyper-parameter is varied by applying each of them to the *test data* `tumour_test.csv`. Represent your results using a *receiver operating characteristic (ROC) curve*. Use the ROC curve to discuss your choice of the optimal hardness hyper-parameter obtained in 2.3.1.

2.3.3 - Repeat tasks 2.3.1 and 2.3.2 but now training on the balanced data set `tumour_samples_bal.csv`. Using ROC curves (or other measures), compare and discuss the performance of SVM classifiers learnt from the (unbalanced) data set `tumour_samples.csv` obtained in Tasks 2.3.1 and 2.3.2 versus SVM classifiers learnt from the balanced data set `tumour_samples_bal.csv` obtained in Task 2.3.3.

Task 3: Mastery component (25 marks)

This task is to be completed by MSci (4th year) and MSc students.

3.1 Logistic regression and bagging (15 marks)

3.1.1 - Train a logistic regression classifier on the data set (`tumour_samples.csv`) with gradient descent for 5000 iterations, using a learning rate of 0.005. Measure the accuracy of the classifier using the decision threshold 0.5.

3.1.2 - Implement code that applies *bagging* to the training of the logistic regression classifier demonstrating that you have used a grid search with 5-fold cross-validation to choose the optimal number of bootstrap samples. Use *accuracy* as the measure of performance for the grid search, setting the decision threshold to 0.5, the gradient descent iterations to 5000, and the learning rate to 0.005, as in task in 3.1.1.

3.1.3 - Using the test data set `tumour_test.csv`, discuss the accuracy of the model achieved through bagging in 3.1.2 compared to the accuracy obtained in 3.1.1. To calculate the accuracy, use the decision threshold 0.5 as in 3.1.1.

3.2 Kernelised SVM classifier (10 marks)

3.2.1 - Starting from the code of the linear SVM (task 2.3.1), implement a soft margin kernelised SVM classifier with a Radial Basis Function (RBF) kernel.

3.2.2 - Fix the hardness hyper-parameter to the optimal value found in 2.3.3 and train the soft margin kernelised SVM classifier with RBF kernel on the data set `tumour_samples_bal.csv`, using the same 5-fold cross-validation subsets as in 2.3.3. Demonstrate that you have used a grid search with 5-fold cross-validation to find the optimal hyper-parameter of the RBF kernel. Use *accuracy* as a measure of performance for this hyper-parameter optimisation. Using appropriate measures, compare the results of the linear SVM in 2.3.3 to the results of the kernel SVM obtained here.