Imperial College London

Department of Mathematics
MATH60026 - MATH70026
Methods for Data Science
Years 3/4/5

Coursework - General instructions - please read carefully

The goal of this coursework is to analyse datasets 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, hence competent Python code is expected.

You are allowed to use:

- Python code that you have developed in your coding tasks;
- any other basic mathematical functions contained in `numpy`;
- `pandas` to build and clean up data tables;
- `tqdm` as a utility to visually track the progress of for-loops.

You are *not* allowed to use:

- any model-level or automatic differentiation Python packages (*e.g.*, scikit-learn, statsmodels, PyTorch, JAX, Tensorflow/Keras, etc);
- ready-made code found anywhere online;
- conversational AI tools such as ChatGPT, Microsoft Bing, GitHub Copilot etc. to generate code and written answers.

Needless to say, your submission must be your own individual work: *You may discuss the analysis with your colleagues but code, written answers, figures and analysis must be written independently on your own. The Department uses code profiling and tools such as Turnitin to* ***check for plagiarism of any sort****. Plagiarism is a major form of academic misconduct.*

Marks

This coursework is worth **40% of your total mark for the course.**

Mastery component: The mastery component amounts to 20% of this CW: two tasks (**1.1.3** and **1.2.3**) have one version for 3rd year BSc students only and one version for MSc and 4th year MSci students only (Mastery component) - pay attention to this and choose the right option in each of those tasks! You should only answer the version that applies to your case.

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, 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 a relevant 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 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.

*When you comment on your results, **precise pointers to your code and your plots must be provided**: generic, boiler-plate comments on a method that are not based on the specific problem and the analysis carried out by you will receive zero marks.*

Similarly, 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 to the mark if it reveals lack of understanding of the required task.

Submission

For the submission of your coursework, you need to save **two documents**:

- a **Jupyter notebook (file format: ipynb)** with all your tasks clearly labelled. You should use the template notebook called *CID_Coursework1.ipynb* provided on Blackboard (folder 'Coursework/Coursework 1'). The notebook should have clear headings to indicate the answers to each question, *e.g.* 'Task 1.1'.

  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. *(Important: Before submitting you must run the notebook, and the outputs of the cells must be printed. Having all cells executed sequentially is a way of showing to us that your notebook correctly produces the displayed output. The absence of this output will be penalised).* You can use Google Colab or 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 save the notebook as an **html file** (with name *CID_Coursework1.html*). Your ipynb file must produce all the output that appears in your html file, *i.e.*, make sure you have run all cells in the notebook before exporting the html.

The submission is done **online via Blackboard,** using the drop boxes inside the folder 'Coursework' on Blackboard.

The deadline is Friday, 23 February 2024 at 1 pm.

The submission of your coursework must consist of **two items,** to upload **separately**:

1) *A **single zip folder** containing your Jupyter notebook as an **ipynb file** and your notebook exported as an **html file**. Name your zip folder 'CID_Coursework1.zip', where CID is your student CID, e.g. 123456_Coursework1.zip.*
2) *The html file, named CID_Coursework1.html, which will be used for the plagiarism check.*

The submission should consist *only* of these 2 items - **Do not submit multiple files.**

Important note: Make sure you submit to the right drop box on Blackboard

- If you are a 3rd year BSc student: use the 'Coursework Drop Boxes' inside the folder 'Coursework'. The html file must be uploaded to 'Coursework 1 Drop Box Spring 24 - HTML', the zip folder must be uploaded to 'Coursework 1 Drop Box Spring 24 - ZIP'.

- If you are a 4th year MSci or MSc student: use the ‘Mastery Coursework Drop Boxes’ inside the folder ‘Coursework’. The html file must be uploaded to ‘Coursework 1 Drop Box Spring 24 - Mastery HTML’, the zip file to ‘Coursework 1 Drop Box Spring 24 - Mastery ZIP’.

Any mistake in the submission folder will cause a delay in the release of your mark. **Do not put your name on the files you submit** (only the CID), because the marking must be carried out preserving your anonymity.

Notes 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 on**, before the deadline. You will be able to update your coursework until the deadline, but having this early version provides you with some safety backup. For the same reason, keep **backups of your work**, e.g. save regularly your notebook with its outputs as an .html file, which can be useful if something unpredicted happens just before the deadline.*
- *If you have any issue with the submission, or you realise you have submitted your work to the wrong drop box, please contact directly the UGMathsOffice at maths-student-office@imperial.ac.uk or your MSc programme administrator, in such a way that they can help you solve the issue.*
- *If you need an extension, or happen for any reason to submit your work late, please make a request for mitigating circumstances directly on ZINC.*

  *For these last two points, **do not contact us - we, as lecturers, are not able to grant extensions nor to make changes in the submission folders! We only get to see anonymised submissions.***

Coursework 1 – Supervised learning

Submission deadline: Friday, 23 February 2024 at 1 pm

Coursework

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

- **an engineering dataset measuring different properties of graphene-based supercapacitors**
- **a medical dataset used for the diagnosis of brain cancer**

You will perform a **regression task** with the former, and a **classification task** with the latter. All datasets are made available inside the folder 'Coursework/Coursework 1/Data' on Blackboard.

Task 1: Regression (50 marks)

Dataset: Your first task deals with an **engineering dataset**. It contains the design properties (our data features or descriptors) of graphene-based electrodes, and each set of features is associated to a resultant electrical capacity. Graphene-based electrodes are a promising alternative for electricity storage, but we still lack understanding of what contributes to its capacitive properties. Each of the 558 samples in the dataset (rows) corresponds to a graphene-based electrode, described by 12 design properties (like surface area, electrolyte concentration, etc — each of them measured with appropriate units of measure, see the columns). We will consider the resultant electrical capacity (column 'Capacitance (μF/cm²)') as the target variable to regress, while the other 12 variables are our features.

- This dataset is made available on Blackboard as `nanoelectrodes_capacitance_samples.csv.`
- We also provide on Blackboard a test set in the file `nanoelectrodes_capacitance_test.csv.`

Important: *The test set should* ***not*** *be used in any learning, either parameter training or hyperparameter tuning of the models. The test set should be put aside and only be used a posteriori to support your conclusions and to evaluate the* ***out-of-sample*** *performance of your models. Only the dataset* `nanoelectrodes_capacitance_samples.csv` *should be used for the cross-validation tasks, where you will be in charge of choosing an appropriate set of hyperparameter values (at least 5 values per hyperparameter) to scan. If you wish to standardise the dataset, please use the convention by which we use mean and standard deviation of the training set to standardise the test set, as discussed in the lecture.*

Questions:

1.1 Random Forest (20 marks)

1.1.1 (5 marks) - Train a Decision Tree regression model to predict the electrical capacity from the 12 features. Use the following hyperparameters: `max_depth=10, min_samples_leaf=10`. Evaluate the generalisation power of the Decision Tree on the test set using the Mean Squared Error (MSE) and R² score as your metrics of performance. Discuss your findings.

1.1.2 (5 marks) - Perform bagging and feature bagging starting from the Decision Tree structure of task **1.1.1** to construct a Random Forest regression model to predict the electrical capacity. Use the standard rule-of-thumb to determine the number of features for feature bagging, while you need to find the optimal value of B (the number of trees) by 5-fold cross-validation using the MSE of the Random Forest as performance metric. Using the MSE and R² score, evaluate the generalisation power of the Random Forest on the test set and compare it to the one of a single Decision Tree (task **1.1.1**) and to the one of the ensemble of B Decision Trees alone (without feature bagging). Discuss your findings.

1.1.3 (10 marks, BSc students only) - Search for optimal values of `max_depth`, `min_samples_leaf` (keeping B fixed) for the Random Forest of task **1.1.2** by 5-fold cross-validation using the MSE as metric**.** Evaluate the performance of the Random Forest with these optimal hyperparameters on the test set using the MSE and R² score, and compare it to the results from task **1.1.2**. Next, use the Out-Of-Bag (OOB) samples from bagging to estimate the importance factors of each feature, using the MSE as performance metric. Express them as a percentage of the most important feature, plot them and draw conclusions about which data features contribute the most to the prediction of electrical capacity.

1.1.3 (10 marks, MSc/4th-year students only) - As introduced in the lectures, build a Gradient Boosted Decision Tree (GBDT) regression model trained over 50 iterations (weak learners) with a learning rate equal to 0.4 and optimise the GBDT model over the hyperparameters `max_depth` and `min_samples_leaf` of the weak learners via 5-fold cross validation, with the MSE as performance metric. Evaluate the performance of the GBDT model on the test set using the MSE and R² score and compare your results to the performance of the models from tasks **1.1.1** and **1.1.2**.

1.2 Multi-layer Perceptron (30 marks)

1.2.1 (10 marks) - ***Using NumPy alone*** like in your Week 5 notebook (*i.e.*, without using TensorFlow/Keras, PyTorch or equivalent libraries), implement a Multi-Layer Perceptron (MLP) to perform regression according to the following architecture description:

Architecture of the network: Your network should have an input layer, **2** hidden layers (with **50** neurons each), followed by the output layer with one neuron (the outcome variable to predict). For both hidden layers, apply the following activation function:

$$\sigma(x) = tan^{-1}(x) ln(|x| + 1)$$

Use mini-batch stochastic gradient descent (SGD) as your optimisation method and the MSE as your loss function.

Train the MLP on the training set using mini-batches of 8 data points for 300 epochs and setting the learning rate to 5×10^{-5}. Plot the loss as a function of the number of epochs for both the training and test sets to demonstrate convergence. Evaluate the generalisation power of the trained MLP on the test set by measuring the MSE and R² score.

1.2.2 (10 marks) - Use a different optimiser to train the MLP from task **1.2.1**: use SGD with momentum. Use mini-batches of 8 data points for 300 epochs, set the learning rate to 5×10^{-5} and the momentum parameter β to 0.4. Evaluate the MSE on the training and test sets, and discuss the effect of momentum on model training and performance compared with the MLP from task **1.2.1**. Compare the model performance on the test set achieved here to the MLP from task **1.2.1** and to the Random Forest from task **1.1.2**, drawing a conclusion on which model performs best.

1.2.3 (10 marks, BSc students only) - Compare the MLP to another simpler approach to include some non-linearities in the regression task under consideration. Perform what is called linear regression with quadratic basis functions: extend your set of 12 features to a set containing also the quadratic terms, *i.e.*, the squares of features and the products of different features. The extended set of features $\mathbf{X}'$ is given by:

$$\boldsymbol{X}' = [\underbrace{X_1, X_2, \ldots, X_p}_{\text{original features}}, \underbrace{X_1^2, X_1X_2, \ldots, X_1X_p, X_2^2, X_2X_3, \ldots X_2X_p, \ldots X_p^2}_{\text{quadratic terms}}]$$

Use this extended set of features $\mathbf{X}'$ to implement Ridge linear regression to predict the electrical capacity. First, perform Ridge regression with $\lambda = 0.0001$, $\lambda = 1$, $\lambda = 1000$ and plot the distribution of the inferred coefficients for these 3 values of λ. Explain and justify the trends you see. Next, find the optimal penalty λ using 5-fold cross-validation. Next, compare the performance of this model (given by the MSE and R² score on the test set) to linear regression on the 12 original features: here, implement linear regression with and without Ridge penalty, finding the optimal Ridge penalty λ by 5-fold cross-validation.

1.2.3 (10 marks, MSc/4th year students only) - Nesterov's Accelerated Gradient (NAG) is closely related to SGD with momentum. In this research article by Sutskever et al. 2015 (also available on Blackboard), it was found that it can outperform SGD with momentum, when used in conjunction with well-designed parameter initialisations and a schedule for the momentum parameter. Repeat this comparison in the case of your MLP regression model (task **1.2.1**). Specifically, implement the NAG introduced in section 2 of Sutskever et al. 2015; implement it with the iteration-dependent schedule for the momentum parameter here called μ (section 3), and use the sparse initialisation (section 3.1) that the authors used for their experiments with deep autoencoders. To keep your experiment focussed on the schedule for μ, set the learning rate to 5×10^{-6}. Choose a minibatch size of 8. Using the MSE on the training set to evaluate performance, draw conclusions on the performance of NAG in this context, comparing it to that of SGD with momentum (task **1.2.2**).

Task 2: Classification (50 marks)

Dataset: Your second task involves working with a dataset designed for the diagnosis of brain cancer that uses characteristics of a scanned lump, including its density, diameter, and the specific region in the brain where it is located. The brain cancer diagnosis corresponds to a classification task, since the characteristics detected through imaging allow one to predict a benign tumour ('Class=0') or malignant types glioma ('Class=1') and meningioma ('Class=2') . The other 11 columns correspond to the data features to use for training the classifier.

- The dataset is available on Blackboard in the file `brain_cancer_samples.csv`.
- The test set is in the file `brain_cancer_test.csv`.

Important: *The test set should **not** be used in any learning, either parameter training or hyperparameter tuning of the models. The test set should be put aside and only be used a posteriori to support your conclusions and to evaluate the **out-of-sample** performance of your models. Only the dataset* `brain_cancer_samples.csv` *should be used for the cross-validation tasks, where you will be in charge of choosing an appropriate set of hyperparameter values (at least 5) to scan. If you wish to standardise the dataset, please use the convention by which we use mean and standard deviation of the training set to standardise the test set, as discussed in the lecture.*

Questions:

2.1 k-Nearest Neighbours (25 marks)

2.1.1 (10 marks) - Train a *k*-Nearest Neighbour (*k*NN) classifier of the tumour type (with classes 0, 1, 2), using 5-fold cross-validation to find an optimal value of *k,* and assess its performance on the test set. Use the micro-averaged accuracy as the metric when evaluating the performance in cross-validation and on the test set.

The training set is an example of an imbalanced dataset, *i.e.*, one class is under-represented. Identify the minority class, next calculate the macro-average, micro-average and class-weighted average of accuracy and precision, and use these six metrics to assess whether the *k*NN classifier correctly predicts the minority class, justifying your answer. (In a class-weighted average, the weights are the class frequencies, which then multiply the class-wise metric).

2.1.2 (7 marks) - Design a weighted version of *k*NN to improve the prediction of the minority class. In this weighted *k*NN, each data point needs to be reweighted appropriately when computing the predicted class by majority vote. Explain your choice of the reweighting strategy. Use the same *k* as in Task **2.1.1.** Calculate the macro- and micro-average of accuracy and precision, and discuss your findings making a direct comparison with the results from task **2.1.1**.

2.1.3 (8 marks) - To investigate the model's ability to discriminate cancer diagnoses (classes 1 and 2), implement a 2-step *k*NN as follows. First, reformulate the previous classification task as a binary classification task, where the two classes are 'benign tumour diagnosis' (class 0) and 'malignant tumour diagnosis' (classes 1 and 2 combined). Train a *k*NN model for this binary classification task with the same *k* as in task **2.1.1**. Next, use class 1 and class 2 data to train another *k*NN model for the subsequent binary classification between class 1 and class 2, setting *k=1*. Then use these two *k*NN binary models to predict which of the test data points belong to class 0, class 1 and class 2, thus performing a 2-step binary classification for the original three-class classification problem. Measure the performance on the test set of this 2-step *k*NN by calculating again the macro- and micro-average of accuracy and precision. Explain your findings making a direct comparison with the results from tasks **2.1.1** and **2.1.2**.

2.2 Logistic regression *vs* kernel logistic regression (25 marks)

2.2.1 (10 marks) - Start from the formulation of the classification problem as a binary classification task ('benign tumour diagnosis' vs 'malignant tumour diagnosis'). For this binary classification task, train a penalised logistic regression model specified by the following loss function:

$$E(L) = -\frac{1}{N}\sum_{i=1}^{N} y^{(i)} \log h_{\boldsymbol{\beta},\beta_0}(\boldsymbol{x}^{(i)}) + (1-y^{(i)})\log(1-h_{\boldsymbol{\beta},\beta_0}(\boldsymbol{x}^{(i)})) + \frac{\lambda}{2}\|\boldsymbol{\beta}\|^2$$

$$h_{\boldsymbol{\beta},\beta_0}(\boldsymbol{x}) := \frac{1}{1+e^{-(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{\beta}+\beta_0)}}$$

where the term containing the hyperparameter λ is a Ridge-like penalty term on the magnitude of $\boldsymbol{\beta}$ (note that it does <u>not</u> include the intercept). Set λ = 0.0025 and initialise $\boldsymbol{\beta}$ and β_0 with zeros. Train the model using gradient descent with learning rate = 0.1 and evaluate its performance on the test set via the Precision-Recall curve and the area under this curve (AUC-PR).

2.2.2 (10 marks) - Formulate the kernelised version of the logistic regression model from task **2.2.1**, using the Laplacian kernel:

$$k(x,y) = \exp\left(-\alpha\|x-y\|_1\right), \quad \alpha > 0$$

Construct and write down explicitly the appropriate loss function. Comment on whether optimising this loss is a convex optimisation problem, justifying mathematically your answer.

2.2.3 (5 marks) - Train the kernel logistic regression model from task **2.2.2** via gradient descent, using the same values of λ and learning rate as in task **2.2.1**, and setting the kernel's parameter to: α = 100 and α = 0.3. Evaluate the model's performance for both values of α on the test set, using the Precision-Recall curve and AUC-PR, and compare the performance to the one of penalised logistic regression (task **2.2.1**).