Coursework for *Machine Learning* (COMS30083)

James Cussens, Xiyue Zhang, Wei-Hong Li, Xiang Li

1 Submission Instructions

Submission Deadline: 1pm Thursday 4 December 2025.

1. When you are ready to submit, put your files into a zip file called `cw_⟨userid⟩.zip` (Replace `⟨userid⟩` with your university username, note that is **not** your examination number or student number.)
2. You should submit your coursework via Blackboard. Go to the same 'Course' where you got this coursework PDF from, go to the 'Assessment, submission and feedback' section and then scroll to the bottom of that page to find the submission point.

2 General Instructions and Advice

1. You will see that some of the tasks you are asked to do are called **Code Tasks** and some are called **Report Tasks**. These labels are intended to clarify what you need to do. The marks for a **Code Task** are determined entirely by your submitted code scripts and the marks for a **Report Task** are determined entirely by your report.
2. Note that rather than submitting a single report for the entire coursework you are asked to submit 4 different reports: `report_regression.pdf`, `report_classification.pdf`, `report_clustering.pdf` and `report_seq.pdf`.
3. Your Python scripts should run without throwing errors.
4. Although in a real-word machine learning project producing quality code adhering to established software engineering practices is important, in this coursework assignment, quality of, or 'elegance' of, code does not contribute to your mark.
5. You can and should import appropriate libraries (e.g. scikit-learn, PyMC, PyTorch) to help you complete the tasks you are required to do.

6. Feel free to use any or all of the following as a starting point for your code: lab exercises, examples from the documentation for the libraries we used in the labs and also tutorial material from there. You do not need to acknowledge using code from such sources. Using code from elsewhere is allowed (as long as it does not come from another student on this unit), but please add an acknowledgement of its source as a comment in the code (not in your report).

3 Support available

1. You will see that the coursework is divided into 4 equally weighted parts and that each part is associated with one of the four lecturers on this unit. If you have questions on a task **please address your question only to the lecturer responsible for the task**.

2. There are *Coursework Support Sessions* which take place 1500-1700 in MVB 2.11PC on the following Tuesdays: 18 Nov, 25 Nov and 2 Dec. Attendance at these is, of course, optional.

3. The key principle behind providing support to students for this coursework is that all students receive identical information. To this end all queries about the coursework (other than those raised at Coursework Support Sessions) should be posted on Teams on the 'Open' channel of the COMS30083 group.

4 Regression (Xiyue Zhang) (25 marks)

Download (from Blackboard) the file `regression_insurance.csv`. This dataset contains demographic and lifestyle factors for 1338 individuals, together with their annual medical costs billed by health insurance. Your task is to construct regression models to predict `charges` from the other attributes using three approaches: (1) Linear Regression, (2) Neural Network, and (3) Bayesian Regression. You will compare the models' performance and briefly discuss your findings.

Submission instructions

You must submit two components:

1. **Code submission (all code tasks)**

   - Please submit a Python script (`.py`) file per Code Task (clearly named as `task1.py`, `task2.py`, `task3.py`).
   - Each Code Task must print or plot the requested outputs as stated in the task description (e.g. labelled coefficients/coefficient tables, RMSE values, and scatter plots).

2. **Concise report (all report tasks)**

   - Submit a PDF report.
   - Each report task should have a clear subheading (e.g., Report Task 1: Model summaries, Report Task 2: Model comparison).
   - Figures and Tables should be captioned and labelled clearly.
   - You do **not** need to include the code in the report.
   - Suggested length: (i) Report Task 1 (summary of results for all models): up to 300 words (excluding coefficient tables, figures). (ii) Report Task 2 (comparisons and discussion): up to 300 words.

3. **File naming and format**: Put your three Python script files (`task1.py`, `task2.py`, `task3.py`) and your PDF report (`report_regression.pdf`) in a folder named `regression_⟨userid⟩`. Then include this folder alongside your other coursework folders into your submission zip file `cw_⟨userid⟩.zip`.

Dataset description

The dataset contains the following variables:

`age`	Age of the primary beneficiary (numeric)
`sex`	Gender (`male`, `female`)
`bmi`	Body mass index (numeric)
`children`	Number of dependents (numeric)
`smoker`	Smoker status (`yes`, `no`)
`region`	Residential area (`northeast`, `northwest`, `southeast`, `southwest`)
`charges`	Individual medical costs billed by health insurance (numeric, **target variable**)

Preprocessing guidance Split the data into 80% training and 20% test sets using:

```
import pandas as pd
from sklearn.model_selection import train_test_split

# Load the data
file_name = "regression_insurance.csv"
data = pd.read_csv(file_name)

# Split into training and testing sets (80% / 20%)
X = data.drop(columns=['charges'])
y = data['charges']
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
```

You may preprocess the data in any reasonable way (e.g., one-hot encode categorical features, standardise numeric ones). An example preprocessing procedure is outlined below:

- Categorical features (`sex`, `smoker`, `region`) can be preprocessed using one-hot encoding (e.g., with `OneHotEncoder()` in `scikit-learn`).
- Numeric features (`age`, `bmi`, `children`) can be standardised (mean 0, variance 1) using `StandardScaler()`.
- Fit the preprocessing encoder/scaler **using the training data only**, then apply the same fitted encoder/scaler to process the test data. This prevents information leakage from the test set.
- Concatenate numeric and categorical features before training the model.

Code Task 1 Train a linear regression model to predict the insurance charges. Print the learned coefficients for each feature: round each coefficient to three decimal places for clarity and clearly label each coefficient with its corresponding feature name. Compute and print the Root Mean Squared Error (RMSE) on both training and test sets. Produce a scatter plot of predicted versus actual `charges` on the test set. (5 marks)

Code Task 2 Train a neural network model using `PyTorch`. The neural network architecture is up to you (a small network with 1–2 hidden layers is sufficient). Use mean squared error as the loss function and an optimiser such as Adam. Print training and test RMSE, and plot predicted versus actual `charges` on the test set. (8 marks)

Code Task 3 Build a Bayesian linear regression model for the insurance dataset using `PyMC`. Specify reasonable priors for regression coefficients and noise. Run MCMC sampling to obtain posterior draws, and print posterior means for model coefficients and noise. (6 marks)
Guidance: Please refer to the `PyMC` documentation and examples when completing this task. For sampling posterior predictions, you can also find guidance in the PyMC example Jupyter notebook *Counterfactual inference: calculating excess deaths due to COVID-19*, which is used in the Sequential Data section.

Report Task 1 Summarise the training and performance results for three models. (3 marks)

(a) For Code Task 1, include the learned coefficients for each feature: round each coefficient to three decimal places for clarity and clearly label each coefficient with its corresponding feature name in your report. Also include the RMSE on both training and test sets. In addition, include the plotted figure for the linear regression model.

(b) For Code Task 2, include the RMSE on both training and test sets computed in your report. In addition, include the plotted figure for the neural network.

(c) For Code Task 3, include the posterior means for each coefficient and noise. Round each coefficient to three decimal places for clarity and clearly label each coefficient with its corresponding feature name.

Report Task 2 Summarise and compare the performance of the linear regression model and neural network on the test data. (3 marks)

(a) Compare RMSE on the test set.

(b) Comment on model complexity and interpretability.

(c) State which model performs better and why.

5 Classification (Wei-Hong Li) (25 marks)

Download the *CIFAR–10* dataset from the website `https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz`. This dataset consists of $60,000$ 32×32 colour images in 10 classes, including airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck, with 6000 images per class. In this part, you will implement different classification algorithms and compare these classifiers using the typical approach of comparing performance (i.e., classification accuracy) on the testing set. The dataset has already been split into $50,000$ training images and $10,000$ test images. Using the official training and testing splits, apply (1) a single decision tree method, (2) some ensemble method and (3) SVM, to build a classifier from the training split. You should test performance on the testing set and explain your results. The goal when training the classifiers is to achieve good test set results, although, of course, you are not permitted to use the test set to achieve this goal.

Code Task 4 Using the provided code for preprocesssing data, where each sample is 3072 dimensional, you will need to reduce the dimension to 200. You can simply random sample 200 dimensions or use the dimensionality reduction technique, PCA `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`) to reduce the data dimension. Submit your work as a python scipt, i.e., `task4.py`. (3 marks)

Code Task 5 Using the data after dimension reduction, train a single decision tree classifier on the CIFAR–10 training set. Use cross-validation on the training split to select appropriate hyperparameters (e.g. depth, split criteria). Evaluate the final model on the official test set and report the test accuracy. Submit your work as a python scipt, i.e., `task5.py`. (4 marks)

Code Task 6 Train an ensemble classifier. Use cross-validation on the training set for hyperparameter tuning. Evaluate the tuned model on the test set and report the test accuracy. Submit your work as a python scipt, i.e., `task6.py`. (4 marks)

Code Task 7 Train a Support Vector Machine (SVM) classifier. Use cross-validation to choose hyperparameters such as kernel type, C, and γ. Evaluate the final SVM model on the test set and report the test accuracy. Submit your work as a python scipt, i.e., `task7.py`. (4 marks)

Report Task 3 For each of the three classifiers, explain the choices you made in a document called `report_classification.pdf`. This includes hyperparameters, and any experiments you performed to select these settings (excluding the test set). If you tried approaches and eventually abandoned, briefly describe them and explain why. In addition, for at least one of your classifiers, visualise 5 misclassified test images and comment on any patterns or common failure modes you observe. **Word limit: 350–450 words.** (5 marks)

Report Task 4 Discuss and explain the differences in performance between the three classifiers on the test set. Your explanation should relate differences in accuracy to model properties such as model complexity, bias–variance behaviour, suitability for high-dimensional image data, and computational constraints. You may refer to insights from your failure-case visualisation where relevant. Include all this in the document `report_classification.pdf`. **Word limit: 300–400 words.** (5 marks)

Guidance for preprocessing dataset. You can download the CIFAR–10 by wget:

```
wget https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
```

Then you will need to extract the dataset and you can run:

```
tar -xvzf cifar-10-python.tar.gz
```

After this, you extract the dataset under `cifar-10-batches-py`. And you can preprocess the dataset by:

```
import pickle
import numpy as np
import os

def load_batch(batch_path):
    with open(batch_path, 'rb') as f:
        batch = pickle.load(f, encoding='bytes')
        X = batch[b'data']
        y = np.array(batch[b'labels'])
        return X, y

def load_cifar10(data_dir):
    X_all = []
    y_all = []
```

```
    # 5 training batches
    for i in range(1, 6):
        batch_path = os.path.join(data_dir, f"data_batch_{i}")
        X, y = load_batch(batch_path)
        X_all.append(X)
        y_all.append(y)

    X_train = np.concatenate(X_all)
    y_train = np.concatenate(y_all)

    # test batch
    X_test, y_test = load_batch(os.path.join(data_dir, "test_batch"))

    return X_train/255, y_train, X_test/255, y_test

# Load the data
# You may need to specify 'data_dir' the directory of dataset folder
data_dir = "cifar-10-batches-py"
X_train, y_train, X_test, y_test = load_cifar10(data_dir)
```

This will give you the normalized training samples `X_train, y_train` and the testing set `X_test, y_test` and you should be able to build your classifier and complete this part.

6 Clustering (Xiang Li) (25 marks)

Objectives:

The MNIST dataset is a benchmark collection of 70,000 handwritten digit images (0–9), each sized 28×28 pixels. You will use the *scikit-learn* library and *MNIST* dataset to explore image clustering with k-means and GMM. By completing this coursework, you will:

- Understand and implement clustering workflows for image data.
- Perform clustering using k-means and Gaussian Mixture Models (GMM) and use cross-validation to select hyperparameters.
- Evaluate performance both quantitatively and qualitatively.

Task Specification:

- Load the MNIST dataset and visualize sampled images and labels. Show a grid plot with one row per class and five images per class.

- Building images clustering models. Using both K-Means and Gaussian Mixture Models (GMM) with different configurations. At least **two** hyperparameters should be explored for each method. You are suggested to use dimensionality reduction before modeling (e.g. PCA `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`) to make training computationally feasible.
- Evaluate clustering performance. Using at least **two** metrics (such as Rand Index, Mutual Information, Silhouette Score). More details about evaluation metrics can be found at: `https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation`.
- Visualize clustering results. Using the best-performing model and configuration. Show a grid plot with one row per cluster and five images per cluster.
- Results discussions. Which clustering method worked better for this dataset? Discussion on hyperparameter selection.

Deliverables:

Deliverable	**Description**
`clustering.ipynb`	Implement all required steps with **comments**. Max 50MB.
`report_clustering.pdf`	Describe tasks, methods, experiments, results, and analysis. Include tables and figures. Max 3 pages.

Marking Criteria:

Component	**Criteria**
Code Implementation	Load packages and data (10%), data visualization (5%), k-means and GMM models (15%), performance evaluation (10%), hyperparameter selection (10%), results visualization (5%), code correctness and readability (5%).
Report	Completeness and structure (5%), clarity (5%), figures and tables (10%), discussion on experiment results (15%), conclusion and self-reflection (5%).

Table 1: Marking criteria for the clustering task.

7 Sequential data (James Cussens) (25 marks)

Download the PyMC example Jupyter notebook *Counterfactual inference: calculating excess deaths due to COVID-19* from `https://github.com/pymc-devs/`

`pymc-examples/blob/main/examples/causal_inference/excess_deaths.ipynb`. This notebook learns a Bayesian model from data collected before COVID-19 broke out and then compares the number of deaths predicted by that model to the number of deaths that actually happened after COVID-19 arrived. I would encourage you to have a look at the entire notebook since I think it is an interesting application of the Bayesian approach, and there are a number of interesting plots which help us understand the data. However the tasks you need to do for this coursework do not require you to understand all the methods used in the notebook. For example, you do not need to understand the terms 'counterfactual inference' and 'causal reasoning' which are mentioned, and you can safely ignore, for example, the section **Prior predictive check**.

The notebook was written before some changes were made to PyMC so you need to edit it to run it using the current version of PyMC.

1. Change all occurrences of `pm.MutableData` to `pm.Data`.
2. Find the function `ZeroSumNormal`, find within that function a method call `model.add_coord` and remove the keyword argument `mutable`.

Now run the entire notebook and check that there are no errors.

Report Task 5. In the section with the title **Inference**, MCMC is used to (approximately) draw samples from a posterior distribution. You will see that there is a call there to `az.plot_trace` which generates plots of posterior distributions over the various parameters. Add to the notebook some code which will generate a tabular summary of these posterior distributions together with $\hat{R}$ values. You do not need to submit this code as a separate file or notebook. Instead, in your report: (1) include a copy of the code you added, (2) report on the $\hat{R}$ values and (3) explain why the $\hat{R}$ value is of interest. In the graphic with the title "Counterfactual: Posterior predictive forecast of deaths if COVID-19 had not appeared" we see a clear disparity between the number of deaths reported and the number of deaths predicted by the model. Using no more than 60 words, explain, what, if anything, is wrong with the model that it makes such poor predictions. This explanation and your answers to all other Report Tasks in this section should be included in a file called `report_seq.pdf`. (5 marks)

Report Task 6. To investigate the impact of COVID-19, the data used in the notebook is split into `pre` (data before 2020) and `post` (data after 2020). Make a change so that `pre` is now data before 2010 (and `post` is data after then). Use PyMC to produce (approximate) posterior distributions over the model parameters when trained on this new version of `pre`. In your report compare and explain any differences in the (approximate) posterior distributions between training on the original `pre` and the new one. Use at most 50 words. (5 marks)

Report Task 7. As a check, the notebook sees how well the trained model predicts the data from which it was trained—a model that cannot predict well its own training data is probably underfitting. Revert to `pre` being data before 2020 and `post` being data after 2020. In your report include:

- The original plot with the title *Posterior predictive distribution in the pre COVID-19 era*.

- The plot you get for *Posterior predictive distribution in the pre COVID-19 era* if the month indicator variables are excluded from the model, and
- The plot you get for *Posterior predictive distribution in the pre COVID-19 era* when both the month indicator variables and the temperature variable are excluded from the model

In fewer than 60 words explain the differences between the three plots. (5 marks)

Code Task 8. The data used for the notebook above can be obtained by:

```
pandas.read_csv(pymc.get_data("deaths_and_temps_england_wales.csv"))
```

Create an HMM for this data where the state of the HMM for each month is the temperature and the observation is the reported number of deaths. You will need discretise the temperature variable; how you do this is up to you. Also in order to make the task manageable you are to replace the original `deaths` variable with a discrete variable with only 3 values: `low`, `medium` and `high`, where it is up to you how to do this.

You should learn the parameters of the HMM in two ways:

1. Use both the sequence of (discretised) temperatures and the sequence of (low, medium or high) deaths as data. So this is supervised learning which is not the normal sort of learning for HMMs. `hmmlearn` does not appear to implement supervised learning so you will have to do it 'by hand'. Call the HMM with parameters learned in this way: HMM1.

2. Use just the sequence of (low, medium or high) deaths as data. This is normal HMM training and you can use `hmmlearn` to do it. Call the HMM with parameters learned in this way: HMM2.

Put the code you have written in a file called `hmms.py` and include it in your submission. (5 marks)

Report Task 8. Both HMM1 and HMM2 define a distribution over sequences of death values. Use `hmmlearn`'s `sample` method to sample sequences of death values from each HMM to get an approximate picture of this distribution. (This method will also sample values for the 'state', i.e. temperature but you can discard that.) Compare (in a manner of your choosing) this approximate distribution to the actual sequence of (low, medium or high) deaths and assess the quality of the two HMM models. Use at most 100 words to do this, but you can use as many plots as you like. (5 marks)