University of Lincoln Assessment Framework Assessment Briefing 2021-2022

Module Code & Title: CMP3751M-Machine Learning, Assignment
Contribution to Final Module Mark: 100%

Task 1 (30%). You are required to solve a polynomial regression example. You will need to download the Task1_data.csv file, which contains a simple 1-dimensional dataset with inputs x (1 input dimension) and output y (1 output dimension). The dataset has been generated from an unknown polynomial function plus a noise term. In this task, you are required to analyse the performance of a polynomial regression algorithm fitting the data and estimating the degree of the polynomial as well as its parameters. To achieve the task, you will need to implement the algorithm, evaluate its performance on the training as well on an independent test set. Then, you will need to discuss the relation of the degree of the polynomial to over- and underfitting.

Section 1.1: Implementation of Polynomial Regression (10%).

You are asked to implement the polynomial regression algorithm. To do so, you are required to use the following function:

```
def pol_regression(features_train,  y_train, degree):
    # code comes here
    parameters = ...
    return polynomial coefficients
```

The *pol_regression* function takes as arguments the training input and output values as input (both should be given as 1D numpy array with number of data-points elements). The last argument in the function defines the degree of the polynomial that we want to fit. It should return the parameters of the polynomial (note that for a polynomial of n-th degree, we have n + 1 parameters, i.e., polynomial_coefficients); parameters should be a 1-D numpy array. You will need to use the least squares method in order to calculate the polynomial parameters. You are not allowed to use functions from any pre-built libraries to calculate the least squares solution, but instead you are required to implement the least squares method through polynomial feature expansion and get the solution using only numpy matrix operations. **Do not use other function definitions different from the above one**; that means you should not change the definition of the main function "pol_regression". However, you can use helper functions outside this main function, should they help.

Section 1.2: Regress a polynomial of the following degrees: 0, 1, 2, 3, 6, 10 (10%).

After implementing the *pol_regression* function, use this function to regress the data set given in the .csv file of task 1 as part of this assignment. Regress a polynomial of the following degrees: 0, 1, 2, 3, 6, 10. Plot the resulting polynomial in the range of [-5, 5] for the inputs x. In addition, plot the training points. Interpret and elaborate on your results. Which polynomial function would you choose?

There is no need to split the data here. Just simply treat the whole data as training data. The plots for each degree can be either in the same graph (preferable) or in different ones.

Section 1.3: Evaluation (10%).

Using the returned results of the *pol_regression* function from the previous section, you need now to evaluate the performance of your algorithm. To do so, you first need to implement the following function:

```
def eval_pol_regression(parameters, x, y, degree):
        #your brilliant code comes here
        rmse = ...
        return rmse
```

The *eval_pol_regression* function takes as arguments the parameters computed from the *pol_regression* function and evaluates the algorithm's performance on the input x and output y data (again 1D numpy arrays). The last argument of the function again specifies the degree of the polynomial. In this function, you need to compute the root mean squared error (rmse) of the polynomial given by the parameters' vector. You should avoid using pre-built functions like numpy.polyval when evaluating a polynomial.

However, this time, split the same dataset provided into 70% train and 30% test set. You can use the "train_test_split" function in sklearn library since you are not asked here to implement polynomial regression functions.
Once again you need to train your polynomial functions that you developed in the previous section with degrees 0, 1, 2, 3, 6 and 10 on split training data.
Evaluate the training set rmse and the test set rmse of all the given degrees. Plot both rmse values using the degree of the polynomial as x-axis of the plot. Interpret your results. Which degree would you now choose? Are there any degrees of the polynomials where you can clearly identify over and underfitting? Explain your findings in great detail.

Task 2 (30%): In this task, several data for specific dog breeds have been collected. You will need to download the Task2 - dataset - dog_breeds.csv file. The data include four features, which are height, tail length, leg length, and nose circumference for each dog breed. The purpose of this experiment is to figure out whether these features could be used to cluster the dog breeds into three groups/clusters.

Section 2.1: Implementation of the K-Means Clustering (15%).

Implement the (simple) K-Means Clustering algorithm by creating the following functions:

```
def compute_euclidean_distance(vec_1, vec_2):
        # your code comes here
        return distance

def initialise_centroids(dataset, k):
        # your code comes here
        return centroids

def kmeans(dataset, k):
        # your code comes here
       return centroids, cluster_assigned
```

- The function compute_euclidean_distance() calculates the distance of two vectors, e.g., Euclidean distance
- The function initialise_centroids() randomly initializes the centroids
- The function kmeans() clusters the data into k groups

Section 2.2: clustering the dog breed data (15%).

After implementing the kmeans() function, use it to cluster the dog breed data. You need to run your k-means algorithm with all the four features as input. You should not slice the data to have two features for calculating Euclidean distance in k-means. Having said that, you need to cluster the data based on the four features instead of a portion of them. Use two different values for the parameter k: 2 and 3.

Draw the following three plots for the K-Means clustering results:

a) The first plot is a scatter plot, where x axis is the "height" feature and y axis is the "tail length" feature; Use different colours for the different cluster data points.
b) The second plot is also a scatter plot, where x axis is the "height" feature and y axis is the "leg length" feature; As before use different colours to depict the data points from different clusters.
c) The third plot is a line plot, where x axis is the 'iteration step' and y axis is the 'objective function value'. The plot should show a decreasing trend! You must plot the objective function (error function) at each iteration when you train the model.

Please generate the above three requested plots for k = 2 and do the same for k =3. The plotting tasks are just to display your clustering results in a 2-D space (height-length) for one k value. Since there are two k values, you should run your k-means ONLY TWICE, no matter how many plots you need to produce. Anyway, you will need to produce three plots for each k value, and they should not suggest you to re-run your algorithm for each plot.

Note: You need to implement the simple K-Means clustering following the above steps by yourself, group work is not allowed! You are allowed to use numpy, pandas and matplotlib libraries for processing and plotting data. However, other data science and machine learning libraries such as scikit-learn or other built-in libraries that have implemented the k-means algorithm are prohibited in this coursework.

Task 3 (40%)

HIV (human immunodeficiency virus) is a virus that attacks the body's immune system. If HIV is not treated, it can lead to AIDS (acquired immunodeficiency syndrome). In this task, you are required to develop machine learning models to predict HIV infection from measured features. The dataset "Task3 - dataset - HIV RVG.csv" provided consists of clinical records of both HIV patients and controls (healthy). The data has 5 retinal vascular geometry (RVG) features extracted from retinal images; the RVG features are described in [1].

Image number	Bifurcation number	Artery (1)/ Vein (2)	Alpha	Beta	Lambda	Lambda1	Lambda2	Participant Condition
1	1	1	0.60009863	2.14118476	0.77466034	1.15678779	0.89611761	Patient
1	2	1	0.82261224	1.83585755	0.90697974	1.00362703	0.91026938	Patient

Each image consists of a set of bifurcations. A bifurcation can be either artery or vein. The extracted features from each bifurcation are Alpha, Beta, Lambda, Lambda1 and Lmbda2. Participant condition is either patient or control.

[1] Al-Diri, B. and Hunter, A., 2009. Automated measurements of retinal bifurcations. In World Congress on Medical Physics and Biomedical Engineering, September 7-12, 2009, Munich, Germany (pp. 205-208). Springer, Berlin, Heidelberg.

The class membership of each row is stored in the field "Status". Status refers to the health condition of patients, or in other words, we consider this to be our label/annotation for the sake of all implementations. Unit of measurement or range of values of each feature are not relevant. However, features can be at different scales and/or measured in different units. Our task is to develop a set of classification models for automatically classifying patients as healthy or HIV infected, based on the RVG features. No prior knowledge of the domain problem is needed or assumed to fulfil the requirements of this assessment whatsoever.

Section 3.1: Data import, summary, pre-processing and visualisation (10%)

As a first step, you need to load the data set from the .xlsx file into a Python IDE. You should then provide a statistical summary of the dataset (i.e. mean values, standard deviations, min/max values for each feature). In data pre-processing, you need to think about whether you should normalise the data before starting training/testing any model and justify your decision.

To visualise the data, you need to generate two plots. The first one shall be a box plot, which will include the two classes ("Status"), i.e. control/patient, in the x-axis and the "Alpha" in the y-axis. The second one shall be a density plot for the feature "Beta", with the graphs of both classes appearing in the same plot. What information can be obtained from each of these two plots? Can one use any of these two plots to identify outliers? If yes, please elaborate.

Please include your explanation of implementation alongside the plots.

Section 3.2 Designing algorithms (15%)

You will now design an artificial neural network (ANN) classifier for classifying patients as patient or control, based on their RVG features. You will use the provided data set to train the model. To design an ANN, use a fully connected neural network architecture with two hidden layers; use the sigmoid function as the non-linear activation function for the hidden layers and logistic function for the output layer; set the number of neurons in each hidden layer to 500. Now randomly choose 90% of the data as training set, and the rest 10% as test set. Train the ANN using the training data, and calculate the accuracy, i.e. the fraction of properly classified cases, using the test set. Please report how you split the data into training and test sets. In addition, please report the steps undertaken to train the ANN in detail and present the accuracy results.

You will try different numbers of epochs to monitor how accuracy changes as the algorithm keeps learning, which you can plot using the number of epochs in the 'x' axis and the accuracy in 'y' axis [it should show the change of accuracy as the number of epochs increases].
Now use the same training and test data to train a random forests classifier with 1000 trees. The minimum number of samples required to be at a leaf node has two options, i.e. 5 and 10. Please report the steps for training random forests for both options and show their accuracy results on the test set.

Section 3.3: Model selection (15%)

You have designed ANN and random forests classifiers with almost fixed model parameters (e.g. number of neuros for ANN, number of trees for random forests, etc). The performance of the model could vary when those model parameters are changed. You would like to understand, which set of parameters are preferable, and also to select the best set of parameters given a range of options for ANN and random forests. To do so, one method is to employ a cross-validation (CV) process. In this task, you are asked to use a 10-fold CV. As a first step, randomly split the data into 10 folds of nearly equal size, and report the steps undertaken to do this.
For ANN, create three ANN classifiers with 50, 500, and 1000 neuros in each hidden layer (remember there are two hidden layers), respectively. Similarly, create three random forest classifiers with 50, 500, and 10000 trees, respectively, and set the "minimum number of samples required to be at a leaf node" =10 for all of them.

Please do the following tasks for both methods:

1) Use the 10-fold CV method to choose the best number of neurons or number of trees for ANN and random forests respectively.
   a) Report the processes involved when applying CV to each model.
   b) Report the mean accuracy results for each set of parameters, i.e. for different number of neurons and different number of trees accordingly.
   c) Which parameters should we use for each of the two methods, i.e. specifically for ANN and random forests?
2) Until now, you have selected the best parameters for each method, but we have not decided the best model yet. With the results you have had so far,
   a) which method is best, ANN or random forest?
   b) Please discuss and justify your choice, reflecting upon your knowledge thus far.

Report Guidance

Your report should be self-explanatory without referring to the source code. You describe and justify each step that is needed to reproduce your results by using code snippets, screenshots and plots. Code snippets are especially important for the functions that you are asked to implement.

Task 1: Your report must conform to the structure from below. Information on specific marking criteria for each section is available in the accompanying CRG document. You must submit a written report containing three distinct sections that provide a full and reflective account of the processes undertaken. You will be asked to provide several figures that demonstrate the performance of the algorithm. For all figures, please provide all necessary code snippets (including basic documentation) such that your results are reproducible. Use Python as the programming language to complete this assessment. You are allowed to use the pandas library for data management, however, scikit learn or other pre-built libraries, such as sklearn.preprocessing. PolynomialFeatures, numpy.polyfit, and scipy.polyfit, are prohibited and usage of them will lead to significant reduction of your overall mark. The regression algorithm needs to be implemented by yourself using numpy matrix operations only. Do not use pre-built functions to perform the regression, but you are allowed to use the numpy.linalg.solve or numpy.linalg.inv function to calculate the inverse of a matrix. You can reuse material from the workshops as long as the solutions do not include unauthorised pre-built libraries.

Task 2: Your report must conform to the below structure and include the required content as described. Information on specific marking criteria for each section is available in the accompanying CRG document. You must submit a written report containing two distinct sections that provide a full and reflective account of the processes undertaken.
You need to write a short report to discuss how you complete the task (see Report Guidance) and go into enough depth to demonstrate knowledge and critical understanding of the relevant processes involved.

Task 3: You are allowed to use any pre-built in python libraries for this task, but make sure you explain them when use them. Your report must conform to the below structure and include the required content as outlined in each section. Information on specific marking criteria for each section is available in the accompanying CRG document. You must supply a written report containing three distinct sections that provide a full and reflective account of the processes undertaken.
You need to write a report that discusses how you completed the tasks and go into enough depth to demonstrate knowledge and critical understanding of the relevant processes involved. 100% of available marks are through the completion of the written report, with clear and separate marking criteria for each required report section.

Learning Outcomes Assessed:

On successful completion of this component a student will have demonstrated competence in the following areas:

- [LO1] Critique and appraise the scope and limits of machine learning methods by identifying their strengths and weaknesses
- [LO2] Using a non-trivial dataset, plan, execute and evaluate significant experimental investigations using multiple machine learning strategies

Knowledge & Skills Assessed:

Subject Specific Knowledge, Skills and Understanding: You are expected to demonstrate a good knowledge of machine learning algorithms, by exploring the machine learning literature. You will show your ability to use these algorithms to complete tasks set in this assignment. You will also demonstrate your academic writing skills.

Personal Skills: You will work independently to complete tasks. The assessment will also assess your responsibility to manage your coding and data. It will explore your creativity and problem-solving skills.

Assessment Submission Instructions:

The report should be a **maximum of 15 pages (including everything!).** Keep in mind that:

- The report must contain your name, student number, module name
- The report must be a single PDF file
- The report must be formatted in **single line spacing** and use an **11pt font**
- The report does not include this briefing document, no cover page and table of content
- You describe and justify each step that is needed to reproduce your results by using code-snippets, screenshots and plots

The deadline for submission of this work is included in the School Submission dates on Blackboard. You must make an electronic submission of your work to the **Blackboard Turnitin upload area** for assessment item.

This assessment is an individually assessed component. Your work must be presented according to the School of Computer Science guidelines for the presentation of assessed written work. Please make sure you have a clear understanding of the grading principles for this component as detailed in the accompanying Criterion Reference Grid. If you are unsure about any aspect of this assessment item, please seek the advice of the delivery team.

Date for Return of Feedback:

You can find the date for the return of feedback in the School Hand-in Schedule which is available on Blackboard under the Assessment Page of this module

Format for Assessment:

This is an individual assignment. Your work must be presented according to the Lincoln School of Computer Science formatting guidelines for the presentation of assessed written work. The final submission must have a report in PDF format.

The source codes created for this assignment can be submitted in a zipped file if you wish to do so. The submission should be through the Blackboard upload area for this assessment item.

Feedback Format:

Feedback will be given to you through Blackboard. Additionally, face-to-face feedback can be obtained on request with the module staff.

Additional Information for Completion of Assessment:

The delivery team strongly recommends for you to strictly follow the following points:

- Please add sensible labels and captions to plots you will provide.
- Please describe and justify each step that is needed to reproduce your results by using code-snippets, screenshots and plots. When using screenshots or plots generated from Python, please make sure they are clearly readable with sensible axis labels
- Please use references effectively to support your arguments in the report.
- Please interpret the results of your data analysis and model developments
- Explain trends, characteristics or even outliers when you summarise and describe data
- Always refer to accuracy as performance metric when reporting the "performance" of the algorithm.
- Should you decide to use prebuilt python libraries, such as scikit-learn, rather than implementing them yourself, you will need to provide extra justification for the steps undertaken to arrive to the conclusions, and also demonstrate adequate understanding of the algorithms. Analytical approach is required to arrive to credible and justifiable solutions.

The delivery team strongly recommend for you to strictly follow the following points:

- Please add sensible labels and captions to plots you will provide.
- Please explain clearly the plots, code-snippets or screenshots that you will provide in the report.
- Please use references effectively to support your arguments in the report.

Assessment Support Information:

There will be an assignment discussion during the lectures/workshops which will help you to better understand the numerous requirements of this assignment. If you are unsure about any aspect of this assessment document, please seek advice from a member of the delivery team.

There will be an assignment discussion during the lectures/workshops which will help you to better understand the numerous requirements of this assignment. If you are

unsure about any aspect of this assessment document, please seek advice from a member of the delivery team.

Important Information on Dishonesty & Plagiarism:

University of Lincoln Regulations define plagiarism as 'the passing off of another person's thoughts, ideas, writings or images as one's own...Examples of plagiarism include the unacknowledged use of another person's material whether in original or summary form. Plagiarism also includes the copying of another student's work'.

Collusion is defined as when a student submits work for assessment done in collaboration with another person as entirely their own work or collaborates with another student to complete work which is submitted as that other student's work. Collusion does not apply in the case of the submission of group projects, or assessments that are intended to be produced collaboratively.

Plagiarism and collusion is a serious offence and is treated by the University as a form of academic dishonesty. Students are directed to the University Regulations for details of the procedures and penalties involved.

For further information, see www.plagiarism.org

Questions and answers:

Q1: What do I need to do when I have the assessment?

A1: Please read the brief very carefully as it clearly outlines what you are and are not allowed to use in terms of python libraries.
The best way to go about the assignment is to read it carefully a couple of times and allow yourself some time to understand what you are actually required to do. Algorithms need to be developed without the use of pre-built libraries (e.g. Scikit-Learn) as that would be two lines of code, but you are allowed to use numpy matrix operations. Again, the brief explicitly mentions this, so please read it carefully.

Q2: Are there any additional materials to support my assessment?

A2: Lecture and workshop material should be enough to give you guidance on how to go about implementing the algorithms but for some extra help, you might want to have a look at the links below. They provide you with step-by-step guidance not only for the requirements of the assignment but also for your own independent learning process.
https://mubaris.com/posts/kmeans-clustering/
https://pythonprogramming.net/k-means-from-scratch-machine-learning-tutorial/
https://en.wikipedia.org/wiki/Polynomial_regression
https://en.wikipedia.org/wiki/Ordinary_least_squares
https://www.mathsisfun.com/algebra/systems-linear-equations-matrices.html
https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.solve.htm

Q3: What python libraries am I allowed to use?

A3: Unless otherwise stated, python libraries that have machine learning algorithms are not allowed to use. Non-ML specific libraries, such as matlibplot for plotting and numpy for numerical operation, are allowed.

Q4: What kind of support for assessment can I ask from the delivery team?
A4: Although we can certainly clarify things that might be confusing or unclear, we cannot help you with implementing the tasks and answering questions that are part of the assignment implementations, which may constitute an answer to tasks in your assessment.
For example, you are asked to elaborate on the objective functions in the assignment for polynomial regression and k-means, i.e. what it is, why is needed etc. These are the questions designed for you to answer, and it would be strange if students ask the delivery team about "what is an objective function?" after corresponding lectures and workshops.

Q5: Can I send you my report /my code to give me feedback?
A5: For the sake of fairness and equal treatment of all students, we cannot respond to questions such as "Can you please have a look at my code?", "Is my report ok?", "I am not sure I understand polynomial and/or k-means", and other similar ones. Please DO NOT send us your report for review when you seek support from us.

Q6: Can I use methods different from the lectures and lectures in my assessment?
Yes. You can use any method you like as long as you can complete the tasks set for you, with a clear explanation of the method you use and do not use built-in python libraries.

Q7: Do I need to submit my source code as supplementary materials?
Not necessary. It is up to you whether or not you want to upload source codes as additional supporting material for your report. However, your report should be self explanatory without referring to the source code. In other words, as mentioned in the brief, you describe and justify each step that is needed to reproduce your results by using code snippets, screenshots and plots. Code snippets are especially important for the functions that you are asked to implement.

Q8: Are we allowed to use python libraries such as Keras, TensorFlow , Statistics and SK. Learn Preprocessing?
Yes, you are allowed to use these libraries, but we recommend using sklearn.

Q9: Can we use sklearn for our ANN and random forests and achieve a 1st grade (providing we give extra justification for the steps and explain our algorithms)?

Yes.

Q10: Is it okay to use quasi-newton methods as the optimizer instead of stochastic gradient descent? I yield far better results using quasi-newton methods.

Yes.

Task 1 Clarification

section 1.1, we expect a generic least squares solution for any order/degree of polynomial regression, not just one specific for degree =2.
section 1.1, the statement "Do not use other function definitions different from the above one" means you should not change the definition of the main function "pol_regression". However, you can use helper functions outside this main function, should they help.

section 1.2, there is no need to split the data here. Just simply treat the whole data as training data. The plots for each degree can be either in the same graph or in different ones.

section 1.3, you should split the data as required, as re-run the polynomial regression that you have developed in section 1.2, with the split training data.

Task 2 Clarification

section 2.1, "Randomly initialise the centroids" - This means you will use the simple k-means which initialises the centroids with randomly sampled data points. However, you may interpret it as drawing random numbers from the data range. It is fine if you initialise centroids this way. However, either way, you will need to justify the approach you choose.

section 2.2, "The four features of the data are used as the input to your algorithm" means you need to run your k-means algorithm with all the four features as input. You should not slice the data to have two features for calculating Euclidean distance in k-means. Having said that, you need to cluster the data based on the four features instead of a portion of them. The plotting tasks are just to display your clustering results in a 2-D space (height-length) for one k value. Since there are two k values, you should run your k-means ONLY TWICE, no matter how many plots you need to produce. Anyway, you will need to produce three plots for each k value, and they should not suggest you re-run your algorithm for each plot.

Task 3 Clarification

Section 3.2: “The minimum number of samples required to be at a leaf node has two options, i.e. 5 and 10” means you need to create two models, one with 5 and the other with 10.

Section 3.3. Until now, you have selected the best parameters for each method, but we have not decided yet, which the best model is. With the results you have had so far, which one is the best model across all combinations of ANNs and random forests for this data set? Please discuss and justify your choice, reflecting upon your knowledge thus far.

You should identify the best parameters for ANN and forest (i.e. best ANN, best forest) and then choose one from these two best models.