QBUS6850 Assignment 1

Due date/time: 2:00pm, Monday, 09-Sep-2019

Value: 10% of the final mark

Notes to Students

1. **Required submissions: ONE** written report (word or pdf format, through Canvas- Assignments - Report Submission (Assignment 1)) and **ONE** Jupyter Notebook .ipynb or .py code file (through Canvas- Assignments- Upload Your Program Code Files (Assignment 1)).
2. The assignment is due at **2:00pm on Monday, 09-Sep-2019**. The late penalty for the assignment is 5% of the assigned mark per day, starting after 2:00pm on the due date. The closing date **2:00pm on Monday, 16-Sep-2019** is the last date on which an assessment will be accepted for marking.
3. As anonymous marking policy, only include your Students ID in the report and do **NOT include your name**.
4. The name of the report and code file must follow: **SID_QBUS6850_Assignment1_S2_2019.**
5. Your answers shall be provided as a report giving full explanation and interpretation of any results you obtain. Output without explanation will receive **zero** marks.
6. Be warned that plagiarism between individuals is always obvious to the markers of the assignment and can be easily detected by Turnitin.
7. The data sets for this assignment can be downloaded from Canvas.
8. Presentation of the assignment is part of the assignment. **Markers will allocate up to 10% of the mark for writing in clarity and presentation**. You may insert small section of your code into the report for better interpretation when necessary. Think about the best and most structured way to present your work, summarise the procedures implemented, support your results/findings and prove the originality of your work.
9. Numbers with decimals should be reported to the **four-decimal point**.
10. The report should be **NOT more than 10 pages** including everything like text, figure, tables and small sections of inserted codes, etc, but excluding the appendix.

Key rules:

- Carefully read the requirements for each part of the assignment.
- Please follow any further instructions announced on Canvas, particularly for submissions.
- You **must use Python** for the assignment.
- Use "***random_state= 0***" when needed, e.g. when using "*train_test_split*" function of Python. For all other parameters that are not specified in the questions, use the default values of corresponding Python functions.
- Reproducibility is fundamental in data analysis, so that you will be required to submit a code file that generates your results. Not submitting your code will lead to **a loss of 50%** of the assignment marks.

- Failure to read information and follow instructions may lead to a loss of marks. Furthermore, note that it is your responsibility to be informed of the University of Sydney and Business School rules and guidelines, and follow them.
- Referencing: Harvard Referencing System. (You may find the details at: http://libguides.library.usyd.edu.au/c.php?g=508212&p=3476130)

Task A (30 Marks)

You will work on the White Wine Quality dataset "**Wine_2019.csv**" for Task A.

This dataset is related to white vinho verde wine samples, from the north of Portugal. The goal is to set up a **linear regression task** to model wine quality based on physicochemical tests.

Due to privacy and logistic issues, only physicochemical (the input/feature) and "*Quality*" (the target) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).

You can download the dataset from Canvas, and find the detailed information of this dataset as here: https://archive.ics.uci.edu/ml/datasets/Wine+Quality

1. You decide to use this dataset for a regression task, that is to use the existing information to predict the "*Quality*". But before that, you want to exam the data and conduct exploratory data analysis.

   Please choose **two most relevant features** and present your results. Carefully explain your feature selection rules and criteria (No need to train a regression model here).

2. Write your own Gradient Descent algorithm to estimate the parameters of the given linear regression problem **without the intercept**. You may write a python function named such as `Gradient_Descent_Algo`, with various inputs, e.g. data matrix $\boldsymbol{X}$, target $\boldsymbol{t}$, an initial parameter $\boldsymbol{\beta}^{(0)} = [0, \dots, 0]^T$, learning rate, the number of GD iterations T, stopping criteria and other arguments you see appropriate.

   Use 3 features: "*density*", "*residual sugar*" and "*volatile acidity*".

   - Find the approximate range of acceptable learning rates and explain why some learning rates are defective. You do NOT need to do train test split and cross validation for this question, and only need to grid search a range of learning rates, based on your justified choice.
   - Find the optimal learning rate in this range and explain why this is the optimal value. You should decide your criteria of judging the optimal learning rate.

3. After your first trail, you think the experiment is feasible. You now decide to build a linear regression model and **use all the features** (all the columns except the "*Quality*" column) to predict the "*Quality*" value.

(1) Use "*train_test_split*" function to split 80% of the data as your training data, and the remaining 20% as your testing data.
(2) With the train set, use the linear regression model *LinearRegression* in the *scikit-learn* package to build two linear regression models to predict the "quality" value, with and without the intercept term.
(3) Compare the test performance (using the below loss function (1)) of two models and explain the importance of the intercept term.

$$L(\boldsymbol{\beta}) = \frac{1}{2N}\sum_{n=1}^{N}(f(\mathbf{x}_n, \boldsymbol{\beta}) - t_n)^2 \quad (1)$$

4. As feature engineering is important for regression, suppose for this time you want to centralize your data first and then build a linear regression model again.
   (1) Take the same set of the training as in the previous question (Question 3). Construct the **centred** training dataset by conducting the following steps in your Python code:
      (i) Take the mean of all the training target values ("*Quality*" value), then deduct this mean from each training target value. Take the resulting target values as the new training target values $\mathbf{t}_{new}$;
      (ii) In the training data, take the mean of the first feature column "fixed acidity", then deduct the mean value from the original feature column. Take the result as the new feature $\mathbf{x}_{new}^{\mathbf{1}}$;
      (iii) Repeat the step (ii) for the rest feature columns and take the results as new features $\mathbf{x}_{new}^{\mathbf{2}}, \mathbf{x}_{new}^{\mathbf{3}}, \dots, \mathbf{x}_{new}^{\mathbf{11}}$;
   (2) Build a linear regression model to fit to the new data. Then test the model performance over the test data (the same set of the test data as in the Question 3), using *LinearRegression* in the *scikit-learn* package. Note that, when you take your test data into the model to calculate predictive performance scores, you should transform the testing data and you need to decide how to do this.

      Report your new results with loss function (1) and report the values of coefficients $\boldsymbol{\beta}$ and intercept $\boldsymbol{\beta}_{\mathbf{0}}$.

      Compare and report your results/findings from Question 3 and Question 4.

5. For some reasons, you decide to (1) randomly select (with your own selection of the random sampling methodology) only 400 data points from good quality wine (*quality >6*) and 400 data points from poor quality wine (*quality <=6*), so 800 data points in total. Although this strategy could significantly reduce the training time, it could also make your trained linear regression not "general" enough (due to the limited number of data). Therefore, you decide to (2) add LASSO regularization into your loss function to try to improve the model performance. In here you need to select a suitable value of regularization term λ to control the regularization factor, using cross validation. Use 80% and 20% "*train_test_split*", and you may consider feature engineering for LASSO.

   Write python scripts to finish sub-task (1) and (2).

Then (3) re-train the linear regression model with all training examples & the selected LASSO regularization term λ, and produce the test performance.

Report your new test performance with loss function (1) and compare them with the ones from the previous question (Question 4) and discuss why this happens. You must provide solid evidence and detailed explanations to prove your conclusion.

Task B (30 marks)

You will work on the credit risk defaulting classification task, using dataset "**Loans_Data_2019.xlsx**". Variable descriptions are in "**LCDataDictionary.xlsx**". "*debt_settlement_flag*" is the target variable.

1. Load the data and conduct exploratory data analysis.

   Your analysis should present the distributions of features, discuss the properties, e.g. skewness and kurtosis, of features' distributions, the relationship between features and the target, etc.

   Please choose **three most relevant features** and present your results. Carefully explain your feature selection rules and criteria (No need to train a logistic regression model here).

2. The loss function of logistic regression and gradient descent algorithm are defined as in slide 24 of Lecture 3.

   Now your task is to implement the Gradient Descent algorithm for the logistic regression **with intercept**. You also might consider **feature engineering** to make your algorithm more efficient and decide whether engineering the target variable.

   - Use 4 features: "*loan_amnt*", "*annual_inc*", "*int_rate*" and "*installment*"
   - Use $\boldsymbol{\beta}^{(\mathbf{0})} = [0, \dots, 0]^T$ as your initialization point.
   - Find the approximate range of acceptable learning rates and explain why some learning rates are defective. You do NOT need to do train test split and cross validation for this question, and only need to grid search a range of learning rates, based on your justified choice.
   - Find the optimal learning rate in this range and explain why this is the optimal learning rate and report your parameter estimates. You should decide your criteria of judging the optimal learning rate.

3. For this question, you can use *LogisticRegression* from *scikit-learn.*

   In Question 1, based on your exploratory analysis and domain knowledge in credit risk defaulting, you have selected the **three most relevant features** that best fit the logistic regression model for predicting the "*debt_settlement_flag*".

   Use "*train_test_split*" function to split 80% of the data as your training data, and the remaining 20% as your testing data. Use your selected three most relevant features, fit the logistic regression model **with intercept** with the training data, and generate the test set prediction performance (loss function value of logistic regression defined as in slide 24 of Lecture 3).