Assessment – Data Classification (40 marks)

This assessment counts for 20% of the overall assessment for this module. You must submit your Python code in a Jupyter Notebook by 12 noon on the 3rd of April 2020 via Canvas. A demo will be taking place during the practical sessions in the week starting on the 20th of April 2020. The final mark of this assessment will be given based on your performance during the demo. Therefore, you **must** attend the demo.

The programming language you should use to finish this assessment is Python (in version 3 and above). You can use functions from the following packages: Numpy, Pandas, Matplotlib, Seaborn and Sklearn. All Python skills needed to do this assessment have been covered in the practical sessions –practical notes are available on Canvas.

Information on the Data

Fozziwig's Software Developers have contracted you to explore the possibility of an automated software defect prediction system. They want to know if developing such a system would be cost-effective, based on the predictive *accuracy* that you can achieve with a sample of their data.

Static code metrics are measurements of software features. They can be used to quantify various software properties which may potentially relate to defect-proneness, and thus to code quality. Examples of such properties and how they are often measured include: size, via lines of code (LOC) counts; readability, via operand and operator counts (as proposed by [1]); and complexity, via linearly independent path counts (this relates to the control flow graph of a program, and was proposed by [2]).

The data that you have been given contains the static code metrics for each of the *functions* which comprise a software system. This system was developed by *Fozziwig's Software Developers* several years ago. As well as the metrics for each function, it has also been recorded whether or not a fault was experienced in each function. This data came from the software testers who examined the system before it was publicly released.

You have been given two labelled data files, a training data set (*trainingSet.csv*) and a testing data set (*testingSet.csv*). Each data set contains 13 features (each one a software metric). Class labels are shown in the last column of each file: a value of `+1' means `defective' (the software module contained a defect (fault)) while a value of `-1' means `non-defective'. Note that this is clearly a simplification of the real world, as both fault quantity and severity have not been taken into account.

Task 1: Data pre-processing and data exploration (**6 marks**)

a) Use Pandas to load both trainingSet.csv and testingSet.csv (1 mark).
b) Find the number of patterns in each class for both loaded data sets using Python (1 mark).
c) Choose an attribute and generate a boxplot for the two classes in the training set (1 mark).
d) Show one scatter plot, that is, one feature against another feature. It is your choice to show which two features you want to use. You need to use the training set (2 marks).
e) Divide the original training set into a smaller training set (II) and a validation set. In this task, you need to use 55% of total training data points as the validation set (1 mark).

Task 2: Do a principal component analysis (10 marks)

a) Perform a PCA analysis on the original training data set (3 marks).
b) Plot a scree plot to report variances captured by each principal component (2 marks).
c) Project the test set on the same PCA space produced by the original training dataset (2 marks).
d) Plot two subplots in one figure: one for the training data in the PC1 and PC2 projection space and label the data in the picture according to its class; the other one for the test data in the same PCA space and label the data in the picture according to its class (3 marks)

Task 3: Do a classification using the Naïve Bayes Classification model (4 marks)

Train the model using the original training set and report the performance on the test set including accuracy rate.

Task 4: Investigate how the number of features in the training dataset affects the model performance on the validation set (16 marks)

a) Use the training set (II) to train 13 Naïve Bayes Classification models, with 13 different feature sets. That is: the first one is to use the 1st feature only; the second one is to use the 1st and the 2nd features; the third one is to use the 1st, 2nd, and 3rd features, the fourth one is to use the first 4 features, and so on.
   Measure the accuracy rate on both the training set and the validation set. Report the results by plotting them in a figure: that is, a plot of the accuracy rate against the number of features used in each model. There should be two curves in this figure: one for the training set (II); the other one for the validation set (10 marks).
b) Report what is the best number of features you would like to use in this work and explain why you choose it. Write it down in your Jupyter notebook (3 marks).
c) Use the selected number of features to train the model and report the performance on the test set (3 marks).

Task 5: Summarize your findings, write your conclusions using critical thinking (no more than 100 words) and write it down in your Jupyter notebook (**4 marks**).

Hand in date: by 12 noon on 03/04/2020 via Canvas.

What to submit:

Submit a .ipynb file to show your completed Python code. The file should be identified by your student ID number.

Demo:
A demo will be taking place during the practical sessions in the week starting on the 20th of April 2020, depending on which group you are timetabled for your practical session.

During the demo (10 minutes - 15 minutes per student), you will be asked to run the code you have submitted in your .ipynb file and you also need to answer some questions to show that you understand the work you have done.

The final mark of this coursework will be given based on your performance during the demo.

Demonstration Mark (out of 5)	Maximum Coursework Mark (out of 20)
0-1: (almost) no understanding of the work	0~9
2-3: some understanding of the work	10~15
4-5: fully understands the work, maybe with a minor error	16 ~20

Serious adverse circumstances

Serious adverse circumstances are significant circumstances beyond a student's control that would have affected the student's ability to perform to their full potential if they were to submit or attend assessments at the appointed time. If, despite such circumstances, you decide to sit/submit an assessment, the University will not normally accept a claim of serious adverse circumstances in respect of that assessment.

If there are Serious Adverse Circumstances that have affected your assessment(s), you must communicate details to the University together with appropriate evidences, using the form provided by the School. You should read the University's guidance on Serious Adverse Circumstances before you submit an assessment (including sit for the demo). Full guidance can be found in your Programme Handbook**.**

References

[1] Maurice H. Halstead: *Elements of Software Science (Operating and programming systems series)*. Elsevier Science Inc. 1977

[2] Thomas J. McCabe: A complexity measure, in *ICSE '76: Proceedings of the 2nd international conference on Software engineering*. IEEE Computer Society Press. 1976