Graded Assignment 2 – Modelling and Prediction June 2022

For this assignment you are expected to apply the skills and knowledge you have developed in this unit so far. You are also expected to independently engage in further investigation and research of the topics that are covered in this assignment in order to advance and develop your knowledge.

This assignment will be marked out of 100, and is split into two tasks, each accounting for 50% of the total mark respectively. Note that the second part of the assignment is designed to be more difficult, and will likely take more work to achieve high marks.

Data

You are provided with a dataset of reviews for cars in car-reviews.csv linked on the Engage page of this Graded Assignment. Each review is labelled with either 'Pos' or 'Neg' to indicate whether the review has been assessed as positive or negative in the sentiment it expresses. You should treat these labels as a reliable indicator of sentiment, and you can assume that there are no neutral reviews.

There are 1,382 reviews in the CSV file in total, with an equal representation of both positive and negative classes (i.e. 691 reviews each).

Task 1 (50%): Baseline solution - Naïve Bayes classifier

As a preliminary task, you need to parse the CSV data file provided into an appropriate Python data structure with suitable (and convenient) data type handling. For this you are free to use either the CSV-parser that you implemented for Graded Assignment 1, or any other CSV reading library that you wish.

Then, in the same Jupyter notebook, implement a Naïve Bayes binary sentiment classifier using 80% (1106) of the reviews as training data. The training data should be selected from the full dataset using a sensible random sampling approach. Test your classifier using the remaining 20% (276) of the reviews and report the classifier's performance using a confusion matrix.

Data Leakage. It is important that you avoid issues of data leakage, meaning that your classifier should only be trained using data that it has access to from within the training data set. In this light, consider the mitigations that are necessary in two cases:

- **case 1:** there are words that only appear in the test data, but not at all in the training data;
- **case 2:** there are words that only appear in one class of the training data reviews, but not the other.

It is up to you to decide how you handle these cases. However, regardless of your choices, words that appear *exclusively* in the test data (i.e. do not appear in either class of the training data) must not be part of the classifier's vocabulary. Further, as a minimum, your code should not crash when encountering unseen words in the test dataset.

Preprocessing. You will need to perform some basic data preprocessing before using it in your classifier. This should include:

- **Redundant token removal.** Identifying and excluding all punctuation and common words that are not likely to affect sentiment:
  e.g. stopwords – see https://en.wikipedia.org/wiki/Stop_word).
  As an example, *Natural Language Toolkit (NLTK)* in Python has lists of common stopwords that you may wish to use, but you are also free to find and use other libraries or tools for this.

- **Redundant formatting removal.** Ensuring that remaining words are **not** case-sensitive i.e. the classifier should not distinguish upper/lower case characters.

Your sentiment classifier should use a bag-of-words technique for text data vectorisation, in which you build a vocabulary of individual words that appear in the dataset after preprocessing.

You should consider treating variations of a word (e.g. 'fault', 'faults' and 'faulty') as instances of the same root token (e.g. 'fault') when you are using them in your classifier. Hence you need to investigate and implement *stemming* as part of data preprocessing.

For each review you should create a vector as input for your classifier, containing EITHER binary values indicating whether a word/stem occurs in the review OR a numerical count of the number of times each word/stem appears. As described above, vectors that are used to train the classifier should only include words that appear in the training data (and not words that only exist within the test data).

Note on the use of libraries: You do not need to code everything required from scratch. For this graded assignment, you are encouraged to make use of existing libraries for all tasks. For example, for this task, you may find the *MultinomialNB classifier* in *scikit-learn* and natural language processing tools such as *NLTK* and *spaCy* useful.

It is also important to note that there is no single correct answer in terms of the output and performance of your classifier. This will depend on the choices you make about how you deal with the data at each stage of the process – your markers will not be looking for a specific level of performance, rather that you have taken appropriate steps and implemented them correctly.

Your solution to Task 1 will be assessed according to the following criteria:

Assessment Criteria for Task 1	**Marks Awarded**
Does the submission demonstrate that words and punctuation, which are unlikely to affect sentiment, have been excluded from the sentiment classifier? *Note: passing this section is required for subsequent marks*	5
Are words from the reviews being handled in a case-insensitive way?	5
Does the submission demonstrate that words with the same stem have been appropriately recognised and treated as variations of the stem? This should be demonstrated for at least 3 different stems.	5
Does the code produce some output to demonstrate that a vector has been created for each review, where each element in the vector represents EITHER a binary variable indicating the presence of a word/stem in a review OR the number of times that a word (or word stem) appears? Note that the output does not need to show the vector for all reviews, this only needs to contain a small sample of reviews.	5
Does the code clearly show correct implementation the Naïve Bayes model for the given classification task, either through the use of an existing library or coded from scratch?	10
Does the code clearly show that 80% of the data has been used to train the classification model, and that the remaining 20% of the data set has been used as test data? Does the code show that only the training data (i.e. no test data) has been used for the classifier training?	5
Is the code showing any handling of the presence in the test set of any words unseen in the training set (as a whole or a specific class thereof)? The reasoning behind the implemented handling mechanism must also be explained in the comments to the code.	10
Does the code output a confusion matrix demonstrating the performance of the Naïve Bayes classifier? The confusion matrix must clearly indicate the proportion of True Negatives, False Positives, False Negatives and True Positives.	5

Task 2 (50%): Improved solution

For Task 2, you are asked to identify, research and implement a way to improve on your solution to Task 1. Your choice of approach modification must be supported by sound arguments as to why you would expect the revised method to do better at the specific task of binary sentiment classification of text-based reviews.

In devising your revised method, you may either:

- identify an alternative classification algorithm, or
- apply modifications to the Naïve Bayes implementation: for example, changing the data representation by using n-grams (multi-word phrases).

Please note that in this task your proposed approach modifications evidencing independent research and thinking process will attract higher marks than going for the exact suggestion supplied above or other more straightforward extensions.

After implementing your modified approach, compare the results to the initial Naïve Bayes classifier from Task 1.

Implemented improvement assessment. It does not matter if your approach for Task 2 does not actually outperform the baseline approach in Task 1: you will not lose marks if the performance does not improve. However, you must consider and explain why this might be the case when you are addressing the final assessment criterion of this task (see below).

Task 2 will be assessed according to the criteria listed in the table below. For this part, full marks in each section will only be awarded to submissions that show extremely high levels of academic and technical proficiency and will require significant independent research and study.

Assessment Criteria for Task 2	**Marks Awarded**
Does the Jupyter notebook include a markdown/comment section that clearly explains how the approach taken in Task 2 is expected to improve on the solution to Task 1. Are the reasons for the expected improvements clearly justified and explained with one or more references (e.g. to a published source scientific paper, article, book)?	20
Does the submission clearly explain the steps that have been taken to implement the improved approach? These should be written in a way that one of your peers who may not have researched the same approach could understand. This may consist of code comments, Jupyter markdown, or a mix of both. Does the code implement the described approach appropriately (i.e. does the code actually do what is described?)? Is the code clear and of good quality?	15
Does the code output a new classification matrix for the "improved" (Task 2) approach AND Is there markdown or comment that clearly discusses and compares the performance of the classification approaches of Task 1 and Task 2 and further explains whether or not the expected improvements were achieved (and why this may be the case).	15