1. Chosen representation and data preprocessing

In the first stage, Pandas DataFrame is used for inputting and preprocessing the trg csv document, because of Pandas' convenience in visualising and manipulating the tabular data structure, especially for model training. We need to use sklearn package to do training/validation splitting, Pandas DataFrame integrates well with other Python libraries. Also, we split the document into X (abstract part) and y (class column) for further processing.

In the next step text preprocessing and data splitting are performed. Firstly, we identify all occurring different words. We count the total amount of all different words for further calculation, which is called |Vocabulary| in Mitchell, T. M. (1997). Moreover, we choose occurrence frequency for the representation of the text in X instead of 0-1 attributes, because occurrence frequency of each word is needed in the model training stage. This data processing procedure is written as a function, so it can be easily reused for handling other datasets in the future modelling steps.

Finally, we use sklearn library to do single 80/20 training/validation split because there are 4000 abstracts in the original data set, which is large enough. Also, more data for training would generate a better classifier, so I decided to use 80% of the raw data would be randomly chosen for the model training data.

2. The idea behind the model implementation and their improvements

In the second part in the code, we train a Standard Naive Bayse (SNB) classifier using the training dataset X_train and y_train. Our training model is based on the algorithm and three formulas provided in Mitchell, T. M. (1997). Machine learning. Firstly, in the training function, we use a formula to calculate the conditional probability of each word in the text: **P(wk|vj) = (nk + 1) / (n + |Vocabulary|)**, where wk is a specific word in the text, nk refers to the amount of times wk appears in the text, and n is total amount of different word with different target class. |Vocabulary| is same as the description above in the data preprocessing part. This formula uses Laplace Smoothing to avoid the very tiny probability and to prevent the situation where an unseen word has a probability of 0, resulting in the entire posterior probability becoming 0.

Secondly, we use another formula to calculate the Prior Probability for each target class: **P(vj) = |docsj| / |Examples|**, where Examples refer to the text with their target class, and docsj refers to the subset from Examples with their target class is vj, so P(vj) is the Prior Probability.

Finally, in the classify and prediction stage, we split a text into a list of word. We use the trained classifier to multiply the conditional probability of each word and prior probability for each class. Then we get the class of all abstract with the highest posterior probability in the document. The procedure will be discussed in the following part.

To improve my model, I concatenate some relevant words into one word. Also, according to Mitchell, T. M. (1997). Machine learning, and Rennie at al. (2003) "Tackling the Poor Assumptions of Naive Bayes Text Classifiers", the most frequent and common words are less likely to be relevant to the actual class of text. Therefore, I set the occurrence frequency of some most frequent words to 0, to make these words look as same as unseen or removed word.

3. Explanation of the evaluation procedure

In evaluation part, 20% of the raw data is randomly selected to be the validation data, named X_valid and y_valid which is utilised to test the SNB classifier. The dataset has been split into two subsets: a training set and a validation set in the data preprocessing part. The model has been trained on the training set to get the SNB classifer and will be evaluated on the validation set. This method provides an independent dataset for evaluating performance of the model and helps prevent overfitting.

To begin with, we write a function to calculate accuracy. Accuracy is the probability of correct predictions given the total size of samples, used to measure how well the SNB classifier perform. After we train a SNB classifier and get freqency for each word in each class, we use this classifier to get the conditional probabilities of all word. Afterward, we multiply the prior probability and the conditional probabilities of all word, using **VNB =argmax P(vj)πP(ai|vj)**. The result is posterior probability given each of the four classes for text or abstract in validation or test data. Then the predict function compares four posterior probabilities and selects the class with the highest posterior probability as the predicted result for the specific text. The prediction is saved in a list. We loop through y_valid which is the actual class of the abstract, and compare with the predicted class. Finally we get 94.375% accuracy for the initial SNB model.

4. Training/validation results for the standard and improved Naive Bayes model

The initial model is applied to the unseen Kaggle test data and luckily we get the accuracy or the public score of 0.953, which is a little bit lower than the threshold 0.96. Therefore, we have to improve our initial model to increase the accuracy.

To improve the model, I concatenated some correlated words including "homo sapiens", "escherichia coli", "human immunodeficiency virus" by adding "-" between these words. Following that, I also tried to use common sense to select some common words to remove like "an", "a" and "the". Next, I removed them from the classifier by setting their occurrence times into 0. After improving the model, the results of validation data show that the improvement did work with the accuracy increased to 95.5%. Thus, I believe that this improvement is a good idea. In addition, the improved model's performance on the Kaggel test data improve as well, increasing to 0.98 which was much higher than expected. Furthermore, I also tried to find the most frequent word in the document and the result is "the". However, I tried to only remove one word "the", the result on Kaggle was 0.973 accuracy which was slightly worse than removing three common words. Therefore, finally I decide to keep the model improved by removing "an", "a" and "the".

SNB Classifier	Validation data	Kaggle test data
Standard	94.375%	95.3%
Improved v1	95.5%	98%
Improved v2	95.5%	97.3%