THE HONG KONG UNIVERSITY OF SCIENCE & TECHNOLOGY
Department of Computer Science and Engineering
COMP4211: Machine Learning
Spring 2018 Assignment 1
Due time and date: 17:00pm, Mar 16 (Fri), 2018.

IMPORTANT NOTES

- **Your grade will be based on the correctness, efficiency and clarity.**
- **Late submission: 25 marks will be deducted for every 24 hours after the deadline.**
- **ZERO-Tolerance on Plagiarism: All involved parties will get zero mark.**

Q1. Soccer fans like to predict the outcome of a match by referring to history. Most people use heuristics to make a guess. A more scientific approach is to use data mining methods. The following table contains the past results for Brazil vs China:

#	Temp	Weather	Played in	Winner
1	Hot	Sunny	Europe	Brazil
2	Hot	Sunny	USA	Brazil
3	Hot	Rainy	Korea	China
4	Cold	Sunny	Korea	Brazil
5	Hot	Rainy	Europe	Brazil
6	Cold	Sunny	USA	China

Here, Temp can be Hot or Cold; Weather can be Sunny, Rainy or Cloudy; Played in can be Europe, USA or Korea.

Suppose that the next match will be played in USA on a hot, cloudy day. Using the naive Bayes classifier (with Laplace correction), what is the probability that the Chinese team will win? Show your steps clearly.

Q2. In this question, you will use the naive Bayes classifier for text classification with the *scikit-learn* toolkit. We use the *20 Newsgroups* dataset in `http://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups`.

You are asked to use the naive Bayes classifier to classify whether a document belongs to either *comp.sys.mac.hardware* or *sci.space*. You are required to write the code to do the following tasks in Azure Machine Learning Studio using Jupyter notebook (with Python 3).

1. In Azure, download all documents belonging to the categories of *comp.sys.mac.hardware* and *sci.space*;

   - You can use *sklearn.datasets.fetch_20newsgroups* to download the dataset (details in `http://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_20newsgroups.html`):

Parameters:	subset	train or test, all, optional. Select the dataset to load: train for the training set, test for the test set, all for both, with shuffled ordering.
	categories	None or collection of string. If None (default), load all the categories. If not None, list of category names to load (other categories ignored).
Attributes:	filenames	A list of filenames for each document
	target_names	A list of category names of the fetched dataset
	data	A list of text documents
	target	A list of category labels of the fetched dataset. Each element is corresponding to that in 'data' with same index.

2. Preprocessing and converting the text documents to document vectors:

   (a) Remove stop-words;

   (b) convert to lowercase;

   (c) extract the 1000 most frequent tokens as features;

   (d) use a binary feature for each token (1 if it appears, 0 otherwise);

   - You can use *sklearn.feature_extraction.text.CountVectorizer* (details in `http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html`):

Parameters:	stop_words	string {english}, list, or None (default)
	lowercase	boolean, True by default. Convert all characters to lowercase before tokenizing.
	max_features	int or None, default=None. If not None, build a vocabulary that only consider the top max_features ordered by term frequency across the corpus.
	binary	boolean, default=False. If True, all non zero counts are set to 1.
Methods:	fit_transform(raw_documents)	Learn the vocabulary dictionary and return term-document matrix.
	transform(raw_documents)	Transform documents to document-term matrix.

3. Build the naive Bayes classifier (with Laplace correction) using *MultinomialNB* in *scikit-learn*;

4. Perform 5-fold cross-validation, and report the accuracy of each fold in the following format:

```
Accuracy in fold 1: 100%.
Accuracy in fold 2: 98.75%.
...
Accuracy in fold 5: 98.5%.
```

- You can use *sklearn.model_selection.KFold* (details in `http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html`). You are required to set the *random_state* = 1234 to shuffle the data:

Parameters:	n_splits	int, default=3. Number of folds. Must be at least 2.
	shuffle	boolean, optional. Whether to shuffle the data before splitting into batches.
	random_state	int, RandomState instance or None, optional, default=None. If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random. Used when shuffle == True.
Methors:	split(X)	Generate indices to split data into training and test set.
Return:	[train_index, val_index] iterator	K consecutive folds train/valid indices to split data.

Submission Guidelines

You should submit:

1. Hardcopy of (i) **Q1**'s answer; and (ii) **Q2**'s codes and outputs. Note that your answers should be typewritten or handwritten properly and submit at the **beginning** of the tutorial.
   - Note that it should be clearly legible, otherwise marks will be deducted.
2. A python notebook of **Q2** (**ipynb**) to the Course Assignment Submission System (CASS): `http://cssystem.cse.ust.hk/UGuides/cass/student.html`. When multiple versions are submitted, only the latest version according to the timestamp will be used for grading. This should be submitted **before** the tutorial.