—

Classifying Textual Data

Learning outcomes

- Applying *k*-nearest neighbors algorithm (*k*-NN) to perform data classification.
- Measuring performance of a supervised learning model

Introduction

This homework is designed as a follow-up to the previous homework on "Descriptive Modeling and Clustering of Textual Data" (homework two).

In this assignment, you will apply a supervised learning algorithm to the textual data that you preprocessed in homework two. You will need to implement the *k*-nearest neighbor (*k*-NN) algorithm to find similar documents to the document in question and predict its label (i.e. topic).

You must write your own *k*-NN algorithm that takes as input:

- a **raw document** (not a vector), specifically the file name or path
- the **document TF-IDF matrix** along with the labels you determined previously, e.g. "Predictive Analytics", "Irma and Harvey", etc.

You may want to create a class or method that takes care of preprocessing a raw document into its corresponding TF-IDF vector representation. You can reuse the code from homework two.

See the chapter on data classification from the "Predictive Analytics" book to learn more about the *k*-NN algorithm. More discussion will follow in class.

It is highly recommended to use an Integrated Development Environment (IDE) such as Eclipse to be able to import the Java library with no issues.

Dataset

We provide a dataset of 10 unknown documents. Download the documents from class website under the link of this homework.

Tasks

The goal is to classify each of the unseen raw documents into the following three categories (from homework two):

- **C1**: Airline Safety
- **C4**: Hoof and Mouth Disease
- **C7**: Mortgage Rates

See below for the detailed list of tasks you should complete.

Implement the *k*-NN algorithm

Develop a routine (or method) that takes a raw document and a similarity measure as input, and returns as output the *k* most similar ("nearest") documents in the TF-IDF matrix, as well as their labels/topics (see above for the list of output categories).

The similarity measure should be either the Euclidean distance or the cosine similarity, based on which measure you discovered works best according to your homework two.

We will be testing your algorithm with more than just the three categories listed above, so please do not hardcode any of your results.

Measuring model performance

A simple way to measure the accuracy is to read/scan manually through documents and decide on the topics (based on what you read) then compare the assigned label by your classifier.

Experiment with different parameters for the *k* (i.e., the number of nearest neighbors) and report your findings.

(Bonus) Fuzzy *k*-NN

The supervised *k*-NN algorithm is very efficient, but only returns a single label, or topic, for each input document. As a bonus, you may implement a *fuzzy k*-NN classifier: instead of returning a single label, your algorithm will now return the percentages with which the document belongs to each of the output categories.

For example, instead of returning as label "**C4**" for an input document, your algorithm should now return that the document is x% about "**C1**", y% percent about "**C4**", and z% about "**C7**".

Submission

Deliverables

Your submission should consist of the following:

- All source code of your program, in Java.
- Any dependency files that your Java program relies on.
- A detailed README file describing how to execute your program from the command line. Any assignments that we are unable to run on a standard Unix terminal will be returned ungraded.
- A .txt or .pdf document that includes your best choice of *k*, the accuracy on the test set, and the confusion matrix.

Grading

Deliverables	**Points**
Preprocessing and vectorization	10
k-NN classifier	30
Model performance	10
(optional) Fuzzy *k*-NN classifier	10
(total)	**50 (+10 optional)**