Computer Science 3202/6915
Assignment 1 – Implement KNN

Goal

Implementing from scratch in Python3 the KNN classification algorithm. "Implementing from scratch" means that you need to write the necessary code to perform KNN (do not use scikit-learn KNN functions). You are allowed to use functions available in packages such as Pandas or NumPy.

Specifications

Your program should be called A1_KNN.py and it should run in Linux. You program should take three command-line arguments (given in the following order):

1. A filename specifying a tab-delimited plain-text file containing the data.
2. An integer indicating the number of closest neighbours (*k*) to consider.
3. An integer indicating the number of instances in the data to be set apart as unknown instances.

For example, we should be able to execute your program as follows:

```
$python3 A1_KNN.py data.txt 5 15
```

where the $ indicates the terminal prompt.

The data you will use is provided in Brigthspace (filename is Data4A1.tsv). This is a tab-delimited text file containing 400 observations. The file has a header. The first column is the observation ID and the last column is the class. There are 172 categorical attributes, and two classes.

Functionality

Your program should do the following:

1. Read the command-line arguments: name of the input file, number of neighbours to consider *k*, and number of unknown instances *U. k* should be in the range [1, number of observations in the file – *U*]*. U* should be less than the number of observations in the file.
2. Read the input data. You can assume the input file is in the working directory.
3. Randomly select *U* instances to be the unknown instances in UnInstance (refer to KNN Algorithm for Classification, slide 5 of Lecture 2).
4. Remove the unknown instances in UnInstance from the input data. The remaining instances are the training data *T.*
5. Perform the KNN Algorithm for Classification to predict the class of the unknown instances in UnInstance. Use a distance metric suitable for categorical attributes.
6. Calculate the proportion of correctly classified unknown instances. That is the number of unknown instances for which the actual class is equal to the predicted class divided by *U*.
7. Print in the screen the value of *k* followed by a tab (\t) and then the proportion of correctly classified unknown instances.

Computer Science 3202/6915
Assignment 1 – Implement KNN

Submission
Submit through Brightspace the following (one submission per team):

a) Your python code in a single file called A1_KNN.py
b) A PDF file containing:
   1. a brief description and justification of the distance metric you used,
   2. a brief explanation of the effect of *k* in the performance of your algorithm and a recommendation of what value of *k* to use (i.e., you need to test your program with different values of *k*),
   3. a brief explanation of how you tested your code to make sure your implementation was correct,
   4. an acknowledgement section listing your collaborations and online sources, and
   5. a program specification section listing the Python version and libraries you used.

c) Optional – bonus point: a PNG of JPG image with a plot showing the accuracy of your program as a function of the number of neighbours considered

Common pitfalls to avoid (i.e., DO NOT do the following or points will be deducted):

1. Submit files in a compressed file.
2. Hardcode filename of the input and/or values of the parameters (*k*, *U*).
3. Forget to test your program in linux (i.e., your program fails to run in linux).
4. Miss to include some of the sections in the PDF file.
5. Forget to acknowledge a collaboration or source.
6. Fail to follow specifications.
7. Use a distance metric unsuitable for categorical attributes.
8. Modify the input data (i.e., your program fails to run with the original data).
9. Hardcode the number of observations in the input data.

Online tutorials on how to implement KNN:

- https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/
- https://towardsdatascience.com/how-to-build-knn-from-scratch-in-python-5e22b8920bd2
- https://towardsdatascience.com/k-nearest-neighbor-classifier-from-scratch-in-python-698e3de97063
- https://nycdatascience.com/blog/student-works/machine-learning/knn-classifier-from-scratch-numpy-only/

To access LabNet:
From home go to https://remote.labnet.mun.ca/ and select “Linux General Access”.

For on campus access see https://www.labnet.mun.ca/ .