The University of Melbourne
School of Computing and Information Systems
COMP30027 Machine Learning, 2024 Semester 1

Project 1: Wine quality classification with K-NN

Due:	7 pm, 12 April 2024
Submission:	Source code (in Python) and written responses
Groups:	You may choose to form a group of 1 or 2. Groups of 2 will respond to more questions, and commensurately produce more implementation.
Marks:	The project will be marked out of 16 points (individual project) or 24 points (group project). In either case, this project will contribute 20% of your total mark.

Overview

In this project, you will implement a K-NN model to classify the quality of wines from their chemical properties. You will perform some analyses on a provided dataset and use these to answer conceptual questions. You will then have a choice of various ways to extend and evaluate the basic K-NN model.

Data

The data for this assignment is a modified subset of the "Wine Quality" dataset from the UCI Machine Learning Repository [1]. The original dataset can be found at `https://archive.ics.uci.edu/dataset/186/wine+quality`. The dataset consists of 11 chemical properties for 2700 wine samples (e.g., pH, density, citric acid, alcohol). The class label for each sample is `quality`, which has been coded as a boolean variable (0 = low quality, 1 = high quality).

Separate training and test datasets are provided, with a 50/50 stratified train/test split. Please use the provided train/test split for this assignment, unless a question asks you to create your own splits.

Figure 1: AI-generated image from stablediffusionweb.com

1. K-NN classifier [4 marks]

Implement a K-NN classifier that returns the labels for a set of test instances given a labeled training dataset and a parameter K provided by the user. Your implementation should work for any integer value of K $1 <= K <= N$ where N is the number of instances in the training dataset. For your base implementation, the K-NN classifier should:

- use Euclidean distance to measure the similarity between instances

- use majority vote to choose the label when K is greater than 1
- if majority vote results in a tie, tie break by taking the label of the 1-NN
- if there is a tie due to 2 or more instances having exactly the same distance, tie break by choosing randomly among these

Your implementation should assume all attributes are numeric. Your implementation should actually compute the distances between instances and identify the nearest neighbors – your implementation must not simply call a library function that does K-NN, like `KNeighborsClassifier` from `scikit-learn`. However, you may use library functions to read in the data, make visualizations, or analyse your results.

2. 1-NN classification [3 marks]

Do a 1-NN classification of the provided test dataset with the provided training dataset and K=1. Do not apply any normalization or scaling to the attributes. In your write up:

1. Report the accuracy of your 1-NN classifier. **[1 mark]**
2. Discuss whether or not this dataset is suitable for a 1-NN classifier – what aspects of the dataset make 1-NN classification easy or difficult? Support your answer by showing 2D scatterplots of the training dataset: pick two attributes to serve as x- and y-axes, and plot a random subset of the training instances on these axes. Use different colors or symbols to indicate the two classes. You can plot the data on any pair(s) of attributes that illustrate your points (you are not expected to show all possible pairs of attributes). **[2 marks]**

Your response should be no more than 250 words.

3. Normalization [5 marks]

Implement the two normalization methods shown in lecture:

1. Min-max scale all attributes to the range $0-1$: $x' = \frac{x - min(x)}{max(x) - min(x)}$
2. Standardize all attributes to have mean of 0 and standard deviation of 1: $x' = \frac{x - mean(x)}{stdev(x)}$

Do 1-NN classification with each type of normalization, and compare the results to your result from Question 2. How does normalization affect 1-NN classification performance? Answer in no more than 250 words, and support your answer by showing 2D scatterplots of the unnormalized, min-max scaled, and standardized training data.

Note that you should use the training dataset to get the parameters for these normalizations, and apply the same parameters to both the training and test instances (e.g., $min(x)$ refers to the minimum value of attribute x in the training dataset; use this same minimum value when scaling attribute x in the test set).

4. Model extensions [4 marks (individual) or 12 marks (group of 2)]

If you are working in a **group of 1, answer 1 of the questions** below for 4 marks. If you are working in a **group of 2, answer 3 of the questions** below for 12 marks. Each question response should be no more than 250 words and include figures and/or tables as appropriate.

1. Compare the performance of your best 1-NN model from Question 3 to a Gaussian naïve Bayes model on this dataset (you may use library functions to implement the Gaussian naïve Bayes model). In your write-up, state the accuracy of the naïve Bayes model and identify instances where the two models disagree. Why do the two models classify these instances differently? **[4 marks]**

2. Implement two additional distance measures for your K-NN model: cosine similarity and Mahalanobis distance (you may use library functions for these distance measures). Do 1-NN classification using each of these new distance measures and the three normalization options from Question 3. Discuss how the new distance metrics compare to Euclidean distance and explain how each metric is affected by normalization. **[4 marks]**

3. Implement either of the two K-NN weighting strategies discussed in lecture (inverse linear distance or inverse distance). Compare the performance of the weighted and majority vote models for a few different values of K. In your write-up, discuss how weighting strategy and the value of K affect the model's decisions. **[4 marks]**

4. Measure the empirical distribution of class labels in the training dataset (what percentage of the training data comes from each class). Then evaluate the distribution of labels predicted by your K-NN model for the test data, for a range of values for K. Does the class distribution of the predicted labels match the class distribution of the training data? Explain why or why not. **[4 marks]**

Implementation tips

K-NN can be computationally expensive, so it's a good idea to look for ways to make your code as efficient as possible. For example, doing matrix operations using `numpy` will generally be faster than doing operations with scalars in `for` loops.

Submission

Submission will be made via the LMS. Please submit your code and written report separately:

- Your code submission should use the provided .ipynb notebook template. Your submission must include comments or a README section that explain how to run your code so we can reproduce your results.

- Your written report should be uploaded separately as a .pdf, using the Turnitin submission link.

If you are working in a group, please include both group members' student id numbers on the written report and in your code file (in the README file or a group.txt file).

Late submission

The submission mechanism will stay open for one week after the submission deadline. Late submissions will be penalised at 10% per 24-hour period after the original deadline. Submissions will be closed 7 days (168 hours) after the published assignment deadline, and no further submissions will be accepted after this point.

Assessment

4 of the marks available for this assignment will be based on the implementation of the K-NN classifier (Question 1); no write-up is required for this question. The remaining questions (2-4) should be answered in a written .pdf report, and will be marked as indicated above. We will be looking for evidence that you have a correct implementation that allows you to explore the problem, but also that you have thought deeply about the data and the behaviour of the relevant classifier(s).

Because the number of questions depends on the group size, individual projects can receive a total of 16 marks and group projects can receive a total of 24 marks. In both cases, the project will contribute 20% of the final mark in this subject. In group projects, both members of the group will receive the same mark.

Updates to the assignment specifications

If any changes or clarifications are made to the project specification, these will be posted on the LMS.

Academic misconduct

You are welcome — indeed encouraged — to collaborate with your peers in terms of the conceptualisation and framing of the problem. For example, we encourage you to discuss what the assignment specification is asking you to do, or what you would need to implement to be able to respond to a question.

However, sharing materials beyond your group — for example, plagiarising code or colluding in writing responses to questions — will be considered cheating. We will invoke University's Academic Misconduct policy (`http://academichonesty.unimelb.edu.au/policy.html`) where inappropriate levels of plagiarism or collusion are deemed to have taken place.

References

[1] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009. ISSN: 0167-9236.