k-NN Binary Classifier for Cancer Patient Outcomes from Molecular Gene Expression

Prepared by Teighin Nordholt (20332323)

Submitted to Dr. Michael Korenberg for the ELEC 409 course

3 December 2024

Introduction

Genes are segments of an organism's DNA that control the expression of proteins [1]. Humans have approximately 20,000-25,000 genes, and their variation from individual to individual are responsible for differences in traits such as physical characteristics, susceptibility to diseases, or responses to medications [2]. The expression of specific genes can be quantified by qPCR or microarrays which allows researchers to investigate the role of individual or groups of genes [3].

Two groundbreaking studies published in the late 1990s and early 2000s demonstrated novel mathematical techniques for microarray analysis to respectively distinguish between cancer classes and predict treatment outcomes for cancered individuals [4], [5].

This report describes the reanalysis of Pomeroy, et al.'s data and development of a k-nearest neighbors (kNN) classifier to predict cancer patient outcomes. kNN is a computational technique that classifies novel data based on the distance to the kth nearest labeled training data in a multidimensional feature space. The objective is to evaluate the performance of the developed classifier in predicting cancer patient outcomes.

Methods

Python (v3.12.3) and scientific computing packages (NumPy v1.26.4, Matplotlib v3.9.1, scitkit-learn v1.5.1, SciPy v1.14.1) were used for all analysis. The dataset was obtained from the supplementary information of Pomeroy, et al.'s paper [5].

An overview of the workflow is shown Figure 1.

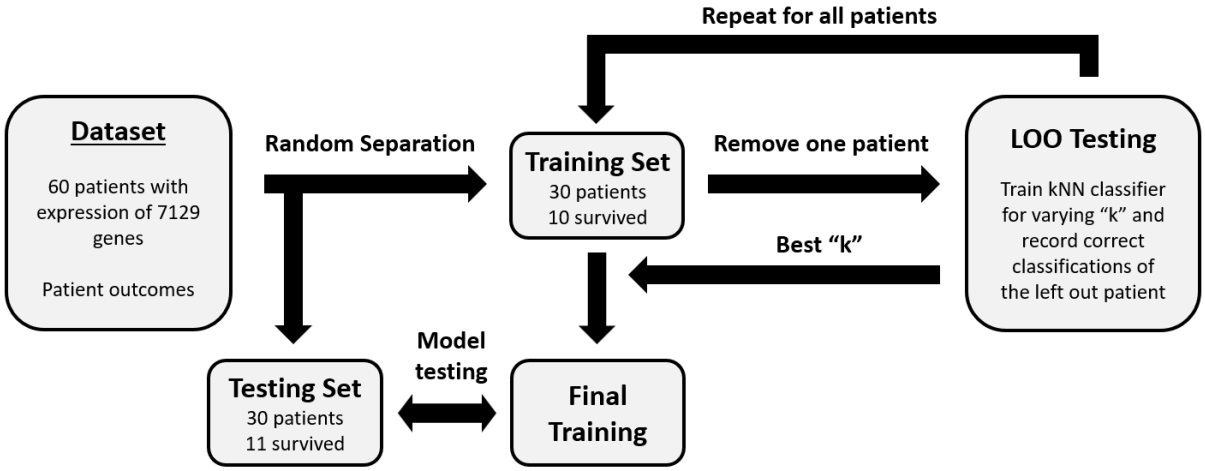

***Figure 1**: Workflow overview for data separation, leave-one-out testing, and final model training, and final model evaluation.*

The raw gene expression data is provided in a spreadsheet for all 60 patients and all 7129 genes. The data is read into Python arrays, and normalized as described in Pomeroy, et al. such that the mean and standard deviation for each gene is zero and one respectively [5]. The data is then randomly separated into training and testing datasets such that the training set contains 10 patients who survived and 20 who did not, while the testing set contains 11 patients who survived and 19 who did not.

Leave-one-out testing is used to automatically determine the optimal value of "k" to classify the dataset. One patient is removed from the training set, and a kNN classifier is trained on the remaining 29 patients for values of "k" ranging from one to 25. The weight of each of the "k" neighbors is weighed by the reciprocal of their distance to the test data, as in Pomeroy, et al. [5]. How often the classifier correctly predicts the left out patient is tracked for each "k", and the maximum of the sum across all left out patients is used to find the best fit of "k".

Finally, the kNN classifier is trained on the full training dataset with the optimal "k" found above. The 30-patient testing set is used to validate the performance of the classifier by constructing a contingency table. Contingency tables summarize the accuracy of a binary classifier, which is shown in Figure 2.

		True class	
		True	False
Predicted Class	True	TP	FP
	False	FN	TN

Figure 2*: Example contingency table. TP corresponds to true positives, FP corresponds to false positives, FN corresponds to false negatives, and TN corresponds to true negatives.*

Fisher's exact test shows the statistical significance of such tables and was implemented in Python using scikit-learn's "fisher_exact" function. The formula is given by Equation 1

$$p = \frac{\binom{TP+FP}{TP}\binom{FN+TN}{FN}}{\binom{TP+TN+FP+FN}{TP+FN}} \tag{1}$$

where

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} \tag{2}$$

p values smaller than 0.05 are generally understood to show statistical significance in the scientific community [6].

Additionally, Matthews correlation coefficient (MCC) is used, which similarly shows the significance of correlations in a contingency table. The formula is shown in Equation 3 and was implemented in Python using scikit-learn's "matthews_corrcoef" function. MCCs range from -1 to 1; values of -1 or 1 indicate perfect disagreement and agreement respectively, while values closer to 0 indicate little agreement between the true and predicted classes.

$$\phi = \frac{TP * TN - FP * FN}{\sqrt{(TP+FP)(FN+TN)(FP+TN)(TP+FN)}} \tag{3}$$

Finally, the results of the classifier were found to vary significantly depending on the initial randomization of the training and testing datasets, thus the classifier was retrained 50 times as described above for varying randomized datasets and the results were averaged.

Data and Code Availability

All code for this classifier was developed and tracked with GitHub in a repository cited here [7]. Package versions are listed above, though the repository also contains a ".yml" file to create a Python Anaconda environment with the required packages. The main script, "knn.py", uses the "data_reader.py" script to read, randomize, and separate the raw data, then trains the kNN classifier as described above.

Results

The kNN classifier was tested for 50 different randomizations of the training and testing data. Variation across iterations is visualized for Fisher's Exact Test and Matthews Correlation Coefficient in Figure 3 and Figure 4 respectively. The mean p-value of Fisher's Exact Test was 0.69 was standard deviation 0.34, while the mean Matthews Correlation Coefficient was 0.08 with standard deviation 0.15.

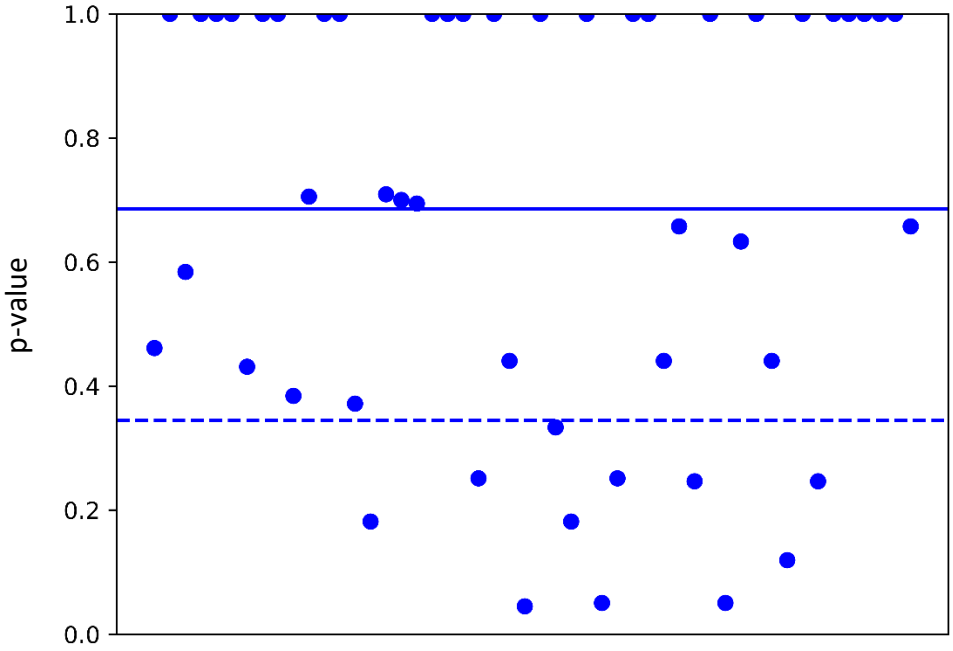

***Figure 3**: Variation of Fisher's Exact Test for 50 random initializations of the dataset. The mean is 0.69 with standard deviation 0.34. Smaller p-values indicate significance.*

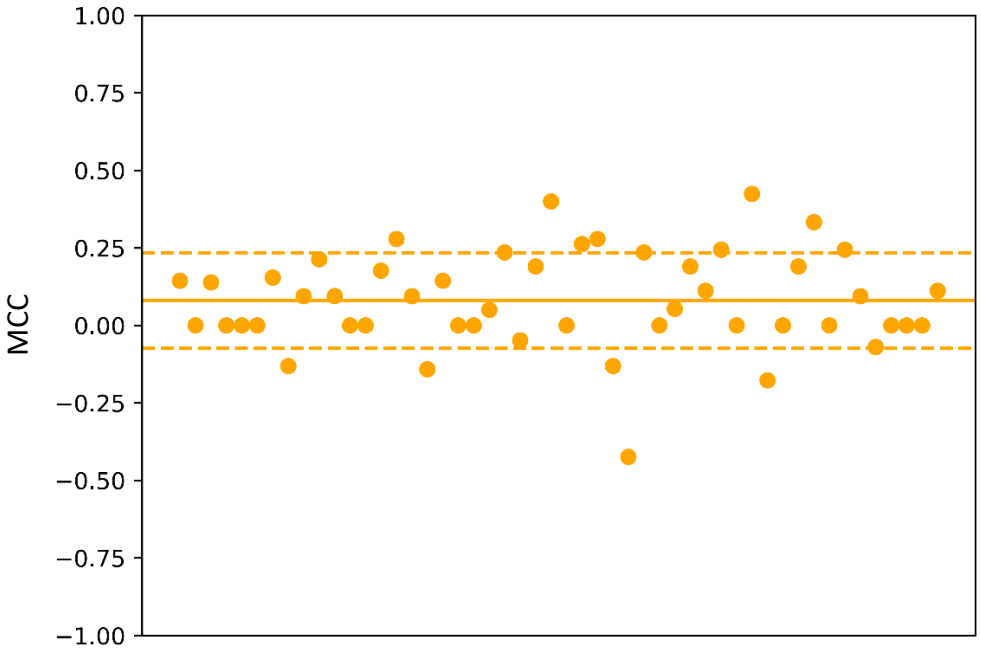

***Figure 4**: Variation of Matthews Correlation Coefficient for 50 random initializations of the dataset. The mean is 0.08 with standard deviation 0.15. Coefficients further from zero show positive and negative correlation.*

It was also found that the mean optimal "k" for the kNN classifier determined with leave-one-out testing was 9.18. The variation across iterations is visualized in Figure 5.

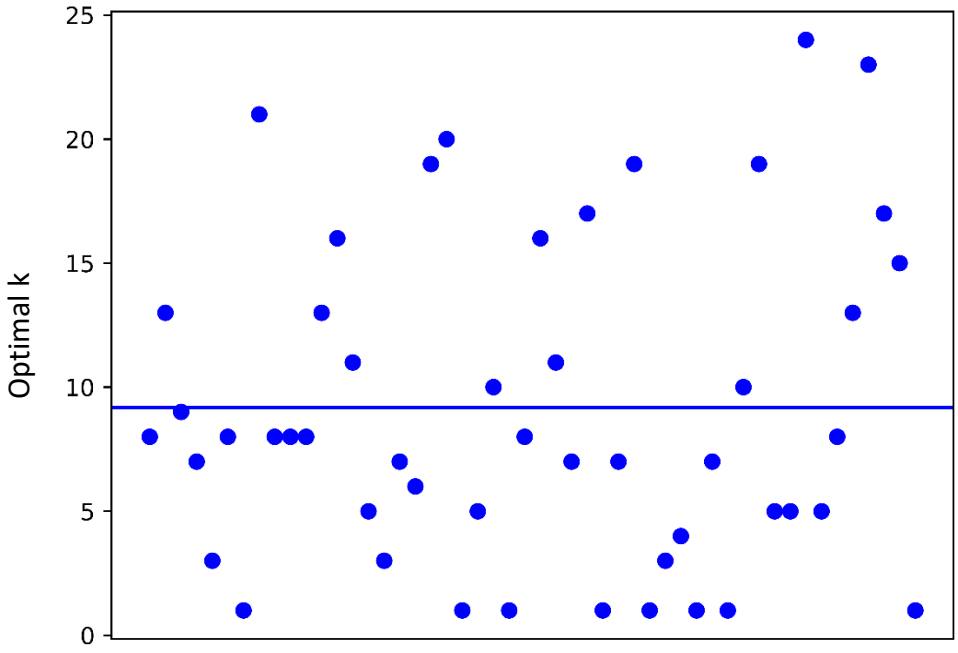

***Figure 5**: Variation of optimal "k" for the kNN classifier for 50 random initializations of the dataset. The mean is 9.18*

Discussion

Firstly, the mean result of the Fisher's Exact Test indicated that the results of the classifier were not significant. Additionally, the mean result of Matthews Correlation Coefficient showed that the correlation between the true and predicted outcomes of patients were not correlated more than a random classifier. The combination of these two quantitative metrics shows that, despite tuning the kNN classifier with leave-one-out testing, it is not effective in classifying the outcomes of cancer patients.

The mean optimized "k" value determined with leave-one-out testing was approximately 9, though it varied significantly with different random initializations of the training sets. This, in combination with the results of the statistical tests, indicate that the kNN classifier is not a stable prediction scheme for this specific dataset as its performance varies significantly.

The poor performance of the classifier is very likely a result of the dataset's small size. Literature suggests that the number of samples in a kNN classifier should be much larger than the number of features in the data [8]. This is not true for the classifier developed, whose data has 7129 features and only 30 samples in the training set. In the future, dimensionality reduction techniques like principal component analysis should be used to isolate genes that greatly contribute to the outcome of the patient.

Conclusions

A kNN classifier for cancer patient outcome prediction based on gene expression was developed. Leave-one-out validation was used to rigorously optimize the parameters of the kNN model, though it ultimately struggled to correctly classify patients. Its performance was quantified using Fisher's Exact Test and Matthews Correlation Coefficient, both of which indicated that the classifier performed no better than a random guesser. This is primarily due to the very small sample size of the training and testing data, which can be mitigated in the future using dimensionality reduction.

References

[1] “DNA vs Genes vs Chromosomes: An Overview,” Cleveland Clinic. Accessed: Dec. 03, 2024. [Online]. Available: https://my.clevelandclinic.org/health/body/23064-dna-genes--chromosomes

[2] “What is a gene?: MedlinePlus Genetics.” Accessed: Dec. 03, 2024. [Online]. Available: https://medlineplus.gov/genetics/understanding/basics/gene/

[3] H. Butz and A. Patócs, “Brief Summary of the Most Important Molecular Genetic Methods (PCR, qPCR, Microarray, Next-Generation Sequencing, etc.),” in *Genetics of Endocrine Diseases and Syndromes*, P. Igaz and A. Patócs, Eds., Cham: Springer International Publishing, 2019, pp. 33–52. doi: 10.1007/978-3-030-25905-1_4.

[4] T. R. Golub *et al.*, “Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring,” *Science*, vol. 286, no. 5439, pp. 531–537, Oct. 1999, doi: 10.1126/science.286.5439.531.

[5] S. L. Pomeroy *et al.*, “Prediction of central nervous system embryonal tumour outcome based on gene expression,” *Nature*, vol. 415, no. 6870, pp. 436–442, Jan. 2002, doi: 10.1038/415436a.

[6] H. Aguinis, M. Vassar, and C. Wayant, “On reporting and interpreting statistical significance and p values in medical research,” *BMJ Evid.-Based Med.*, vol. 26, no. 2, pp. 39–42, Apr. 2021, doi: 10.1136/bmjebm-2019-111264.

[7] T. Nordholt, *teighinnordholt/ELEC-409-Assignment*. (Nov. 20, 2024). Python. Accessed: Dec. 03, 2024. [Online]. Available: https://github.com/teighinnordholt/ELEC-409-Assignment

[8] N. Kouiroukidis and G. Evangelidis, “The Effects of Dimensionality Curse in High Dimensional kNN Search,” in *2011 15th Panhellenic Conference on Informatics*, Sep. 2011, pp. 41–45. doi: 10.1109/PCI.2011.45.