Comparison of Machine Learning Algorithms

Project Plan

Nathan Kurien
CS3821: BSc Final Project

Supervised by: Prof. Zhiyuan Luo
Department of Computer Science
Royal Holloway University of London

1 Abstract

Machine learning (ML) has taken the world by storm in recent times. In the era of unprecedented data generation, ML has emerged as a pivotal technology with profound implications for industries and everyday life. Supervised Learning algorithms exist to solve regression and classification tasks, and this project will focus primarily on classification. Within the ML landscape, classification algorithms play a critical role in tasks such as image recognition [1], spam filtering [2], medical diagnosis [3], and more.

Classification [4, 5] problems involve using an algorithm, and a given set of inputs used as trained features, to classify any given input with a predefined label. In other words, Classification problems involve searching for a discrete target value, often among a subset of discrete values, whereas Regression [5] problems involve finding a continuous target value.

This project will involve an implementation of the K-Nearest Neighbour [6] algorithm, widely known as one of the simplest [7] learning algorithms. This algorithm will run parallel with an implementation of the Decision Tree [8, 9, 10] algorithm, another relatively simple algorithm with graspable interpretability and easy possibilities for visualisation. Various tree implementations exist, but I intend to focus on the Classification variation of the CART algorithm [10], using Gini Impurity [11, p.309-312] as a goodness measure. Both algorithms will be evaluated and compared in their performance to tackle classification problems.

Focusing on these simplest examples of supervised learning classifiers, I aim to examine their application, performance and caveats. I hope to use performance metrics, such as accuracy, precision and F1 scores [12], to evaluate the bias and variance of these algorithms. I also hope to extend these evaluations by adding alterations to my algorithms and using more complex datasets. These metrics will be explored to also evaluate the bias-variance tradeoff [13] of these algorithms.

I plan to use model validation techniques such as hold-out tests and cross-validation [14] to determine these metrics. I aim to experiment with k-fold cross-validation [11, p.241-249] in particular, and establish the metrics mentioned during each iteration, and deduce the bias and variance present in the algorithms being assessed.

As a further study, I also plan to investigate the robustness of the K-Nearest Neighbours model by evaluating the consistency of performance over various folds during cross-validation. Within the context of decision trees, feature importance will be explored while investigating the decrease in impurity during validation of the model.

The models will ultimately be implemented in Python [15], creating the data structures using Lists and NumPy [16], and using various existing libraries as points of reference. Sci-kit Learn [17] is a highly regarded Python library by data scientists and includes various implementations of the models I hope to build. I will initiate research in this project by utilising this library to further my understanding of how these models function and how they should be evaluated.

Jupyter notebooks [18] will be used initially while the models are in development. They will also be tremendously useful for building visualisations and validating the models. Later on in development, I hope to implement these models in a standalone interface that can take datasets as inputs and provide useful insights on the performance of chose models on this data. Creating the interface will potentially be implemented using Python libraries such as PyQt5 [19]. There are vast amounts of resources I've found helpful in deciphering the theory behind this project.

I intend on using openly available datasets found on the UCI Repository [20], including the Delve datasets specified for Classification, and this may be sufficient to evaluate the performance of these chosen algorithms. If I choose to extend further, I could use datasets from open-source communities such as Kaggle [21], OpenML [22] and huggingface [23] for further training. Datasets from the latter are likely to require more preprocessing and normalisation.

Within this project, I hope to uncover and harness skills within data mining and machine learning that are proving invaluable and immensely relevant in the current climate of technological advancement. This project aims to delve into the core fundamentals of statistical learning and optimisiing such techniques to find valuable insights. Within industry, as investment into this field skyrockets, I believe that insights made in this project will be valuable in many applications. I am deeply grateful to be working on a project with such great relevance in the current atmosphere, and believe it will help me with my personal aims moving forwards.

2 Milestones

Listed below are a set of deliverables I hope to undertake in the process of putting together my project.

Along with reports and programs, I've included a success criteria section for both terms. I've noticed while planning this project that it's quite easy to get carried away with theory and dive deeper and deeper into algorithms without focusing on implementation. In the success criteria, I hope to clearly define what is essential to deliver in order for me to deliver a working project in due time.

2.1 First Term

2.1.1 Reports

Report 1: Nearest Neighbour Algorithms - In this report I hope to discuss the theory behind nearest neighbour algorithms, 1-Nearest Neighbour and K-Nearest Neighbour, and further details behind how they function.
Report 2: Decision Tree Algorithms - In this report, I hope to discuss the theory behind the Decision Tree algorithm and explore its measures of uniformity.

I sense that completing both of these reports will fulfill my understanding of the inner workings of these algorithms, and will supplement my interim report at the end of the year.

2.1.2 Programs

I will begin by implementing Proof of Concept programs in Python, starting very simple, and then expanding as I begin to expand both on the algorithms, and the complexity of my datasets. My PoC Programs will be listed as follows, in three sections:

Nearest Neighbours

- 1-NN algorithm implemented on very small synthetic dataset - in essence, coordinates on a 2-dimensional axis. The algorithm will essentially calculate Euclidean Distance between two points and set a solid starting point for development.
- 1-NN algorithm implemented on the infamous iris [24] dataset from the UCI repository [20], starting on two classes and a single feature, and then eventually implemented more features and classes from the dataset.
- Implement K-neighbours:
  - Starting with k=3 - and find differences with results from the iris dataset
  - Increase value for k, towards k=10.
  - Find cases when tie-breaking is necessary i.e. when k is even

- Introduce tie-breaking policies in the algorithm
- At this point, introduce performance metrics from PoC programs made in *model validation.*
- Also, the algorithm at this point should be able to handle larger and more complex datasets.

Decision Trees

- Start on simple synthetic dataset containing features and labels and build a very basic decision tree classifier to classify data points based on their features
  - Initially, a Decision Tree Node class will be created which should store information about the split criterion - which will likely expand as the algorithm is extended with more hyperparameters
  - A tree classifier class that follows the CART algorithm will recursively partition the dataset into various subsets until the optimum Gini impurity is found. This PoC program should explore how to do that.
- Visualisation - Before an interface is implemented, and while a library isn't being used, it would be wise to investigate how to visualise the tree using the console output.
- Once the program has progressed this far, overfitting will rapidly become an issue. Adding splitting criterion hyperparameters will be useful at this point. Adding a maximum depth will be my first step into implementing this.
  - There are a wide number of extensions I can go into, when it comes to reducing overfitting on trees, such as pruning techniques and implementing forests

Model Validation

- Start a program that can perform a train-test-split on a dataset, similar to that of Scikit-Learn
- Produce small programs that can calculate metrics such as accuracy, precision, recall and f-score after the training process.
- Implement K-folds Cross-Validation - program that can split dataset into various subsets and cycle through each data set for training + testing
  - The assessment phase during each iteration can be extended with various different metric calculations, including robustness for KNN and feature importance for DTs

2.1.3 Success Criteria

By the end of the first term, it is essential that I have implemented both algorithms, K-Nearest Neighbours and Decision Trees, in some manner using Jupyter Notebooks, as well as using any form of model validation techniques to determine performance metrics of both algorithms. This is what I'd call an essential deliverable before the interim review, and I think it's feasible with the time and preparation given, and necessary compromises made.

2.2 Second Term

My goals in the second term will become clearer as the term comes closer, but I've made my priorities clear below.

2.2.1 Reports

The priority in second term will be working on the final report. However, I think it's possible to also potentially make a report covering Model Validation Techniques and Algorithm Performance Metrics.

This will greatly depend on how much was delivered in the First Term and how much time can be allocated to more reports in the Second Term.

2.2.2 Programs

We make the heavy assumption that the First Term goals were delivered, the emphasis on the Second Term will then be to focus on delivering the algorithms on a standalone interface.

Assuming that the data structures are implemented end-to-end, second term may also allow some room to add extensions and further study into these algorithms with larger datasets, more performance testing and further algorithm optimisation.

2.2.3 Success Criteria

In essence, I hope to produce a fully object-oriented user interface that will provide meaningful evaluations of the model performances for both Nearest Neighbours and CART algorithms, at the very least. There's a hope that by this point, several extensions have been applied upon this - but this would be my bare minimum for a successful project delivered.

3 Planning and Time-Scales

Now that we've covered various deliverables and milestones to mark my progress with the project, this section will cover the projected timeline of when these items will be delivered, with a balance of caution and optimism.

3.1 Term 1 Timeline

Timeframe	Tasks
Week 1 (Oct 2 - 6)	Focus on implementing Project Plan Handwritten example of NN algorithm Test out Scikit-Learn classifier functions in Jupyter Notebooks
Week 2 (Oct 9 - 13)	1NN on simple dataset Design NN data structures and UML diagrams Begin PoC Programs. Fully test DTs and NN on Scikit-Learn
Week 3 (Oct 16 - 20)	1NN on iris data, Work on multi-class data Begin implementing K-Neighbours on Iris data Demonstate KNN in Jupyter Notebooks, Start basic DTs
Week 4 (Oct 23 - 27)	Complete NN Report - 29th October Start working on DTs on Iris Data with binary classification Begin working on hold out test set validation programs
Week 5 (Oct 30 - Nov 3)	Start DT Report Continue working on DT data handling with missing values Start working on cross validation programs and test on KNN
Week 6 (Nov 6 - 10)	Use hold out tests on Decision Trees Implement K-folds Cross Validation on both algorithms Continue working on report
Week 7 (Nov 13 - 17)	Complete DT Report - 18th November Deal with any issues that have arisen over previous weeks Use different datasets from UCI (e.g breast cancer)
Week 8 (Nov 20 - 24)	Focus on completing Interim Report Amend any issues with supplementary reports if not yet finished Draw meaningful tests and conclusions from
Week 9 (Nov 27 - Dec 1)	Work on Interim Report Prepare for Demonstration and Presentation Finalise project work + prepare for submission
Pres. Week (Dec 4 - 8)	Presentation Week! Submit project, and present work Make demo and presentation

3.2 Term 2 Timeline

It's rather challenging to build a detailed timeline for the next year, without context on the success of the first term, nor the progress made during the

Christmas break. To allow for contingency planning, I've kept planning for Term 2 to be very sparse and open to changes - with just a few deadlines in mind.

- Week 1 (9th January) - Start working on final report and review goals and progress
- Week 4 (3rd February) - Produce a working standalone user interface that's capable of visualising datasets that have been fitted in both algorithms
- Week 6 (16th February) - Complete any complementary reports made in conjunction (Performance Metrics/Validation Techniques)
- Week 8 (1st March) - Start work on posters and final demos

This timeline is admittedly open and sparse, but I see it as strategic in response to a very rigorous timeline for the first term. I'll be able to respond to feedback and issues made during the interim report in good time.

3.3 Weekly Reviews

I intend on reviewing my progress twice every week, once on Monday 8.30am and once on Friday 5.30pm. These reviews will allow me to keep track of progress and reassess my goals and targets for the upcoming week. These reviews will be made on my project diary.

4 Risks and Mitigations

Even best-laid plans often go astray, and in this section, I hope to list possible difficulties I may face as I undertake this project, along with contingency plans and mitigations in the hopes of reducing great losses.

4.1 Overemphasis on Theory and Research

The most apparent challenge I've found during the process of putting this plan together has often been the tendency to get lost in deciphering and understanding complicated, though interesting, theory behind the project's background - as well as looking into possible extensions before the core deliverables have even been put into place.

This challenge is moreso a feature of my own tendencies rather than the project itself, it is tempting to read and research rather than deliver what the project is demanding - and what the project is demanding is relatively simple compared to the various extensions that are possible.

I sense that the likelihood of this causing issues is high considering it's already had an effect on progress, I hope to mitigate this by reviewing my progress often during the weekly reviews and in my diary entries. I've found that the use of my diary to look at the overall scope of my project, as well as upcoming deadlines, has been useful in pulling me out of theory-induced rabbit holes.

4.2 Data Preprocessing Challenges

One of the most common and time-consuming challenges in any machine learning-related project is said to be related to data preprocessing and handling with datasets used in the project.

It's never clear how much time this may take, but it'll be necessary to mitigate this by simply allocating a reasonable amount of planned time to focusing on this, as well as ensuring that handling missing features in data is an aspect that is considered during the software design process.

For the sake of simplicity and sanity, I will initially focus my PoC programs on well-known and used UCI Respository [20] datasets, that will be likely easier to handle and manage - as well as many resources available if I find any issues. It's important that I focus on the core purpose of this project in evaluating these models, rather than waste time and resources handling data-related hiccups, so I will focus on simple datasets at first before diving into complicated datasets. Further into the project, I could also implement outlier handling in my project, though it may not be vital, it's something worth considering while designing my software.

4.3 Software Complexity

Building these algorithms from scratch can quite easily become a convoluted process without the correct approach applied.

I intend to use Object-Oriented Programming techniques where applicable to give the software a relatively clear architecture that can be built upon and extended effectively. I intend to create UML diagrams to outline the structure of the models, as well as how data is handled by the model. Where applicable, I hope to use unit test programs, as is standard in Test Driven Development, upon the models to keep track of functionality as more and more subroutines are added over time.

Making effective use of my GitLab repository will be helpful in staying on top of software complexity too, I will likely make use of branches when extending my algorithms' functionality.

4.4 GUI Challenges

Implementing a user interface to the project is likely to add complexity to the project and be quite time-consuming. Python GUI libraries can also occasionally come with their own hosts of compatibility issues.

Reflecting on the goals and scope of the project, I feel that implementing a user interface is not a priority matter and can be implemented further down the line - potentially even in the second term of the year. Delaying this implementation allows me to focus on the algorithms and their performance.

For now, PyQT5 will be the expected GUI library to be used, but TKinter [25] would be a decent backup just in case I have issues with the former option.

4.5 Data Availability and Quality

Assessing the functionality of these algorithms will require a sufficient amount of suitable data to work on for Classification tasks. Hence, it is critical that I find enough datasets to work on suitably.

For this reason, I've looked into open-source communities where contributors have made datasets publicly available, and have been thoroughly reviewed by other engineers. This may also be a suitable extension for me if I want to try stretching my models.

I honestly think the likelihood of this risk is quite low considering how much data is available online, but it seems to be a worthy thing to consider in the grand scheme of things.

4.6 Project Scope Creep

This is a highly extensible project. This project title allows for a large scope, and whilst this brings flexibility, it also makes it extremely easy to get carried away by extensions and added features - rather than focusing on the core deliverables and goals of the project.

This has quite a high likelihood of affecting my project - however I think it can be mitigated quite effectively by regularly reviewing my progress at the start and end of every working week, assessing my goals and targets and reminding myself of where priorities need to be made to deliver a final working product in time.

4.7 Time Management

The most obvious and self-explanatory of risks in any of these projects will be simply staying on top of time management. I hope that much of the mitigations mentioned in previous risks will supplement, what is likely, the most common pitfall for every student that embarks on such a project.

The weekly reviews and diary entries will be important for myself to reassess priorities and make amends before issues get bigger. It is also worth mentioning that if software complexity takes a toll, libraries are there to be used as a last resort. Simplifying the scope of the project is always an option whilst still delivering the core objectives of comparing algorithm performances

My first term timeline is intentionally rigorous such that a working deliverable manifests itself early. Conversely, my second term timeline is noticeably flexible - such that I can catch any issues that will surely arise prior to the interim report - and make alterations quickly.

The final thing I'd like to remind myself, is that it's always possible to ask for help - an oft-overlooked solution when there's desperate need for redirection.

References

[1] C. Li, X. Li, M. Chen, and X. Sun, "Deep learning and image recognition," in *2023 IEEE 6th International Conference on Electronic Information and Communication Technology (ICEICT)*, 2023, pp. 557–562. [Online]. Available: doi.org/10.1109/ICEICT57916.2023.10245041

[2] K. Tretyakov, "Machine learning techniques in spam filtering," in *Data Mining Problem-oriented Seminar, MTAT*, vol. 3, no. 177. Citeseer, 2004, pp. 60–79.

[3] A. Choudhury and D. Gupta, "A survey on medical diagnosis of diabetes using machine learning techniques," *Recent Developments in Machine Learning and Data Analytics: IC3 2018*, vol. 740, pp. 67–77, 2018. [Online]. Available: doi.org/10.1007/978-981-13-1280-9_6

[4] S. Kotsiantis, I. Zaharakis, and P. Pintelas, "Machine learning: A review of classification and combining techniques," *Artificial Intelligence Review*, vol. 26, pp. 159–190, 11 2006. [Online]. Available: doi.org/10.1007/s10462-007-9052-3

[5] D. M. Dutton and G. V. Conroy, "A review of machine learning," *The Knowledge Engineering Review*, vol. 12, no. 4, p. 341–367, 1997. [Online]. Available: doi.org/10.1017/S026988899700101X

[6] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967. [Online]. Available: doi.org/10.1109/TIT.1967.1053964

[7] H. A. Abu Alfeilat, A. B. Hassanat, O. Lasassmeh, A. S. Tarawneh, M. B. Alhasanat, H. S. Eyal Salman, and V. S. Prasath, "Effects of distance measure choice on k-nearest neighbor classifier performance: A review," *Big Data*, vol. 7, no. 4, pp. 221–248, 2019, pMID: 31411491. [Online]. Available: https://doi.org/10.1089/big.2018.0175

[8] S. K. Murthy, "Automatic construction of decision trees from data: A multi-disciplinary survey," *Data mining and knowledge discovery*, vol. 2, pp. 345–389, 1998.

[9] B. Charbuty and A. Abdulazeez, "Classification based on decision tree algorithm for machine learning," *Journal of Applied Science and Technology Trends*, vol. 2, no. 01, pp. 20–28, 2021.

[10] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees.* CRC press, 1984.

[11] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction.* Springer, 2009, vol. 2.

[12] A. Tharwat, "Classification assessment methods," *Applied computing and informatics*, vol. 17, no. 1, pp. 168–192, 2020.

[13] P. Domingos, "A unified bias-variance decomposition," in *Proceedings of 17th international conference on machine learning.* Morgan Kaufmann Stanford, 2000, pp. 231–238.

[14] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, no. 2. Montreal, Canada, 1995, pp. 1137–1145.

[15] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual.* Scotts Valley, CA: CreateSpace, 2009. [Online]. Available: https://dl.acm.org/doi/abs/10.5555/1593511

[16] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: https://dl.acm.org/doi/10.5555/1953048.2078195

[18] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt, Eds. IOS Press, 2016, pp. 87 – 90.

[19] PyQT, "Pyqt reference guide," 2012. [Online]. Available: http://www.riverbankcomputing.com/static/Docs/PyQt4/html/index.html

[20] D. Dua and C. Graff, "UCI Machine Learning Repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[21] Kaggle. (2023) Kaggle datasets. [Online]. Available: https://www.kaggle.com/datasets

[22] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "OpenML: Networked science in machine learning," *SIGKDD Explorations*, vol. 15, no. 2, pp. 49–60, 2013. [Online]. Available: http://doi.acm.org/10.1145/2641190.2641198

[23] HuggingFace. (2023) Hugging Face datasets. [Online]. Available: https://huggingface.co/datasets

[24] R. A. Fisher, "Iris," UCI Machine Learning Repository, 1988. [Online]. Available: DOI: https://doi.org/10.24432/C56C76

[25] F. Lundh, "An introduction to tkinter," *URL: www. pythonware. com/library/tkinter/introduction/index. htm*, 1999.