COMP551 - Assignment 1 Report

Abstract

In this assignment, we implemented and measured the performance of two machine learning classifiers, K-Nearest Neighbour and DecisionTree, as well as the effects of their hyperparameters on two benchmark datasets. It was observed that both classifiers performed relatively similarly (and with good results) in terms of ROC score, both in isolation, as well as when compared side by side. We were also tasked with loading, storing, and cleaning the data while following good experimental design practices to ensure performance was affected as little as possible, and that minimal bias was introduced into the results.

Introduction

Two datasets formed the foundation of our predictive models: *Heart Disease* from UC Irvine and *Penguin Dataset: The New Iris*. The former, a collection of real-world anonymous patient data from people examined for heart disease, was comprised of 13 features and a binary target value, indicating the presence (or lack thereof) of heart disease. The latter, a collection of penguin-related data based on the well-known *Iris* dataset[1], was comprised of 5 relevant features and a ternary categorical target value, indicating the penguin's species. The aforementioned features differed in data type from categorical to numerical of varying scales in both datasets, making both of them appropriate for use in clustering and regression-based machine learning algorithm[2][3].

The unprocessed nature of the data necessitated cleaning, feature scaling and manipulation, and analysis of comparative relevance in order to make them suitable for use in a machine-learning model. We were tasked with using these datasets to implement and compare two machine learning models: the K-Nearest Neighbour and Decision Tree strategies. Using these datasets, both models were trained, parameterized, and evaluated, through various experiments, in terms of accuracy achieved when predicting labels. Results were then compared across configurations to determine the optimal model with the highest predictive accuracy.

Various numerical analysis strategies were implemented and compared throughout the experimentation process. It was determined that the KNN strategy, selectively optimized to K=4 neighbours and the Hamming distance function, slightly outperformed the DT strategy, selectively optimized to a max depth of 8 and the gini index cost function, in predictive accuracy.

Methods

The K-Nearest Neighbour model relies on measuring the similarity between data points to classify unseen data under the same label as highly similar, known data. The model is an instance-based learner, as it does not explicitly learn a function during training but rather stores the training data in memory. In order to predict a label for an unseen data point, KNN computes the pairwise distance between the new data point and all seen training examples. From these computed distances, KNN selects the K nearest training points as the unseen data's closest neighbors. Class probabilities, the proportion of each label present in the K nearest neighbors, were computed, and the most frequently occurring label among them was assigned to the unseen data point. Various distance functions, namely Euclidean, Manhattan, and Hamming distance, were experimented with to transform the overall similarity across features into a numerical distance value. Euclidean distance, defined as the square root of the sum of squared differences between feature values, was ultimately selected as the most effective for numerical features. Additionally, for categorical features, Gower's distance was utilized, allowing for mixed-type data. Another important hyperparameter was the number of neighbors considered

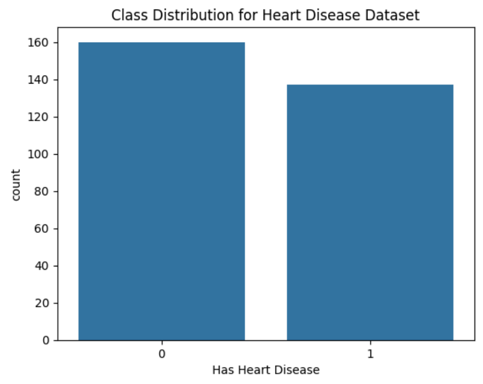
[1] P. Pandey, "*Penguin dataset - The new Iris?*" Kaggle, 2020. [Online].
[2] Anderies, A., Tchin, J. A. R. W., Putro, P. H., Darmawan, Y. P., & Gunawan, A. A. S. (2022). Prediction of Heart Disease UCI Dataset Using Machine Learning Algorithms. *JURNAL EMACS (Engineering, MAthematics and Computer Science)*, 4(3), 87–93.
[3] A. Agarwal and D. Yadav, "Comparative study of classification algorithms for iris dataset," *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, 2020, pp. 516-520, doi: 10.1109/Confluence47617.2020.9057891.

when making a classification decision. Small values of K result in a more flexible model, making more varied predictions on unseen data, whereas large values produce more rigidity in relation to the training data. Various integer values of K were tested to balance the model's rigidity with its adaptability. Through the validation process, in which the model was evaluated based on predictive accuracy on unseen validation data using a standardized random seed, the optimal performance was achieved with hyperparameters K=4 and Euclidean distance for numerical features, combined with Hamming distance for categorical features.

The Decision Tree model classifies unseen data by recursively partitioning the feature space into regions, each associated with a specific label. Unlike KNN, which relies on pairwise similarity, DT constructs an explicit decision function by learning rules that segment the data based on feature values, resulting in a tree structure. Training the DT involved iteratively selecting the best feature and threshold at each node to split the data into smaller subsets. The splitting feature and threshold are determined by a cost function, which relates the certain of the label predictions under a split to a numerical cost value. Three cost functions were considered during experimentation: misclassification rate, entropy, and the Gini index. Though they differ in implementation, the three functions coincide in that they yield a low cost under high confidence towards either the positive or negative prediction, and a high cost under low confidence, ie high uncertainty of a split. Through validation, the Gini index was found to yield the best predictive performance and was chosen as the primary cost function. To prevent overfitting, the depth of the tree was restricted. A deeper tree allows for highly specific decision boundaries but generalizes worse to unseen instances. However, a shallow tree is less sufficient in capturing the complexity of data, especially when training data is highly inseparable. Various maximum tree depths were tested, and the best results were obtained with a maximum depth of 8, which provided a balance between model expressiveness and generalization. Once trained, the DT classified unseen data points by traversing the tree from the root node to a leaf node, following the learned decision rules at each step. The label assigned to a leaf node was determined by the most frequent class among the training samples within that node.

Datasets

The two datasets consisted of a heart disease dataset provided by the University of Irvine, as well as a penguin species dataset hosted on the popular machine learning site Kaggle. For both datasets, we dropped observations that had at least one NaN value. For the UCI dataset, because of the way the observations and labels were loaded separately, the latter had to be filtered to match the observations. Also, the labels were converted to binary (0 if the value was 0, and 1 otherwise), for binary classification. Both datasets also had their continuous features centered and scaled to ensure that no one feature would have a dominating presence over the others when computing quantities such as similarity.



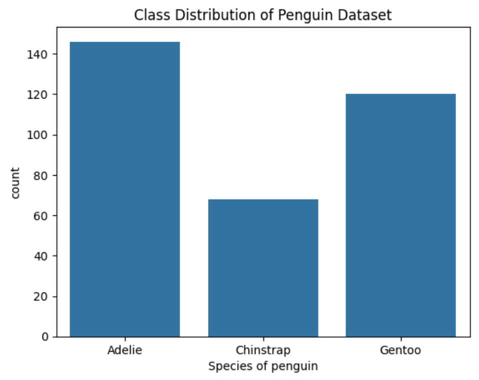


Figure 1 (left) and Figure 2 (right) show the class distribution of the heart disease and penguin datasets, respectively

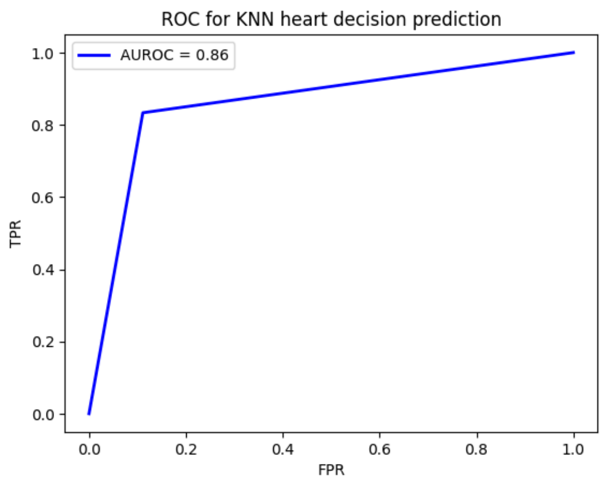
To help us understand the datasets, we also generated class distribution plots for both datasets (shown above), showing the number of patients diagnosed with heart disease vs healthy patients for the first dataset, as well as the number of each species of penguin for the second dataset.

We observe that the second dataset is not quite balanced in terms of having a roughly equal amount of all three penguin species.

Results

Experiment 1

In our first experiment, we tested our KNN and DT using default parameter values to get a general idea of the performance of each model by comparing the AUROC between KNN and DT on dataset 1 (binary prediction of heart disease). We trained and predicted the testing data with a test size of 0.2 using the default KNN hyperparameter K = 1 and Euclidean distance to get the K nearest neighbors for each test point. For the DT, we also used the default values of tree depth 3, the misclassification rate for class probabilities, and a minimum leaf instance of 1 as a requirement for a leaf node to be considered when constructing the DT. Our results showed that KNN performed better by the AUROC metric with a score of 0.86 compared to DT AUROC of 0.85.



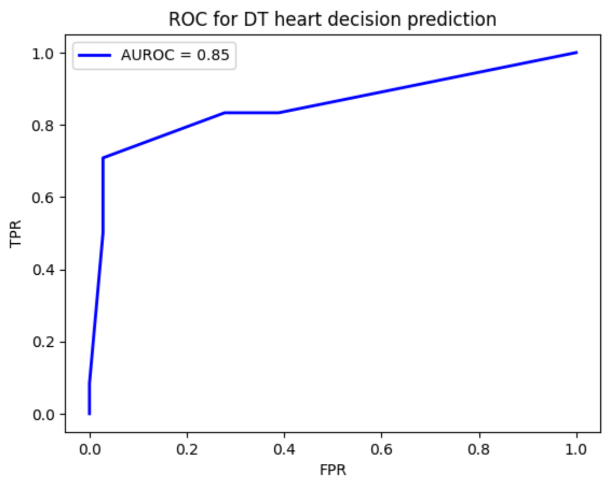


Figure 3 (left) and Figure 4 (right) show the ROC for the “default” KNN and DecisionTree classifiers, respectively

As for model accuracies on dataset 2 (species classification), the KNN model performed significantly better, predicting species with 100% average accuracy versus 94.0 % for the DT model.

Experiment 2

Next, we tried to improve the KNN by investigating how different values for k in KNN influence the test and train accuracy in both datasets. We experimented with different K values, ranging from 1 to 10, and used Euclidean distance as a control to measure how accuracy varies as K varies in KNN. For dataset 1, we saw the best test set results for K = 6 with an accuracy of 91.7%. In our second dataset, we observed K values 1, 3, 4, and 5, which gave a perfect accuracy score of 100% in both training and testing data.

Experiment 3

We also tried to improve the DT prediction performance by experimenting with various max tree depths. We used varying depths ranging from 3 to 8 on both datasets. For the first dataset, depths 6, 7, and 8 showed the best accuracy (90.0%), and for the second dataset, a depth of 8 showed the best accuracy (99.0%).

Experiment 4

In this experiment, we attempted to improve both our models by trying different distance functions for KNN and different cost functions for DT. In KNN, for the sake of consistency, we used the default K value 1 and tried the distance functions Euclidean, Manhattan, Hamming, and Gower. On dataset 1, KNN performed best with the hamming distance function scoring accuracy of 88.3%, euclidean and Manhattan performing equally at 86.7%, and Gower with the lowest accuracy at 60.0%. In DT, for consistency, we used max depth 3 and tried the cost functions misclassification rate, entropy, and gini index. The cost functions entropy and gini index performed best, both with a 90.0% accuracy, while the misclassification rate placed last with 76.7% accuracy.

Experiment 5

From our experiments above, we can get a feel for what combination of parameters might give us the best results. For KNN, k=4 seemed to be the most performant of the neighbour values, and hamming distance was the most performant distance function. For DT, gini index along with a max depth of 8 was the best performing (these results are on average and may vary from run to run). With these parameters, KNN achieved the better AUROC, scoring 0.94 compared to DT’s score of 0.91.

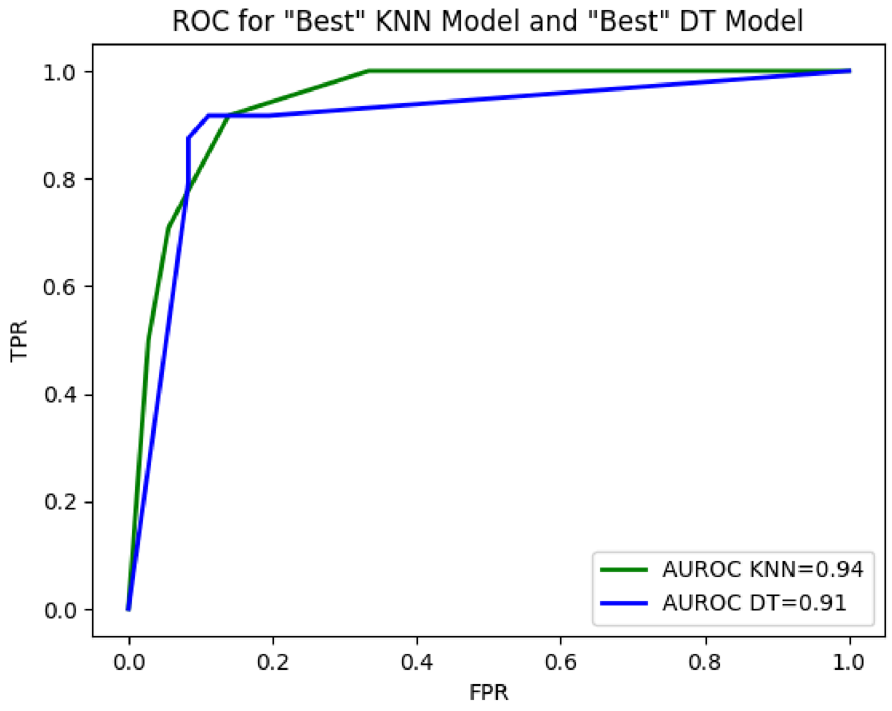


Figure 5: Comparison of AUROC between the best KNN classifier and the best DecisionTree classifier

Experiment 6

To decide which features to keep and which to drop with respect to the response for each dataset, we used cosine similarity (with a “drop threshold” of 0.5) as a measure of feature relevance to class labels. For dataset 1, we noticed quite a few features that had very little similarity to the response variable, so we kept everything with a higher similarity. The features we kept were sex, cp, restecg, exang, slope, ca, and thal. For dataset 2, we calculated the cosine similarity between the feature and each of the three species, then took the average of the three to get an estimate of how related the features were to the response overall. In this case, we had few (5) features to begin with, so dropping those with the lowest similarity to the response according to our similarity measure would leave us prone to underfitting, as we would struggle to capture patterns in the data with only 2 features. As such, we kept all the features in hopes that it would benefit our model when evaluating performance.

Experiment 7

To compare the results of different approaches to determining feature importance, namely squared mean difference and cosine similarity, we also computed feature importance using DT. By traversing through each node, testing if the node has children, and then keeping count of how many times each feature was used in each non-leaf node, we tracked the counts for each feature used in the DT for dataset 1. The top 5 features ranked from most to least used were slope, cp, restecg, exang, and sex. Interestingly, this coincided perfectly with the top 5 features aggregated from our cosine similarity calculations, while the top 5 features from the squared mean difference approach gave an entirely different set of features - fbs, thal, ca, thalach, and oldpeak. The difference in output wasn't surprising to us because the DT and squared mean difference approaches take completely different approaches. Whereas squared mean difference is a simple difference calculation of the means of features between the positive and negative groups, the DT approach uses different criteria like minimizing entropy or gini index costs to determine the best feature and threshold to use at each split, which accounts for threshold splits (which is insensitive to feature scaling), and feature interactions and dependencies. What was interesting to note however, was how the top 5 features from our cosine similarity calculation perfectly coincided with the top 5 features computed using the DT, suggesting that our choice of cosine similarity to determine which features to keep was a sound decision and that cosine similarity is a strong predictor of feature importance in our study.

Discussion and Conclusion

Our experiments clearly showed how different parameter inputs such as K and distance function for KNN, and max depth and cost functions for DT played a crucial role in maximizing the performance of our models. More specifically, we found that for KNN, the appropriate distance function (hamming) and K hyperparameter value (4) improved the AUROC score and prediction accuracy. For DT, the maximum depth (8) and the cost function (gini index) were also critical to maximizing its AUROC and prediction accuracy score. We conclude from our study that fine-tuning model parameters is a key step in creating the best model possible for classification tasks. Though our models showed notable improvements from using the best combination of parameters, for future investigation, it would be interesting to see how k-fold cross-validation and implementing weighted class probabilities could further improve our models.