K Nearest Neighbours and Decision Tree model performance and comparison

Group 47 - Preetesh Rambarun, Ben Cheung, Andrew Chao

Abstract—Machine learning is a dynamic predictive tool that provides unparalleled insight on data patterns using statistical analysis. In this assignment, we explore and build two simple, yet versatile supervised learning classifiers: K-Nearest Neighbour (KNN) and Decision Tree (DT). We implement a KNN model, including a weighted version, using different distance functions and a greedy recursive DT model using different cost functions. We test the models on two health-related benchmark datasets to compare their efficiency and accuracy. We observe that the overall testing accuracy of the KNN models is higher than that of the DT models for the two given datasets. Overall, we also notice a lower testing accuracy for the Messidor dataset in comparison to the Hepatitis dataset. We highlight that, in our implementation, a weighted version of the KNN model improves the prediction accuracy.

1 Introduction

Classification is a powerful machine learning technique. In the aspect of biological research, machine learning helps identify clinically important information from a large biological dataset to support healthcare-related decisions, such as disease prediction and detection. For instance, based on a patient's health data, such as age, gender, weight and family history, classification can predict whether the patient has a specific disease, giving them the opportunity to receive early treatment and preventive medication [1]. In this assignment, we consider two medical datasets - Hepatitis [2] and Diabetic Retinopathy Debrecen Dataset (Messidor Dataset) [3] - from UCI Machine Learning Repository.

1.1 Related Work

In this section, we mention two previous works related to the Hepatitis dataset and one related to the Messidor dataset.

The Hepatitis dataset has been used in research on composite nearest neighbour classifiers and one study [4] investigated whether classifiers from any given model class can be combined to create a composite classifier with higher accuracy. Past research introduced algorithms that combine some components of nearest neighbour classifiers, each storing some prototypical instances. After running the algorithms using several datasets, including the Hepatitis dataset, it was found that composite nearest neighbour classifiers that store a small number of prototypical instances can provide better classification accuracy compared to a normal KNN classifier.

The Hepatitis dataset has also been used in exploring lazy decision tree (DT) ensembles [5]. The lazy DT chooses a test that optimizes the resulting branch path taken by the given test data. By implementing a relevance-cased boosting style algorithm to build a lazy DT for each test data, a significantly improved base learner performance was obtained. With the addition of a new distance-based pruning method, overfitting caused by lazy DT is ameliorated, and the overall accuracy and comprehensibility of both Lazy DT and Boosted Lazy DT are enhanced.

Several studies investigate different ways to carry out feature selection and the Messidor dataset is an example set used in one such work [6]. In the highlighted work, the authors used the Curvature-based Feature Selection (CFS) method to rank the weights of all features on electronic health records from the Messidor dataset. The experimental results reveal that CFS achieved a significantly better performance against principal component analysis and other approaches.

2 Methods

Both KNN and DT are supervised learning models: the models are trained with labelled datasets in order to classify data and predict the labels of the test data.

2.1 KNN classification method

The basic principle of a KNN model works on the assumption that points falling into the same class are more likely to be found next to each other, hence the name ”nearest neighbours”.

For a test point, the labels of points closer to it exert higher influences on the prediction. The closeness of points is formally quantified by distances; commonly, Euclidean distance and Manhattan distance. There are also different regimes used to cast predictions. The unweighted KNN model treats every K-nearest neighbour as equally influential on the test point label, while a weighted version quantifies the importance of neighbours based on the distance to the target point by *similarity*. A careful choice of similarity and K-value can improve the model performance.

In our implementation, we consider three distance functions, two of which are canonical: Euclidean and Manhattan distance. We experiment with a weighted version of Euclidean distance, which can be viewed as assigning weights to features. By interpreting the weight function as an implicit assignment of weights, one simple candidate for the weight vector is the feature-label correlation vector. More precisely, if $x^{(m)}, x^{(m')} \in \mathbb{R}^n$ are the training points, we define the *correlation-weighted distance* by

$$d_w(x^{(m)}, x^{(m')}) = \sqrt{\sum_{i=1}^{n} w_i (x_i^{(m)} - x_i^{(m')})^2},$$

where $w \in \mathbb{R}^n$ is the feature-label correlation vector, with w_i being the correlation between the i-th feature and the label.

We also consider three modes of similarity: constant similarity (equivalent to unweighted KNN), inverse-distance similarity, and cosine similarity. To describe the role of similarity in making predictions more precisely, for a test point $x^{(*)}$, if $\mathcal{N}_{K,d}(x^{(*)})$ is the set of indices of the K nearest samples to $x^{(*)}$ with distance function d, the class probability $p_c = \Pr[y^{(*)} = c | x^{(*)}]$ is given by

$$p_c = \frac{\sum_{n \in \mathcal{N}_{K,d}(x^{(*)})} \text{sim}(x^{(n)}, x^{(*)}) \mathbb{I}(y^{(n)} = c)}{W_K(x^{(*)})},$$

where $W_K(x^{(*)})$ is the sum of similarity $\text{sim}(x^{(n)}, x^{(*)})$ over $n \in \mathcal{N}_{K,d}(x^{(*)})$. The prediction is made by choosing the label c which maximizes p_c.

- *Constant*: $\text{sim}(x^{(n)}, x^{(*)}) = 1$, this corresponds to the unweighted KNN;
- *Inverse-distance*: $\text{sim}(x^{(n)}, x^{(*)}) = \frac{1}{d(x^{(n)}, x^{(*)})}$. For the special case that $d(x^{(n)}, x^{(*)}) = 0$ for some n, the label of $x^{(*)}$ is dictated by the labels of points with zero distance by majority vote. This is, in principle, equivalent to adopting the convention $1/0 := \infty$.
- *Cosine*: $\text{sim}(x^{(n)}, x^{(*)}) = \frac{\langle x^{(n)}, x^{(*)} \rangle}{||x^{(n)}|| \cdot ||x^{(*)}||}$, where $\langle \cdot, \cdot \rangle$ denotes the usual vector inner product, and $||\cdot||$ denotes the Euclidean norm.

2.2 DT classification method

DT is another commonly used learning algorithm because it provides better interpretability and good handling of mixed-type (categorical and numerical) datasets. Compared to KNN which does not require any training (i.e. a lazy learner), DT is an algorithm that recursively selects its optimal parameters - feature, threshold, and tree depth - at every node. DT uses cost functions to find these optimal values - most homogeneous branches or branches having groups with similar responses. Splits with minimized costs are favoured and the algorithm eventually stops as soon as it achieves minimum cost or reaches a terminate condition created to limit the computational time.

In our implementation, we build a DT model that performs dataset split by using a greedy search to minimize a cost function. We consider three cost functions: misclassification rate, entropy (a measure of uncertainty), and Gini Index to determine the split at each node. We remark that the samples are inputted without standardization and normalization since DT splits only consider the ordering of numerical values.

3 DATASETS

3.1 Data inspection

There are 155 instances in the Hepatitis dataset and 1151 instances for the Messidor dataset before data cleaning. Both contain a total of 19 categorical and numerical attributes; all categorical features are binary. Only the Hepatitis dataset contains missing values - 80 instances.

3.2 Data processing

We load the raw data into Pandas dataframes and use the titles of the features as column headers for better interpretation. The class label of both classes is named "CLASS". We record categorical or numerical types by a Boolean array "feature_type", and convert the data to numerical values. The missing values (indicated by "?") are coerced to "NaN" using the built-in Pandas numerical conversion method. Having no prior knowledge about the Hepatitis dataset, we decide remove any instance containing missing values in order to prevent biases towards certain features.

3.3 Data analysis

We perform statistical analysis on each dataset to extract distribution trends. For each categorical feature, we identify the label with the highest frequency. For numerical data, we calculate the mean, standard deviation, maximum and minimum of the data. We observe that the Hepatitis dataset has a small sample

size (80) and high data imbalance: there are 13 positive instances (class label 1, indicating "die") and 67 negative instances (class label 2, indicating "live"). On the other hand, the Messidor dataset is more balanced with a sufficient number of samples: 611 positive samples and 540 negative samples.

3.4 Data split

For the two datasets, we split the data into 80% for training and 20% for testing. Since the Hepatitis dataset is small, we perform a 5-fold split to effectively use the data for training and validation. For the Messidor dataset, we perform another split on the training set for validation purposes; we use 80% for training and 20% for validation.

3.5 Numerical pre-processing

Since KNN models are sensitive to feature scaling, we apply standardization and normalization in preparation for KNN training specifically. To prevent data leakage, standardization or normalization is done separately on the split training and testing sets.

4 Results

4.1 KNN models

We test our KNN model with a range of K values and our suggested distance functions. Table 1 summarizes the best K-value and testing accuracy for each configuration. We refer readers to the Jupyter Notebook file for the full graphical results.

TABLE 1
Distance function and testing accuracy results

Dataset	Distance function	Best K	Accuracy (%)
Hepatitis	Euclidean	10	81.2
	Euclidean (Inverse)	7	87.5
	Euclidean (Cosine)	5	100
	Manhattan	9	81.2
	Manhattan (Inverse)	9	81.2
	Manhattan (Cosine)	1	87.5
Messidor	Euclidean	21	65.8
	Euclidean (Inverse)	36	64.5
	Euclidean (Cosine)	30	65.8
	Manhattan	21	65.8
	Manhattan (Inverse)	24	65.4
	Manhattan (Cosine)	24	67.5

For the Hepatitis dataset, we notice the outstanding 100% testing accuracy with cosine-similarity and Euclidean distance at $K = 5$. However, after multiple runs with different random splitting, we believe that this "perfect" outcome is likely to be a mere coincidence due to randomness and the small size of the dataset. Beside this anomaly, the testing accuracy achieved is around 80% in any other case. Based on our results, a weighted KNN generally improves the prediction accuracy. Moreover, the optimal K-values for the weighted versions are reduced compared to the unweighted version, which can imply a decreased computational time.

The Messidor dataset, however, shows a different trend in the accuracy and optimal K-value. For this dataset, there is no significant boost in the overall accuracy despite the use of weights. In fact, the only improvement occurs from using cosine similarity on Manhattan distance. For both Euclidean and Manhattan distances, the testing accuracy achieved by inverse-distance similarity is reduced by around 1%.

4.2 DT models

Using misclassification, entropy, and Gini Index as cost functions, we test the optimal maximum tree depth and obtain the testing accuracy, as summarized in Table 2.

For the Hepatitis dataset, the best accuracy, 81.5%, is obtained by a tree with max-depth 2 and misclassification cost. The shallow optimal tree depth of 2 for this dataset may be explained by the relatively small dataset size. In contrast, the Messidor dataset requires a greater tree depth and the best accuracy achieved is lower (67.1% using Gini Index). We remark that altering the minimum number of leaf nodes does not result in significant changes in the tree depth or testing accuracy for both datasets. We also point out that Gini Index has a poorer training and validation accuracy on the Hepatitis dataset, while it provides the best training and validation accuracy on the Messidor dataset.

TABLE 2
Max tree depth and testing accuracy results

Dataset	Cost function	Max depth	Accuracy (%)
Hepatitis	Misclassification	2	81.5
	Entropy	1	75.0
	Gini Index	5	68.8
Messidor	Misclassification	4	58.4
	Entropy	5	60.2
	Gini Index	9	67.1

4.3 Feature selection and decision boundaries

To illustrate the decision boundaries, we select a pair of important numerical features for each dataset. We evaluate the feature importance by LinearRegression for the KNN model and RandomForestRegressor for the DT model. Both of these built-in functions from sklearn output a score for each feature, and a higher score reflects a greater significance of the feature for

the corresponding model. Figure 1 shows an illustration of KNN feature importance for Hepatitis feature selection, and Figure 2 presents DT feature importance for the Messidor dataset.

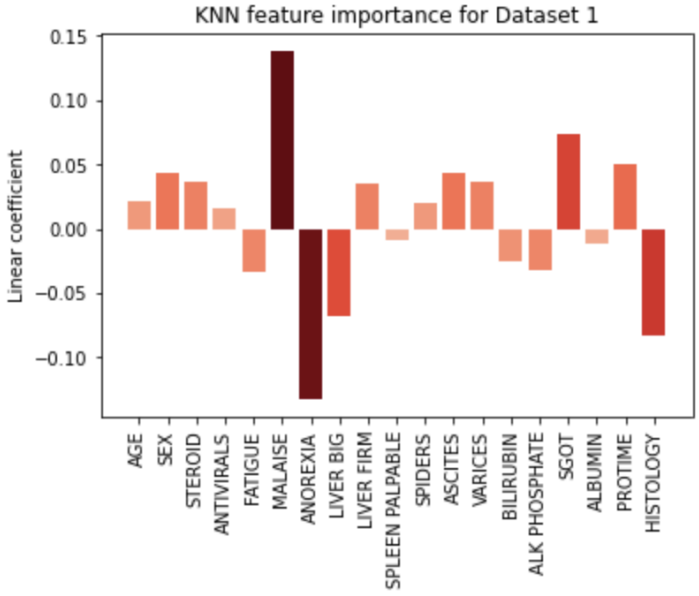

Fig. 1. KNN model feature importance results for Hepatitis dataset

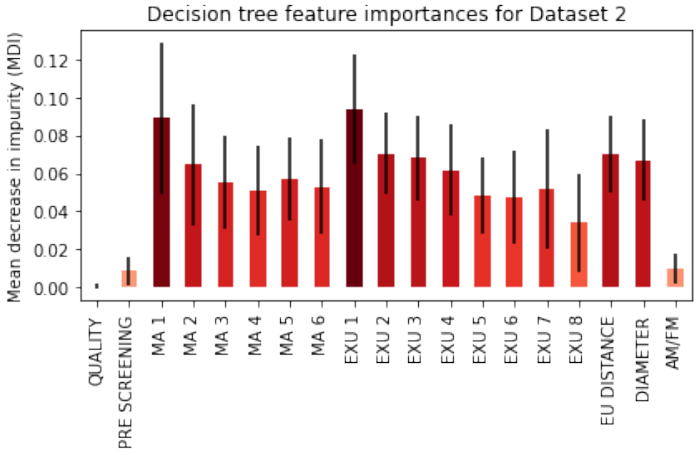

Fig. 2. DT model feature importance results for Messidor dataset

For categorical features, the intermediate values between class labels are not meaningful, and thus decision boundary plots involving categorical features are not directly interpretable. For this reason, we choose two sufficiently important numerical features, reflected by either the feature importance scores or the dataset statistics. For instance, while the absolute LinearRegression scores of the features "MALAISE" and "ANOREXIA" are the highest for KNN on the Hepatitis dataset, we do not choose these features for the decision boundary plots because of their categorical nature.

Figure 3 shows the KNN decision boundaries with $K = 5$ and Figure 4 the DT decision boundaries with max-depth 6, using Hepatitis dataset. As a general observation, a majority of data points can be accurately classified by the decision boundaries for DT as max-depth increases. However, since the decision boundaries only work well with small K values for the KNN model, we notice that the regions are no longer clearly defined as K increases.

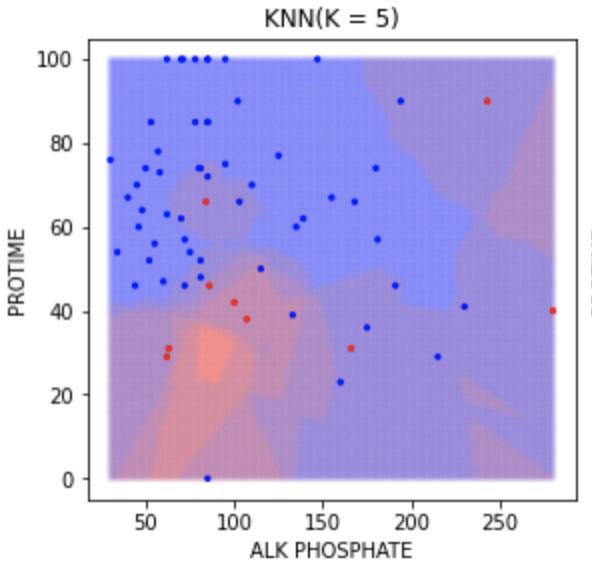

Fig. 3. KNN Decision boundaries for Hepatitis dataset

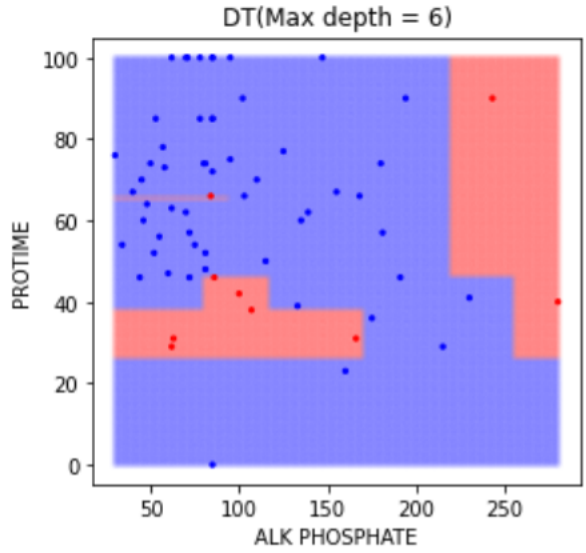

Fig. 4. DT Decision boundaries for Hepatitis dataset

4.4 Model evaluation and comparison

As mentioned before, the Hepatitis dataset is small and imbalanced. Hence, we choose to plot the precision-recall (PRC) curve and compute AUPRC for both models with various K values against a random prediction. We find that for the KNN model, as the

K value increases from 5 to 20, we find that K = 10 and 15 give the best AUPRC with 0.38. For the DT model, as the tree depth increases from 3 to 15, AUPRC is increased from 0.18 to 0.36. Both cases show a definitive difference from the performance of a random classifier. Figure 5 demonstrates the PRC curve for both models on Hepatitis dataset.

For the Messidor dataset, as it is a larger and more balanced dataset, we evaluate the training performance with both ROC and PRC. For the ROC and AUROC, we find that as K increases from 5 to 25, the AUROC for KNN model does not change by much (from 0.66 to 0.69), on the other hand, for the DT model, we find that an increased max-depth from 3 to 23 improves the AUROC (from 0.54 to 0.63). The complete results are attached in the code.

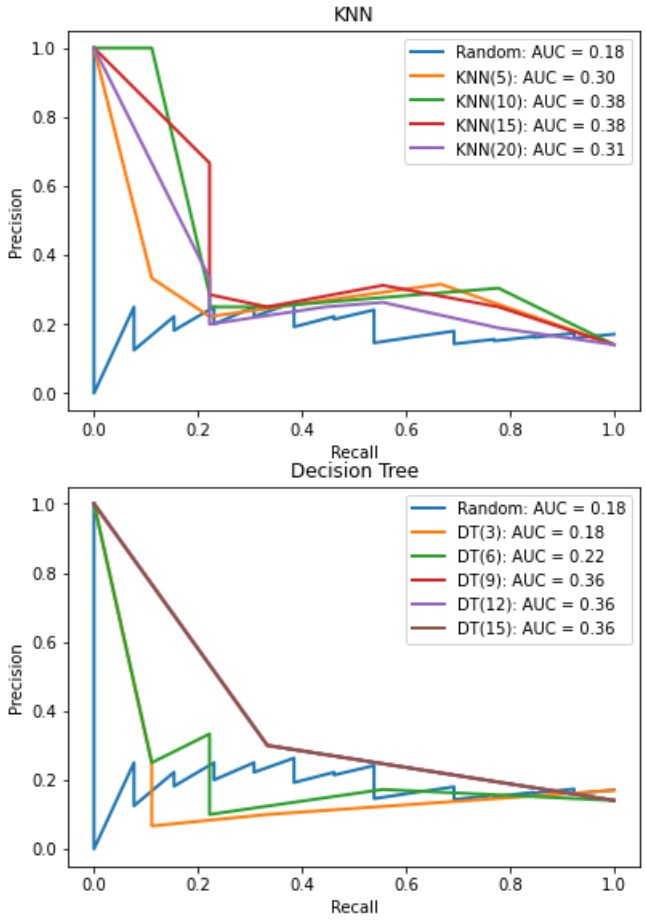

Fig. 5. PRC results for the two models on Hepatitis dataset

5 Discussion and Conclusion

5.1 Effect of standardization and normalization

From our experiments, we notice that standardization is more suitable for the Hepatitis dataset while normalization is slightly better for the Messidor dataset in training with KNN. As mentioned above, we run the learning algorithms without performing numerical pre-processing for DT training, due to its scale-insensitive nature. This aligns with the general machine learning principle of choosing a suitable pre-processing method on a case-by-case basis.

5.2 Effect of function choices

We observe that the choice of distance functions (for KNN) or the choice of cost functions (for DT) varies by much on different datasets. We also notice that the weighted KNN enhances the performances for the Hepatitis dataset but generally degrades the performance for the Messidor dataset. This reflects that the optimality of function choices for learning algorithms are case-specific.

5.3 Decision tree and overfitting

We observe that increasing the tree depth improves the training accuracy for both datasets as expected. However, deepening the tree depth too much can result in overfitting and poor generalizations. This is also evident from the experiments that the training accuracy converges to 100% up to a certain depth, while the validation accuracy fluctuates around the same range away from perfect predictions.

5.4 Feature selections

We perform several experiments to test for the key features. One of the tests is to drop one feature at a time and inspect the testing accuracy of models on the modified dataset. We find that the removal of the features does not result in a significant change in the testing accuracy (within 2% in most cases), and this observation also applies to features known to have high feature importance scores, such as "PROTIME" identified in the Hepatitis data. We refer readers to our code for a more detailed analysis.

5.5 Future directions

One possible direction for investigation is to explore other data pre-processing techniques, such as linear regression, to interpolate the missing values or dropping only the instances that are statistically irrelevant. Retaining more samples can potentially increase the overall accuracy, especially for small datasets like the Hepatitis dataset with limited valuable data.

6 Statement of Contributions

All members contributed to writing the report and discussing ideas. Ben implemented the data cleaning, prepossessing, and set up the KNN and DT models. Andrew implemented feature selection functions and contributed to model evaluation. Preetesh performed related works research, literature review, and data and model results analysis.

REFERENCES

[1] Nayyar, Anand, Lata Gadhavi, and Noor Zaman. "Machine learning in healthcare: review, opportunities and challenges." Machine Learning and the Internet of Medical Things in Healthcare (2021): 23-45.

[2] C. L. Blake and C. J. Merz, "UCI repository of machine learning databases," University of California, Irvine, Dept. of Information and Computer Sciences, 1998, http://www.ics.uci.edu/learn/MLRepository.html

[3] Antal, Bálint, and András Hajdu. "An ensemble-based system for automatic screening of diabetic retinopathy." Knowledge-based systems 60 (2014): 20-27.

[4] Prototype Selection for Composite. Nearest Neighbor Classifiers. David B. Skalak. Department of Computer Science. University of Massachusetts.

[5] Fern, Xiaoli & Brodley, Carla. (2003). Boosting Lazy Decision Trees. ICML. 2003.

[6] Zuo, Zheming & Li, Jie & Al Moubayed, Noura. (2021). Curvature-based Feature Selection with Application in Classifying Electronic Health Records.