COMP 551 Mini Project 1 : Getting Started with Machine Learning Prediction of Breast Cancer and Hepatitis

Alex Le Blanc, Miguel Ibanez, Yang Chen
{alex.leblanc,miguel.ibanezsalinas,yang.chen4}@mail.mcgill.ca
05 Feb, 2021

Abstract

In this project, we investigated the performance of two machine learning classification models, namely decision trees and k-nearest neighbours (KNN), using two benchmark medical datasets (breast cancer and hepatitis). First, in order to gain a greater understanding of the data, we performed careful analysis of the features and classes. We then tested the effect of additional features arising from interactions between existing features and the effect of cost/distance functions as well as modifications of these functions in the accuracy performance using cross-validation. Ultimately, we found that KNN achieved higher test accuracy than decision trees for both datasets. Moreover, we found that misclassifcation cost produced the best accuracy for decision trees and that for KNN, our variant of Euclidean and Manhattan distance performed similarly to each other, but significantly better than the standard metrics.

1 Introduction

Breast Cancer and Hepatitis are common and dangerous diseases with enormous personal and societal costs. We implemented decision tree and nearest neighbors algorithms for disease diagnosis (classification task) on two medical datasets, Breast Cancer Winconsin and Hepatitis(Dua and Graff, 2017). These relatively simple non-parametric methods are easy to implement, work with small amounts of data, and can produce predictions with high accuracy. In our experiments involving the decision tree algorithm, we studied the effect of the maximum depth, minimum number of instances per leaf and cost functions on the model performance (accuracy). Similarly, for nearest neighbors algorithm, we evaluated the model with different values for the number of neighbors to the query point during prediction. Similar experiments have been performed on the same datasets. Using decision tree, Ratanamahatana(Kohavi, 1997) achieved 92.63% accuracy on the breast cancer data, Fern(Fern and Brodley, 2003) had 85.87% on the hepatitis data, while Garcia(Garcia et al., 2012) received 96.2% and 77.3%, respectively, on these two datasets using 1-NN.
Our accuracy results are similar to those previously reported. Based on our experiments, we recommend using KNN for disease prediction. Nevertheless, we investigated interesting modifications to popular cost functions, distances metrics and extra features.

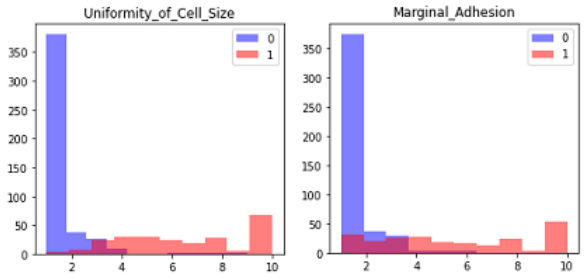

Figure 1: BCW Class Distribution

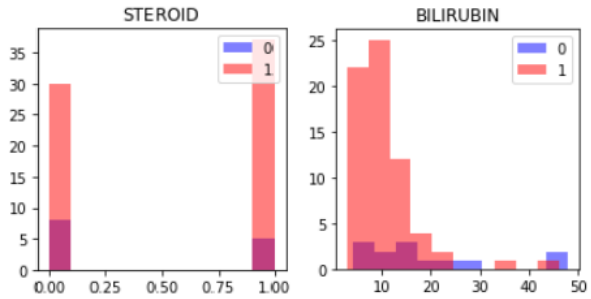

Figure 2: HEP Class Distribution

2 Datasets

We define our main research questions (presence of breast cancer or hepatitis) as classification tasks. We analyzed two multivariate sets: Breast Cancer Wisconsin (BCW) Data Set, presenting nine numerical attributes (1-10 scale) measured from images of breast mass cells (cell nuclei) with target classes Benign (65.5%) / Malign (34.5%) and totaling 699 cases; and Hepatitis (HEP) Data Set, presenting 19 mostly medical diagnosis related attributes - 12 binary variables (e.g., presence of steroid), five numerical variables (e.g., Bilirubin level), age and sex - with target classes Die (20.6%) / Live (79.4%) and totaling 155 cases. We found 16 missing values in the BCW set and 75 missing values in the HEP set. For experimentation,

instances with missing values were excluded, which reduced the HEP set to almost half the original size. For BCW, attributes' boxplots display some presumed outliers, particularly affecting Mitoses. As all values are within the expected range (1 to 10), we do not exclude any. For the BCW set, using boxplot and histogram visualizations by class, we can notice that, for all attributes, the malign class corresponds to higher values than those observed in the benign class (variables are positively correlated to the class)(see figure 1). In addition, most variables are positively and moderately correlated among them. Due to this, we can posit that learning the class labels is feasible.

For the HEP set, we can observe a high level of overlapping in the distributions by class for the numerical variables (see figure 2). We posit that distinguishing between classes would be more challenging. In both datasets, the class labels are not balanced; however, the smaller class label is at least 20% of the total, which is substantial enough to continue with the analysis.

Based on our review, we constructed extra features to help the learning process. For the BCW set, we added two variables that capture the interaction between measures at the cell and the nuclei level: "Clump Thickness (cancerous cells are multi-layered grouped)"/"Bland Chromatin (texture of cell nucleus)" and "Normal Nucleoli (visibility of small nucleoli in nucleus)"/"Mitoses (mitosis taking place)", respectively. We also added a variable that represent the sum of "Clump Thickness", "Uniformity of Cell Shape", "Marginal Adhesion", "Single Epithelial Cell Size" and "Bare Nuclei". For the HEP set, we added interactions between "Histology"/"Sgot" and "Albumin"/"Protime" that also aim to capture the interplay between liver function or damage and the proteins found. We also included a third variable that represents the sum of all categorical diagnosis-related variables in the dataset.

These datasets do not contain personal information about the patients, and it is not possible to link the cases to real identities. Nevertheless, we observe imbalanced proportions for sex in the HEP dataset (medical studies should benefit all populations). It is not clear if patients provided an informed consent for the data collection.

3 Results

We used two classifiers - Decision Tree and KNN - to predict the presence of breast cancer or hepatitis. For our experiments, we defined a unique test set (15%) for a final evaluation of accuracy. Cross-validation (5 folds) was applied during training (85%) for evaluation of hyperparameters. Each dataset, with and without our extra features, were evaluated with both classifiers.

Decision tree model: For the cost functions, in addition to Misclassification Cost, Entropy Cost and Gini Index Cost, we evaluated Entropy Weighted Max Cost and the Gini Weighted Max Index (Alkhalid A., 2013). Entropy Weighted Max Cost is based on the Entropy Cost, but instead of considering the sum over entropy on the classes, it considers only the maximum class entropy (the weighted descriptor is related to the multiplication of $log(p(Ci))$ by $p(Ci)$, where $p(Ci)$ is the probability of class Ci and represents the weight). Gini Weighted Max Index has a similar procedure for the probability of error by class. These Max Cost functions would favor early splits whose branches display a majority class. The hyperparameters considered for the experiments were minimum number of instances per leaf $\{1, 2, 5, 10, ..., 50\}$ and maximum tree depth $\{1, 2, 3, ..., 50\}$.

Table 1 presents the results of the mean accuracy rate reported during training for the Decision Tree model. For the BCW set, we obtain the best training accuracy (96.81%) when we include the extra features, the minimum number per leaf is one and the maximum depth is equal to two, regardless of the cost function. When the extra features are not included, the best performance is found when minimum number per leaf is five and maximum depth is six for Gini Index Cost (95.93%). For BCW classification, including the extra features increases accuracy between 0.73% and 2.83% for all cost functions.

In general, during training in the BCW dataset, the best accuracy rates of each cost function are comparable. After the optimal maximum depth is found (four, five or six), increasing this parameter decreases the accuracy of the model in many cases. Increasing the minimum number per leaf seems to slightly decrease the accuracy (although the two hyperparameters may be correlated as increasing minimum number per leaf may imply less depth of the tree). For the HEP set, we obtain the best training accuracy (83.08%) when using Misclassification Cost and the maximum depth equal to one, regardless of whether the extra features were added or not. For HEP classification, using Misclassification Cost increases accuracy between 2.5% and 4% when compared to the other cost functions. Table 2 presents the accuracy rate in the test sets for the best models as per accuracy during training. Test accuracy rate is 94.9% for the BCW and 80% for HEP sets, regardless of the use of extra features. Misclassification cost is the recommended cost function.

As for the KNN model, since the HEP data has both continuous and categorical variables, measure-

	(min. number per leaf, max. depth) Accuracy (%)				
Data	**Misclassif. cost**	**Entropy Cost**	**Gini Index**	**Entropy WMax**	**Gini WMax**
BCW	(1, 4) 95.22	(20, 5) 95.75	(5, 6) 95.93	(5, 5) 95.93	(5, 6) 95.93
BCW +	***96.81**	***96.81**	***96.81**	***96.81**	***96.81**
HEP	****83.08**	**76.92	**78.46	**78.46	**78.46
HEP +	****83.08**	**76.92	**76.92	**78.46	**76.92

Table 1: Decision Trees classification results (cross-validation). *: min. number per leaf is one and max. depth is two. **: min. number per leaf is one and max. depth is one or two. BCW + and HEP +: analysis includes extra features.

Data	**Cost F. - Min. num. leaf - Max. depth**	**Accuracy**
BCW	Gini Index - 5 - 6	94.9
BCW +	Misclass. - 1 - 2	94.9
HEP	Misclass. - 1 - 1	80
HEP +	Misclass. - 1 - 2	80

Table 2: Decision Trees classification results (Test Set). BCW + and HEP +: analysis includes extra features.

ments of accuracy in cross-validation were done using Hamming distance for categorical variables, combined with either Euclidean or Manhattan distance for the continuous variables. When combining different distance metrics, we also investigated the effects of multiplying the total distance of continuous variables by a scalar. This not only allows the different metrics to be re-scaled appropriately, but potentially for features more correlated to the classes to be given more weight. Thus, the hyperparameters were k (number of training samples nearest to a query point) in $K_1 = \{1, 2, 3, ..., 10, 15, 20, 50, 100, 200\}$ and $K_2 = \{1, 2, 3, ..., 10, 15, 20, 50\}$ for the BCW data and the HEP data, respectively, as well as the distance multiplier when appropriate. Table 3 presents the results for cross-validation using the aforementioned k values. We conclude that for the BCW data, using Euclidean distance with the extra features and a k value of 7 is recommended. For the HEP data, using Euclidean distance with the extra features, a k value of 8 and a multiplier of 11 is recommended. As for the impacts of k on the training accuracy, we see from table 5 that it decreases almost monotonically as k increases, for every data set. The test accuracy, on the other hand, tends to have a peak somewhere between $k = 3$ and $k = 15$ depending on the data set (there may be more than one peak). For the BCW data, the test accuracy generally seems to decrease as k moves away from these peak values. For the HEP data, we only observe that the peaks are in the above range and not at the extremes ($k = 1, 2, 20, 50$). It is of course more difficult to observe the relationship between k and the test accuracy of the HEP data, as the test set is much smaller than for the BCW set (15 versus 98). As for the distance functions, we see from tables 3 and 4 that for the BCW data, Euclidean distance performs only slightly better than Manhattan distance, both in cross-validation accuracy and in test accuracy (by a difference of at most 1%). For the HEP data, Manhattan distance generally allowed for slightly better performance in cross-validation, but was surpassed by the average test accuracy using Euclidean distance without extra features. However, using the multipliers, both distance functions allowed for significantly greater cross-validation accuracy and test accuracy, especially when combined with the extra features. Comparing both distance functions with multipliers, we see that Manhattan distance performed better in terms of test accuracy and that in terms of cross-validation accuracy, it was more even. Finally, we should note that for the HEP data, it is difficult to say whether or not adding extra features changes the results meaningfully, without using multipliers. However, when using the multipliers, we see a marked improvement, both in cross-validation and in testing.

Finally, based on the best set of hyperparameters found during training, the KNN model outperforms the decision tree model in both datasets: slightly better accuracy rate for the BCW dataset (95.9% vs 94.9%, respectively) and better accuracy rate for the HEP dataset (86.7% vs 80%).

In figure 3, we can observe how two different levels of the hyperparameter maximum depth could affect the decision boundary produced by our models. Clump Thickness and Bland Chromatin, from the BCW dataset, represent measures from the cell and the nuclei, respectively, and have a wide spread in the data range. For the decision tree, when using a maximum depth of four, we misclassify most of the yellow points (malignant labels) at the center of the domain

	Average accuracy (%), Maximum accuracy (%)			
Data	**Euc**	**Man**	**Euc (mult.)**	**Man (mult.)**
BCW	(96.2) 97.2	(96.0) 96.8	N/A	N/A
BCW +	**(96.4) 97.2**	(96.0) 97.0	N/A	N/A
HEP	(84.9) 90.8	(86.0) 90.8	(88.2) 93.8	(88.9) 93.8
HEP +	(85.3) 89.2	(86.7) 90.8	**(90.0) 95.4**	(88.9) 93.8

Table 3: KNN classification results (cross-validation). The average accuracy is taken from the results using all $k \in K_i$ and the results for the (mult.) columns are produced using the optimal multiplier for that combination of distance function and data set. BCW + and HEP +: analysis includes additional features.

	Average accuracy (%), Maximum accuracy (%)			
Data	**Euc**	**Man**	**Euc (mult.)**	**Man (mult.)**
BCW	**(95.3) 96.9**	(94.7) 95.9	N/A	N/A
BCW +	(95.1) 96.9	(94.8) 96.9	N/A	N/A
HEP	(80.0) 86.7	(79.0) 86.7	(81.0) 86.7	(82.1) 86.7
HEP +	(77.4) 86.7	(79.5) 86.7	(83.1) 93.3	**(83.6) 93.3**

Table 4: KNN test set results. The average accuracy is taken from the results using all $k \in K_i$ and the results for the (mult.) columns are produced using the optimal multiplier identified in cross-validation for that combination of distance function and data set. BCW + and HEP +: analysis includes additional features.

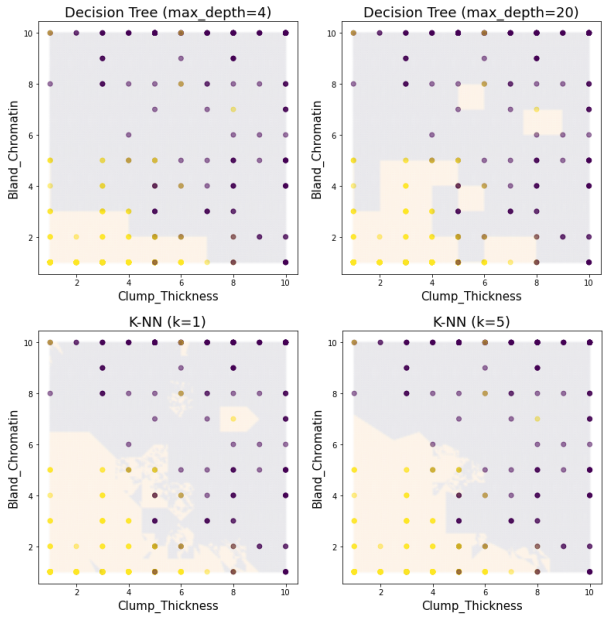

Figure 3: Decision Boundaries: Clump Thickness and Bland Chromatin

region. When we use a maximum depth of 20 instead, we can classify correctly the malign cases in the center of the region. For this particular setting, a deeper tree captures more explanatory power (more and better splits) from these two features. As for the KNN model, when k is one, the decision boundary depends only on the nearest neighboring point, which makes it very sensitive to changes in the training data. For example, when Clump Thickness is eight and Bland Chromatin is seven, a single yellow point (malignant label) would become prominent in a region populated mostly by purple points (benign labels). However, when considering five nearest neighbors, the few yellow points with high Bland Chromatin values lose their impact in favor of the purple points in the surrounding area. This shows how increasing k in a KNN model smooths the decision boundaries and makes the prediction more robust.

4 Discussion and Conclusions

Based on our experiments, for the task of disease classification (breast cancer or hepatitis), KNN produced more accurate predictions than those from the decision trees for both BCW (95.9%) and HEP (86.7%) datasets. We conclude that using distance measures and creating a local approximation (neighbourhood) of the query to the training points provides a better disease prediction on these sets. However, during training, adding extra features capturing interactions increased the performance of decision trees, particularly for the BCW dataset. Exploring and adding different features could potentially overcome the model bias (decision boundaries along the axis) and increase prediction accuracy to a level comparable to that of the KNN model. Aditionally, during data pre-processing, we could apply a missing values imputation step, which would alleviate issues arising from the small size of the HEP dataset. For KNN, with some exceptions, there is not much difference between using Euclidean or Manhat-

Type	K value														
	1	2	3	4	5	6	7	8	9	10	15	20	50	100	200
BCW	100.0	97.4	98.2	97.6	98.2	97.8	97.9	97.0	97.3	96.5	96.9	96.7	96.0	95.5	94.5
BCW +	100.0	97.4	98.0	97.6	98.2	97.5	98.0	97.5	97.5	97.1	96.9	96.7	96.1	95.6	94.3
HEP	100.0	90.0	88.5	87.7	90.8	90.0	92.3	90.0	89.2	89.2	86.9	84.6	84.6	N/A	N/A
HEP +	100.0	89.2	90.0	89.2	91.5	90.0	90.8	90.0	90.0	89.2	86.9	84.6	84.6	N/A	N/A

Table 5: KNN Training Accuracy (average over Euclidean and Manhattan distance), BCW + and HEP +: analysis includes additional features.

Type	K value														
	1	2	3	4	5	6	7	8	9	10	15	20	50	100	200
BCW	95.9	93.9	95.9	95.4	96.4	95.9	96.4	95.4	95.9	95.4	96.4	95.4	93.4	92.9	90.3
BCW +	95.4	92.3	95.9	94.9	96.9	94.9	95.4	94.9	95.9	95.9	96.9	95.9	94.4	93.4	91.3
HEP	80.0	73.3	80.0	83.3	83.3	83.3	83.3	80.0	73.3	73.3	80.0	80.0	80.0	N/A	N/A
HEP +	76.7	76.7	73.3	80.0	83.3	80.0	73.3	76.7	76.7	80.0	83.3	80.0	80.0	N/A	N/A

Table 6: KNN Test Accuracy (average over Euclidean and Manhattan distance), BCW + and HEP +: analysis includes additional features.

tan distance. The major contributing factor was to add a multiplier to the total distance for one metric when using two different metrics, so that the metrics can re-scale themselves appropriately with respect to each other, with the bonus that variables more correlated with the classes become more important. To further improve this effect, given n features, there could be a multiplier applied to $n-1$ features and this vector of multipliers could be a hyperparameter. This could significantly increase the pre-processing time, but dimensionality reduction could also be applied to bring the dataset to a lower dimensional space. For the decision tree models, our modified cost functions, which consider only the maximum entropy (or error probability) per class in the node sample, favor splits with clear majority early in the greedy search. We observed that, most of the time, the results were comparable or slightly lower to those from the traditional Entropy and Gini index cost functions for these datasets. In any case, as per our experiments, Misclassification cost would be the recommended option for both datasets. During training for the decision tree model, the best performing models were those with low depth values, even when there are between 10 and 19 features for analysis. This could be explained by the characteristics of the data. As shown in Figure 1, the distribution of the classes in the BCW set is quite skewed. Most instances of the benign label are concentrated at very low 'uniformity of cell size' and 'marginal adhesion' values, while instances of the malign label spread over higher values of these features. Thus, the models were able to categorize input data into the correct class using only a few tests and provide high accuracy, without the need to scrutinize every feature available. However, this is less effective on the HEP data, where the classes overlap to a non-trivial degree at certain feature values, as we may see in Figure 2. This also contributes to the difference of performance observed in the two datasets. An interesting line of research for decision trees would be the modification of the greedy search method for splitting. Currently, the greedy search looks only for the results of the cost function for the next step, but we could implement a greedy search for two steps ahead. Although this could help overcoming the miopic nature of the greedy search, this modification can be costly from a computational point of view. Another direction of investigation for the decision tree model is to study the depth of trees being constructed by the algorithm and to examine the effects of pruning.

5 Statement of contributions

Miguel Ibanez Salinas:
Code: Model and Experiment design. **Write up**: Introduction, Data sets, Results, Tables, Discussion and Conclusion, and Editing.
Alex Le Blanc:
Code: Model and Experiment design. **Write up**: Abstract, Results, Tables, Discussion and Conclusion, and Editing.
Yang Chen:
Code: Experiment. **Write up**: Introduction, Figures, Results, Discussion, Conclusion and Editing.

References

Moshkov M. Alkhalid A., Chikalov I. 2013. *Comparison of Greedy Algorithms for Decision Tree Optimization*. Springer.

Dheeru Dua and Casey Graff. 2017. UCI machine learning repository.

Xiaoli Fern and Carla Brodley. 2003. Boosting lazy decision trees. 1.

Salvador Garcia, Joaquín Derrac, José Cano, and Francisco Herrera. 2012. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34:417–435.

Ron Kohavi. 1997. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. *KDD*.