COMP551 A1

Yusen Tang, Keyu Yao, Jaylene Zhang

September 2022

Abstract

The assignment task is to investigate the performance of two machine learning models on the same two datasets. We found that both K-Nearest Neighbor and the Decision Tree behave better when there are fewer data points. Our team observed different accuracy by adjusting different amount of features we used, the K on K-Nearest Neighbor, the depth on the Decision Tree, and also the splitting between training data and testing data on both models. Generally, decision tree gives higher accuracy than KNN in most situations.

Introduction

In this assignment, we implemented two machine learning models, K-nearest Neighbors and Decision Tree on the same two datasets hepatitis data and messidor data.

The data talks about the statistic data of some hepatitis symptoms and the life or death result, which means that We need to assess the impact of these characteristics on life and death by using a part of data, then predict the results by using another part of the data and see the accuracy of two models.

Methods

The KNN algorithm stores the features and classifications of all training data points, then predicts new data point based on the calculated similarity between new point and the existing points. We take the K nearest neighbors of our predicting point by calculating the distance between the predicting point and every pair of data points, apply classification by choosing the category for which the number of neighbors is maximum. We calculate the Pairwise distance by using:

$$distance(x^i, x^j) = \|\mathbf{x^i} - \mathbf{x^j}\|_2 = \sqrt{\sum_{d=1}^{D}(x_d^i - x_d^j)^2} \tag{1}$$

We calculate the class probabilities of each class by using:

$$p(y^{(*)} = c \mid x^{(*)}) = \frac{1}{k} \sum_{n \in N_K(x^{(*)}, D)} \mathbb{1}(y^n = c) \tag{2}$$

Decision tree is a classifier, where internal nodes represent the features of a dataset, branches represent the splitting thresholds and each leaf node represents the category. Starting from the root, the algorithm compares each decision rule and picks the one with the highest accuracy, and build the tree recursively until we can no longer split the data points.

Datasets

In this experiment, we processd and analyzed two datasets.To process dataset1, hepatitis.csv, we import the data, dropping all the rows and conlumns with "?". As a result, we are left with 80 rows consisting of 19 features and 1 class representing patients' live or die of hepatitis. The valid data contains the description of 80 patients, including 69 males and 11 females from 20 to 72 years old. Dataset 2 contains features extracted from the Messidor image set to

predict whether an image contains signs of diabetic retinopathy or not. It contains 19 features and one label class.

By using the correlation function, we choose the top 4 features which are most related to the Class result. Given that there are more binary features in the 2 samples, we will avoid the situation that all features used are binary to reduce its negative effect on KNN. Our initial input features are ASCITES,ALBUMIN,PROTIME,VARICES for df1, and MA_alpha=0.5,MA_alpha=0.6,MA14,QUALITY ASSESSMENT for df2.

to fit the model and make the prediction. The distribution of the features, ASCITES, with the highest correlation value is shown below:

```
Class              1.000000
ASCITES            0.479211
ALBUMIN            0.477404
PROTIME            0.395386
VARICES            0.345785
SPIDERS            0.287839
MALATSE            0.275595
FATIGUE            0.181151
SEX                0.175876
SPLEEN PALPABLE    0.135643
STEROID            0.123830
SGOT               0.078731
LIVER FRIM         0.055978
ANTIVIRAILS       -0.108776
ANOREXIA          -0.185042
ALK PHOSPHATE     -0.189360
LIVER BIG         -0.194030
AGE               -0.212769
BILIRUBIN         -0.351557
HISTOLOGY         -0.456856
Name: Class, dtype: float64
```

```
CLASS LABEL                  1.000000
MA_alpha=0.5                 0.292206
MA_alpha=0.6                 0.265941
MA_alpha=0.7                 0.234307
MA_alpha=0.8                 0.197059
MA14                         0.184677
MA15                         0.177231
MA_alpha=0.9                 0.161222
MA13                         0.151322
MA12                         0.142144
MA_alpha=1.0                 0.127498
MA11                         0.104058
QUALITY ASSESSMENT           0.062902
MA8                          0.057820
MA10                         0.038027
EUCLIDEAN DISTANCE           0.007278
MA9                          0.000252
OPTIC DISCDIAMETER          -0.031320
AM/FM-BASED CLASSIFICATION  -0.040900
PRE-SCREENING               -0.076690
Name: CLASS LABEL, dtype: float64
```

We observe that the correlation/covariance between above features and class is high. Furthermore, use df.describe() function we obtain basic statistics for different features.

Results

0.1 KNN

Starting off with 4 features we will fit the model and make the prediction. We first try to find the optimum k value for each data-set.

k value	df1	df2
1	0.90	0.60
2	0.67	0.62
3	0.89	0.66
4	0.86	0.64
5	0.86	0.67
6	0.93	0.64
7	0.90	0.63
8	0.87	0.61
9	0.87	0.61
10	0.87	0.61

The table above presents the changing accuracy with k values ranging from 1 to 10. The accuracy reaches the peak at k = 6 for df1 and k = 5 for df2, although several k values give the same accuracy. We also observe that the accuracy first increases but after it hits the maximum until a certain point.

We also attempt to investigate the hyper-parameter in different variations of KNN. However, it is interesting to notice that there is actually a dramatic fluctuation in the optimum k value, even the permutation step could have influence on it. Hence, we decide to stick to one k value and apply it to all our later experiments.

Method		df1	df2
standard knn	2 features	0.85	0.61
	4 features	0.90	0.63
	8 features	0.89	0.63
	16 features	0.75	0.58
using standardized features		0.80	0.44
metric = Manhanttan dist		0.80	0.66

By running experiments on two data-sets(shown by table above), it can be noticed that standard KNN gives a much better result on df1 than df2, regardless of how we modify the algorithm. The best result is generated from the standard KNN with 4 features being used, which exactly matches our initial choice of input data. Also it surprises us that using too much features does not always give an ideal result,

proving that KNN may give poor result when using high dimension data.

In order to narrow the scale of input variables we apply the standardization over the features, leading to rather opposite results on df1 and df2 as the accuracy of df2 suddenly drops for a quite bit. In addition, we replace the default metric with Manhattan distance, which is calculated as the sum of the absolute differences between the two vectors. The new metric indeed improves the result of df2 by a small amount on average but for df1 no significant enhancement is observed.

To further improve the accuracy on the second data set, we tried increasing the training size and it did have a bit of positive affect. Furthermore, we implement the weighted KNN which finally raises the result to some extent. Among multiple experiments most of the weighted KNN accuracy exceed 0.65, with the highest reaching 0.70. (see table on the next page)

k	1	2	3	4	5	6	7	8	9	10
df2 weight KNN	0.68	0.68	0.68	0.69	0.69	0.68	0.68	0.7	0.7	0.7

prediction result using weighted knn on df2

0.2 Decision Tree

The experiment result of df1 is shown below:

Depth	Training	Testing	Cost Function	Accuracy
1	70	10	gini_index	81.4
2	70	10	gini_index	92.9
10	70	10	gini_index	97.1
20	70	10	gini_index	97.1
1	70	10	entropy	85.7
1	40	40	entropy	82.5
1	40	40	misclassification	82.5
10	40	40	misclassification	72.5
10	70	10	misclassification	98.6

prediction result using top 4 correlated features

Depth	Training	Testing	Cost Function	Accuracy
1	70	10	entropy	90
5	70	10	entropy	90
20	70	10	entropy	90
20	70	10	misclassificaion	91.7
3	70	10	misclassification	91.7
1	70	10	misclassification	91.7
1	60	20	misclassification	91.7
20	60	20	misclassification	91.7
20	40	40	misclassification	91.7
20	40	40	gini_index	93.3
1	40	40	gini_index	93.3
2	40	40	gini_index	95

prediction result using all correlated features

We observe that by solely changing the depth of decision tree, we find that the accuracy data stays the same. For instance, from depth=1 to depth=20, the accuracy of implementing the same cost function are always same. Even we change the proportion of training and testing data set, the accuracy would not have a significant change(or no change at all). The accuracy actually only changes with the cost function. So we have two speculations1.we made mistakes in implementing the model or 2.those four features maybe highly related to the result that could make the model reach a relatively balanced state. Therefore, we make a contrast test, which uses all features to predict the result.

Therefore in the contrast test, we use all the features to predict the result. By Changing the depth only, We could see that the by using a same training or testing dataset and cost function,we could get the different accuracy. But the range of accuracy could be much more bigger than choosing 4 features only.

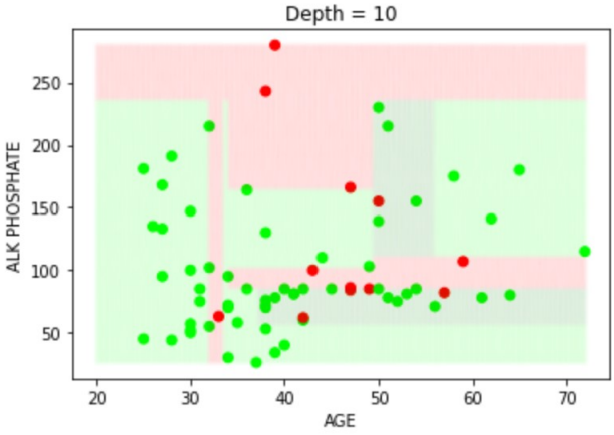
Depth	Training	Testing	Cost Function	Accuracy
20	575	575	misclassification	64.3
1	575	575	misclassification	56.9
7	575	575	misclassification	61.2
1	800	350	misclassification	61.4
7	800	350	misclassification	64.1
7	800	350	entropy	65.9
7	575	575	entropy	60.9
20	575	575	entropy	64.3
1	575	575	entropy	51.1
1	575	575	gini_index	56.5
7	575	575	gini_index	61.7
20	575	575	gini_index	63.5
20	800	350	gini_index	78.4

Implementing the decision tree in the second dataset create lower accuracy with big range, from 56.9 to 78.4. But we could observe a phenomenon that with same cost function and training/testing

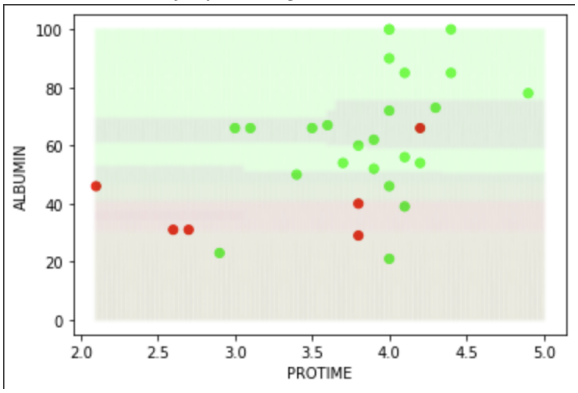
data set, the accuracy is generally increase with the depth. With same depth, the accuracy could be higher with more training samples. Or maybe the testing samples are not enough to lower the accuracy. The influence of the cost function is not obvious.

0.3 Comparing KNN and Decision Tree Prediction Results

The decision boundary is to help us understand how a classification algorithm divides up in the feature space. We can see that different color's areas will explain the new result of the new data set. Decision Tree: As shown below, we have drawn the decision boundary for df1 using DT and KNN respectively.



decision boundary df1 using DT



decision boundary df2 using KNN

As illustrated by the two figures and previous table, DT tends to yield a better prediction than KNN on average as it can be noticed that DT's accuracy exceeds 0.90 more frequently, especially with all features used for training the model. It is interesting to see that neither KNN nor DT appears to be an ideal model for the second data-set. Most of the results are among 0.60, with the highest being only 0.78 and lowest 0.58. And the prediction from the 2 models does not differ by a lot.

Discussion and Conclusion

Generally speaking, in the experiment of df1 and df2, we found that the accuracy for predicting df2 is significantly lower than the one for df1. For this observation we conclude that both the machine learning models behave better with small data set. Also we made some hypothesis based on that different selection of features which creates such various results. In our KNN experiment, we can see that the accuracy grows with the number of features we use, but drops if we use too many features. We figured out that finding the appropriate number of features is the key factor for KNN model, since we have to use the right amount of information to classify but not too many such that noise features may cause interference with our prediction result. Another finding involves with the K number. When we increase K by too much, we are technically including the entire graph as a point. We think that too many neighbors may cause the over-fitting problem, which decreases the accuracy. In addition, we realize that KNN could be very expensive when the sample size is large since it requires traversing all data points. For decision tree, the accuracy does improve with the increasing depth, but our assumption at first was that too many depth would cause the over-fitting problem, since the tree structure fits perfectly with the training data set, it is possible to fail to predict future testing data set.

For future investigation, we will consider implementing multiple linear regression on the two samples.

Statement of Contributions

Keyu Yao cleaned the dataset. Yusen Tang and Jaylene Zhang were responsible for decision tree and KNN implementation respectively(including all variations of the algorithm and cost functions). Each of them summarized their result from experiment under the "Result" section of the report. Keyu Yao finished the remaining part of the report.