Group Assignment 1 Report

Comp 551 Applied Machine Learning
Professor Yue Li

Yujie Gan (260914350)
Kevin Liu (260916466)
Zhixuan Ren (260895556)

Oct. 6th, 2022

Abstract:

In the assignment, we implement two machine learning methods, K-Nearest Neighbors (KNN) and Decision Tree (DT) on two datasets to investigate and compare the classification performance. We preprocess the dataset and implement these two methods to train the model. After the investigation, we find that the Decision Tree model generates slightly better results, but the time cost of the DT is much larger than KNN. Also, the weighted KNN model would help to reduce the effect on the model accuracy from the extreme values in the training dataset.

Introduction:

In this assignment, we are assigned the task to implement two methods, KNN and DT, on the dataset to train the model and predict the class of the target variable.

We are given two datasets to train the model:Hepatitis dataset and Messidor dataset. The first dataset contains the patient health information and its target variable is survivability; the second dataset contains the features from Messidor images and the target variable is whether the image contains signs of diabetic retinopathy or not.

After implementing the method, training the models by trying different parameters on both datasets, making comparisons between the models and visualizing the outcomes, our key findings include mainly two points: even though the difference is not significant, generally DT generates better results with longer operation time; weighted KNN could help increase the accuracy of the model.

Methods:

We implement two methods: KNN and DT. The algorithms of these two methods are described as follows.

In the KNN method, we decide the class of a point by checking the class distribution of its K nearest points, in other words, the k points with the smallest distance to it. The distance between two points is calculated by different distance formulas, and what we applied in the assignment are the Euclidean formula and the Manhattan formula.

For the DT method, we split and cluster the data by setting a series of nodes on the features. We decide the parameter on each node when it minimizes the information loss, and the information loss is decided by the cost function. The cost functions we applied in the assignment are Entropy and Gini Index. After we construct the tree from the training dataset, we input a new point’s dependent variables into the model and thus decide the class of the target variable.

Datasets:

In the assignment, we are given two datasets: Hepatitis and Messidor.

The Hepatitis dataset includes 155 instances and 19 attributes in total. The first attribute, named “Class”, is the target variable and contains two classes 1 and 2 to represent the survivability (die or live) of the patients. The rest 18 columns contain attributes of patients' information to present their health conditions.

For the data process, first we rename the attributes.The second step is to clean the instances of missing values which could affect the prediction model. We once considered directly removing all the instances which contain missing values in any columns, but it would be a large data loss since about half of the instances are not in perfect condition. Therefore we set a threshold of 20 and count the missing value of each column. If the column contains more missing value than the threshold, we

remove the column from the training dataset. After then we remove the instances with missing values, and the dataset after processing contains 128 instances and 18 columns. Thirdly, the dataset also contains several columns with numerical values but object type. For further model training, we transform the type of these inputs to float and int types. Lastly, to better implement the KNN algorithm on the dataset, we change the class values 1 and 2 into 0 and 1when splitting the training, testing and validation data.

Among the 128 instances, we have 104 instances with the class value 1 and 24 instances with class value 0. So we conclude that the class values are relatively unbalanced distributed.

The second dataset is Messidor. It contains a total of 1151 instances and 20 attributes. The target variable is the binary result of presence of signs of diabetic retinopathy with 0=bad quality and 1=sufficient quality. The rest features contain information about the images.

Again, we first rename the dataset labels and thus make the jupyter notebook report more readable. However, compared to the Hepatitis dataset, the Messidor dataset contains no missing values. So we just keep all the instances and columns. Also, all the dependent variables of Messidor are float type, so we only change the sign (target variable) to the int type for further analysis. After dropping the duplicates, we have 1146 instances.

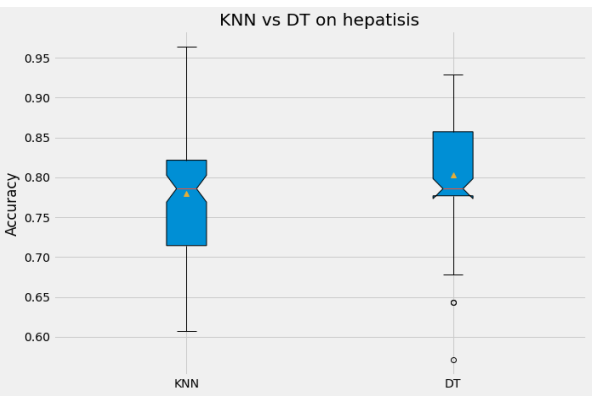
The distribution of the target variable is extremely unbalanced. Only 4 of 1146 instances have negative sign (value 0) and the rest are positive.

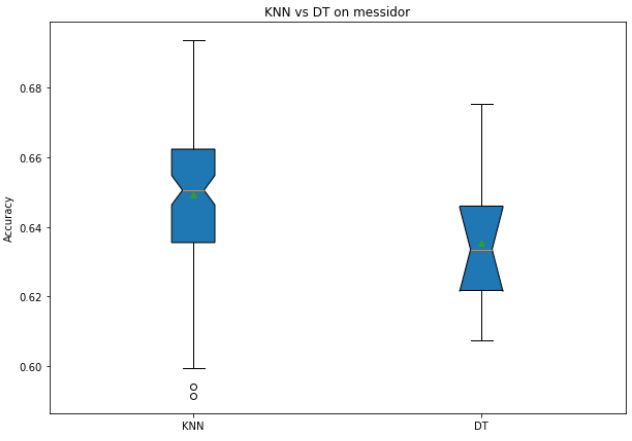
Results:

After implementing the KNN and DT class, we split the datasets and train the model. The best hyperparameter of model, accuracy rate of **one trial** are shown as the table below(**Task 3.1**):

Dataset	Decision Tree	K-Nearest Neighbors
Hepatitis	max_depth=1, 0.785714	k=3, 0.678571
Messidor	max_depth=5, 0.654450	k=11, 0.680628

From above we could see that these two methods produce similar results on two datasets and we need more trials to decide the performance of these two methods. Therefore we made 100 more trials of both methods on both dataset and generated a boxplot to compare the performance (evaluate by accuracy rate). The result are shown as below:

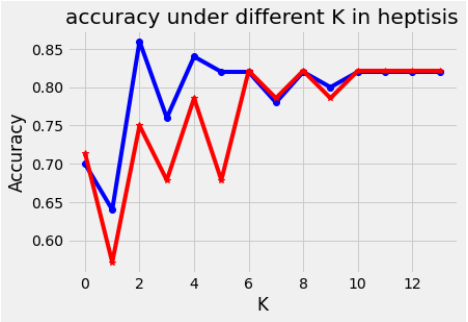


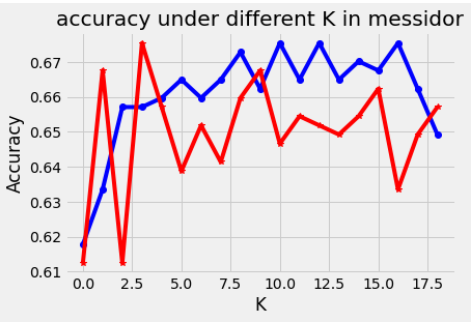


From the boxplot we can see that the DT works slightly better than KNN on the hepatitis data set. The DT has a slightly higher median value and a higher mean value.But the DT works poorly on the messidor data set. Although it doesn't have many outliers, it has lower mean and median value. But from both plots we can see that DT has shorter whiskers, which means that the difference of accuracy between each trial is smaller in DT.

However, because of the hardware constraint, the size of results is still small, the conclusion here is not really convincing. We need to do more trails.

Then we test different k values of the KNN method on two datasets our result are shown as below(**Task 3.2**):



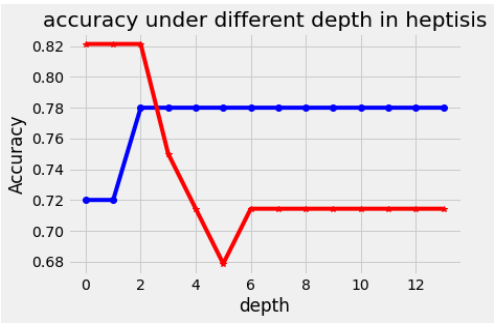


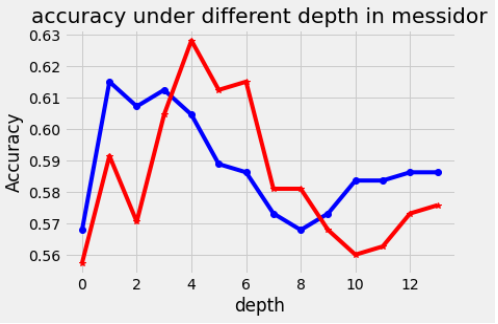
We can observe from the plots that for training accuracy, in both datasets, the accuracy shows a general increasing trend with the increase of k, with slight drop after k exceeding certain values.

However, for the testing accuracy, the trends are different between plots. In hepatitis it moves similarly in the same direction with the training accuracy, but in the Messidor dataset the testing accuracy has maximum values in k=3.

The difference between the train accuracy and the test accuracy is relatively small in trials on both dataset, with maximum difference less than 0.1.

Next we move to check how the max_depth parameter affects the performance of DT on both dataset. The accuracy curve we obtained after trials are shown as pictures below(**Task 3.3**):





The training accuracy rate does not show an obvious pattern on these two datasets. In the Hepatitis dataset, after reaching the maximum accuracy on maximum depth=2, it stays the same until the maximum depth increases. However, in Messidor, it shows that after reaching max accuracy in maximum depth=1, it starts to decrease.

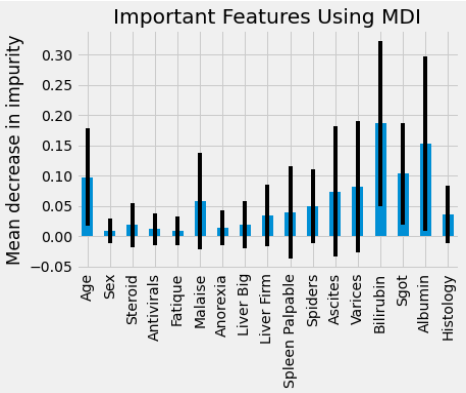
For the testing accuracy, it is also not showing an obvious pattern, but generally the trend is it reaches the maximum accuracy in a relatively small maximum depth and starts to decrease with parameter of maximum depth increases.

In the KNN method, we applied the Euclidean and Manhattan distance function separately. Similarly, we applied Entropy and Gini Index to make comparisons in the DT method. The accuracy result are shown as table below (**Task3.4**):

Dataset	Euclidean	Manhattan	Entropy	Gini Index
Hepatitis	0.8214285714	0.8928571429	0.8571428571	0.8214285714

From the result we could see that the different distance/cost function does lead to difference in final test accuracy rate, but it is relatively small.

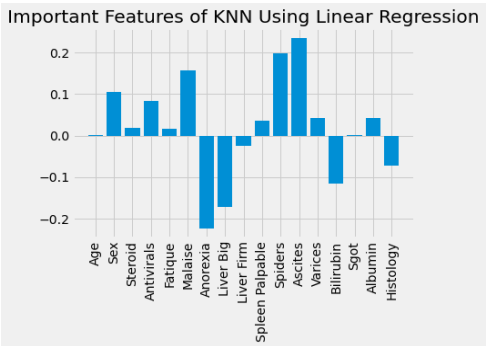
Then we try to seek important features in the DT model on Hepatitis datasets. Here are the results (**Task 3.6**):



We use the RandomForestClassifier in the Sklearn package to obtain the coefficients of importance among all the features. We train this model by using the training dataset in the former steps and then we use importance features to get the array of feature importance from the model. Then we plot them to select the top important attributes.

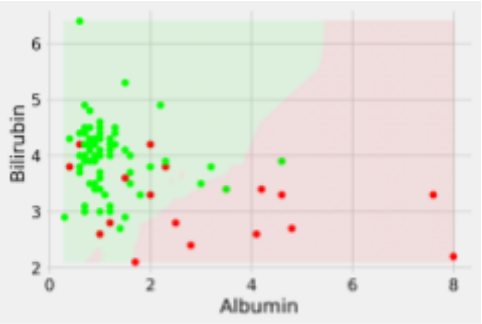
The most significant key features in Hepatitis are Bilirubin, Albumin and Age.

Next we try to seek important features in the KNN model. Again, we make trials on the same dataset and result are as below (**Task 3.7**):

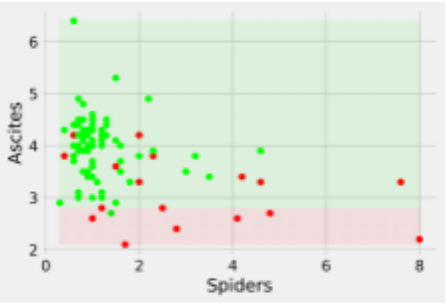


We use the LinearRegression model in the sklearn package to obtain the important features. However, compared the important features obtained by RandomForestClassifier, this time the result is different: Ascites, Spiders, and Malaise are the top three most significant features.

Then we plot the decision boundaries using the top 2 important features of the dataset of the two methods. The results are shown as graphs below (**Task 3.5**).



The decision boundary of KNN (Hepatitis)

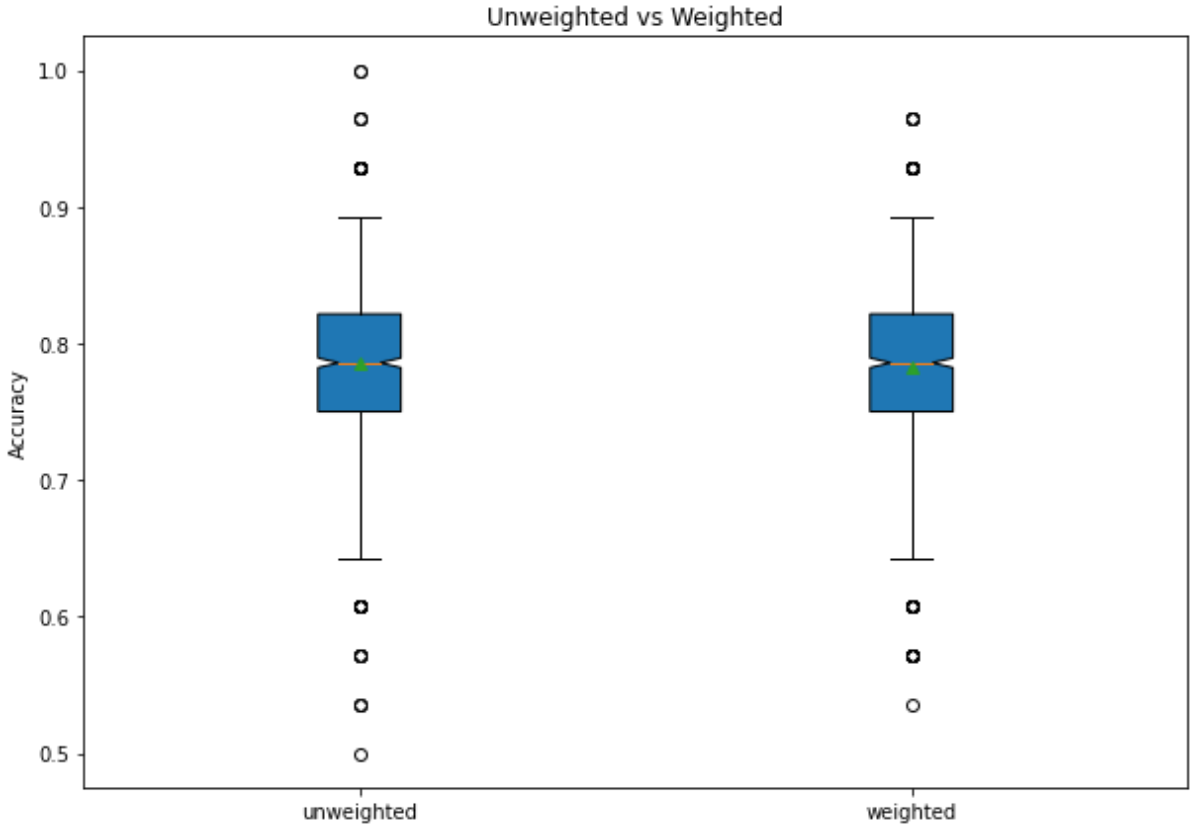


Decision Boundary of DT(Hepatitis)

Discussion and Conclusion:

From the trials on 2 models, we find that the run time of DT is much longer than the rum time of KNN. This is because that DT algorithm takes much time in recursive greedy splitting of nodes.

Also we implement the weighted KNN which performs better than unweighted KNN. The weighted KNN seems to have no improvement on average accuracy compared with unweighted one. However, the outliers of weighted KNN fall higher than the outliers of weighted KNN. This is because the weighted KNN gives more weight to the nearer neighbors and less weight to the farther neighbors in the set of K nearest neighbors. The idea is to be able to be more robust against variations in distances of the k-nearest neighbors which may lead to wrong decisions.



We also find that randomness exists in model training and it has a serious impact on the result of hyperparameter choosing and the test accuracy. In order to

lower the effect of random training data choosing, we should further implement K-folds cross validation to use all of the data as training data.

In conclusion, the key takeaways of our research are: 1. Generally, the DT generates more stable accuracy than KNN in a large number of trails but with the sacrifice of large time costs. 2. Weighted KNN has better performance than unweighted KNN models.

Both dataset given this time are with a large number of attributes. In the future, we would like to investigate how the number of features affect the selection, training and performance of models.

Statement of Contribution:
Yujie contributed more on Task 3.5 (Decision Boundary), Kevin contributed more on Task 3.2 and 3.7 (How hyperparameters affect accuracy and Weighted KNN), and Zhixuan contributed more on the report. We finished all other tasks together.

References:

Feature Importances, https://machinelearningmastery.com/calculate-feature-importance-with-python/, accessed on Oct 4th, 2022

Plot Forest Importances, https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html, accessed on Oct 4th, 2022