$Email$: $LE000288@umn.edu$

ID : 5674954

Question 1

a)

Gini: 1- $\sum_{i=0}^{c-1}(p_i(t))^2 = 1 - 2 * 0.5^2 = 0.5$

Misclassification Error: $1 - max(p_i(t)) = 1 - 0.5 = 0.5$

Entropy: - $\sum_{i=0}^{c-1}(p_i(t))log_2 p_i(t) = -0.5 * log_2(0.5) - 0.5 * log_2(0.5) = 1$

b)

Customer ID:

$Gini_{split}$: $\sum_{i=1}^{k} \frac{n_i}{n} GINI(i) = 0$

Misclassification Error: $\sum_{i=1}^{k} \frac{n_i}{n} Misclassification Error(i) = 0$

Entropy: $\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) = 0$

Shops Weekly:

$Gini_{split}$: $\sum_{i=1}^{k} \frac{n_i}{n} GINI(i) = 0.4167$

Misclassification Error: $\sum_{i=1}^{k} \frac{n_i}{n} Misclassification Error(i) = 0.3$

Entropy: $\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) = 0.875$

Pet:

$Gini_{split}$: $\sum_{i=1}^{k} \frac{n_i}{n} GINI(i) = 0.3125$

Misclassification Error: $\sum_{i=1}^{k} \frac{n_i}{n} Misclassification Error(i) = 0.2$

Entropy: $\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) = 0.7042$

c)

$Gain_{split} = Entropy(p) - \sum_{i=1}^{k} \frac{n_i}{n} Entropy(i)$

Gain for Customer ID:

1 - 0 = 1

Gain for Shops Weekly:

1 - 0.875 = 0.125

Gain for Pet:

1 - 0.7042 = 0.2958

$\therefore$ Customer ID has the highest Information Gain.

d)

$GainRatio = \frac{Gain_{split}}{SplitInfo}$ $SplitInfo = -\sum_{i=1}^{k} \frac{n_i}{n} log_2 \frac{n_i}{n}$

Gain Ratio for Customer ID:

Split Info = 4.32

1/4.32 = 0.23

Gain Ratio for Shops Weekly:

Split Info = 0.97

0.125/0.97 = 0.1289

Gain Ratio for Pet:

Split Info = 1.5219

0.2958/1.5219 = 0.194

$\therefore$ Customer ID has the highest Gain Ratio.

e)

No. Customer ID has the highest Gain Ratio and the highest Information Gain. By adding the Split Info in the denominator, we can see that the difference between the Gain Ratio of Customer ID, Shops Weekly and the Pet is much lower than the difference between the Gain of Customer ID, Shops Weekly and the Pet. The Split Info is used to adjust Information Gain by the entropy of the partitioning that the higher entropy partitioning is penalized. So the Customer ID has high entropy partitioning, which might not be very suitable to put it at the root node. In contrast, the Pet attribute the second highest gain and the moderate Split Info. The pet attribute should be splitting at the root node.

Question 2

a)

training error = (2+4+3+4+7+9+5)/100 = 0.34

b)

$err_{gen}(T) = err(T) + \Omega * (\frac{k}{N_{train}})$

= 0.34 + 0.5 * 7/100 = 0.375

c)

$err_{gen}(T) = err(T) + \Omega * (\frac{k}{N_{train}})$

= (2+4+10+7+15)/100 + 0.5 * 5/100 = 0.405

The generalization error of the pruned tree increased, so we should not prune the tree.

d)

For the original tree, the validation error rate is: 0.4. For the pruned tree, the validation error rate on the validation set after pruning is: 0.3. Since the validation error rate decreases after pruning the tree, the nodes E and F should be pruned.

Question 3

(a)

There are 15 data points in total.

$P(run = Yes) = \frac{11}{15}$

$P(run = No) = \frac{4}{15}$

$\because \frac{11}{15} > \frac{4}{15}$

$\therefore$ The person is more likely to go for a run.

b)

$P(run = yes|Rainy, Mild, High) = (P(Rainy, Mild, High|run = yes) * P(run = yes))/P(Rainy, Mild, High)$

For Naive Bayes Classifier, assume independence among attributes when class is given:

$= (P(Rainy|run = yes) * P(Mild|run = yes) * P(High|run = yes) * P(run = yes))/P(Rainy, Mild, High)$

$= (\frac{6}{11}) * (\frac{5}{11}) * (\frac{4}{11}) * (\frac{11}{15})/P(Rainy, Mild, High) = 0.066/P(Rainy, Mild, High)$

$P(run = No|Rainy, Mild, High) = \frac{(P(Rainy, Mild, High|run=No) * P(run=No))}{P(Rainy, Mild, High)}$

For Naive Bayes Classifier, assume independence among attributes when class is given:

$= (P(Rainy|run = No) * P(Mild|run = No) * P(High|run = No) * P(run = No))/P(Rainy, Mild, High)$

$= (\frac{1}{4}) * (\frac{1}{4}) * (\frac{1}{2}) * (\frac{4}{15})/P(Rainy, Mild, High) = 0.00833/P(Rainy, Mild, High)$

$\because P(run = yes|Rainy, Mild, High) = 0.066/P(Rainy, Mild, High) > P(run = No|Rainy, Mild, High) = 0.00833/P(Rainy, Mild, High)$

$\therefore$ The unseen data point X would be classified as Run = Yes.

c)

Let i be the event that $\{temperature = hot\}$, go be the event that $\{run = Yes\}$ and stay be the event that $\{run = No\}$

$P(go|i) = (P(i|go) * P(go))/P(i) = \frac{3}{11} * \frac{11}{15}/\frac{5}{15} = 0.6$

$P(stay|i) = (P(i|stay) * P(stay))/P(i) = \frac{1}{2} * \frac{4}{15}/\frac{5}{15} = 0.4$

$\because P(go|i) = 0.6 > P(stay|i) = 0.4$

$\therefore$ The person will more likely run given the temperature is hot by NB's prediction.

d)

Let i1 be the event that $\{temperature = hot\}$, i2 be the event that $\{humidity = normal\}$,go be the event that $\{run = Yes\}$ and stay be the event that $\{run = No\}$

$P(go|i1, i2) = \frac{(P(i1,i2|go)*P(go))}{P(i1,i2)}$

For Naive Bayes Classifier, assume independence among attributes when class is given:

$= \frac{(P(i1|go)*P(i2|go)*P(go))}{P(i1,i2)}$

$= \frac{(\frac{3}{11}*\frac{7}{11}*\frac{11}{15})}{P(i1,i2)}$

$= \frac{0.127}{P(i1,i2)}$

$P(stay|i1, i2) = \frac{(P(i1,i2|stay)*P(stay))}{P(i1,i2)}$

For Naive Bayes Classifier, assume independence among attributes when class is given:

$= \frac{(P(i1|stay)*P(i2|stay)*P(stay))}{P(i1,i2)}$

$= \frac{(\frac{1}{2}*\frac{1}{2}*\frac{4}{15})}{P(i1,i2)}$

$= \frac{0.06667}{P(i1,i2)}$

$\because P(go|i1,i2) = \frac{0.127}{P(i1,i2)} > P(stay|i1,i2) = \frac{0.06667}{P(i1,i2)}$

$\therefore$ The person will more likely run given the temperature is hot and the humidity is normal by NB's prediction.

e)

Naive Bayes handles the missing data pretty well. It obeys the advantages of Naive Bayes Classifiers in the lecture – 'Handle missing values by ignoring the instance during probability estimate calculations'

f)
Let i1 be the event that $\{temperature = hot\}$, i2 be the event that $\{humidity = normal\}$,i3 be the event that $\{outlook = tornado\}$, go be the event that $\{run = Yes\}$ and stay be the event that $\{run = No\}$

$P(go|i1,i2,i3) = \frac{(P(i1,i2,i3|go)*P(go))}{P(i1,i2,i3)}$

For Naive Bayes Classifier, assume independence among attributes when class is given:

$= \frac{(P(i1|go)*P(i2|go)*P(i3|go)*P(go))}{P(i1,i2,i3)}$

$= \frac{(\frac{3}{11}*\frac{7}{11}*0*\frac{11}{15})}{P(i1,i2,i3)}$

$=0$

$P(stay|i1,i2,i3) = \frac{(P(i1,i2,i3|stay)*P(stay))}{P(i1,i2,i3)}$

For Naive Bayes Classifier, assume independence among attributes when class is given:

$= \frac{(P(i1|stay)*P(i2|stay)*P(i3|stay)*P(stay))}{P(i1,i2)}$

$= \frac{(\frac{1}{2}*\frac{1}{2}*\frac{1}{2}*\frac{4}{15})}{P(i1,i2,i3)}$

$= \frac{0.0333}{P(i1,i2,i3)}$

Although $P(go|i1, i2, i3) = 0 < P(stay|i1, i2, i3) = \frac{0.0333}{P(i1,i2,i3)}$, $P(i3|stay)$ is 0. Since we have zero conditional probability, the results are indeterminate.

g)

The probability of outlook = tornado given run = yes is zero. This zero conditional probability leads to the overall naive bayes prediction zero. for this class zero. We could use Laplace estimate or m-estimate to avoid this problem.

Question 4

a)

I would choose N_1 classifier. Classification models are generated from the training data, so in most situation, it would fit the training data well. But indeed we need to check the accuracy of the model for the testing data to know the true model accuracy.

In this problem, A1 is the training data and A2 is the testing data. For the situations that N_2 and N_3 classifiers have higher training accuracy for training data than that of the N_1 classifier and lower testing accuracy for testing data than that of the N_1 classifier, these may show the presence of overfitting in N_2 and N_3 classifiers. The N_1 classifier shows similar accuracies for both training data and testing data, which means the N_1 classifier learned from the training data could predict well over testing data.

b)

No. Generally, a more complex model has a better ability to represent complex patterns in the data. However, an overly complex model also has a tendency to learn specific patterns in the training set, which will cause overfitting and the specific patterns would not generalize well over unseen instances. This highly complex deep neural networks has 18% higher accuracy for the training data compare to that of the testing data, which means this high complex model has the overfitting problem. Thus, I would not submit this classifier instead of the earlier three.

c)

I would choose "Neither A1 nor A2". To measure the performance of the model, we need to use the unseen testing data to show the generalization performance of the model. Since A1 and A2 are given, we could treat them as training set and validation set. The participants can adjust the model on these two data sets to get the high accuracy, although they do not know the exact labels in A2. Then the accuracy of a model on the A1 and A2 cannot represent the generalization performance of the model. So we need to use another data set A3 to test the generalization performance of the model

Question 5

a)

Best suited: Decision Trees classifier.

Worst suited: kNN classifier.

Decision Trees. Decision Trees can easily handle redundant attributes. Redundant means that an attribute is strongly correlated with another attribute in the data. Since the gains of redundant attributes are almost the same in purity after they are selected for splitting, only one of them will be selected as an attribute test condition in the decision tree algorithm. For Naive Bayes Classifier, redundant and correlated attributes will violate class conditional assumption, but we may use other techniques such as Bayesian Belief Networks (BBN) to avoid this problem. The presence of irrelevant and redundant attributes can lower the performance of kNN classifier, since the the proximity measure can be overly biased toward the redundant attributes(duplicate copies) and resulting in wrong estimates of distance.

b)

Best suited: Naive Bayes classifier.

Worst suited: kNN classifier.

Naive Bayes classifier. Naive Bayes classifiers can handle missing values in the training set by ignoring the missing values of every attribute while computing its conditional probability estimates. Moreover, Naive Bayes classifier can effectively calculate the posterior probabilities for a test instance with missing values by using only the non-missing attribute values. kNN classifier cannot handle missing values in both the training and test sets, since proximity computations normally require the presence of all attributes.Decision trees can handle the missing attribute values in a number of ways, such as probabilistic split method, surrogate split method and separate class method. But compare to Naive Bayes, the way that decision trees handles the missing value is complicate.

c)

Best suited: kNN classifier.

Worst suited: Neural Networks classifier.

kNN classifier. Because kNN classifier is a lazy-learning algorithm, which means it will store all of the training samples and it will only build a classifier when a new instance needs to be classified. Unlike decision trees and Neural Networks classifiers, which are eager-learning algorithms and generalize models from the training data before receiving new input, kNN classifier requires less computation time during the training phase. In general, Neural Networks classifier take longer time in training model than that of the Naive Bayes, so the Neural Networks classifier would be the worst one in this case.

Question 6

Strength:

Decision Trees can easily handle redundant or irrelevant attributes. Redundant means that an attribute is strongly correlated with another attribute in the data. Since the gains of redundant attributes are almost the same in purity after they are selected for splitting, only one of them will be selected as an attribute test condition in the decision tree algorithm. The irrelevant attributes has little connection with the target class labels. The gains of irrelevant attributes are very small in purity and thus will be passed over by other more relevant features. So, Decision Trees can easily handle redundant or irrelevant attributes.

Weakness:

There are some attributes in the data are not discriminative by themselves but are discriminative when used in combinations, which means the individual data will only provide little information. Due to the greedy nature of the splitting criteria in decision trees, such attributes could be passed over in favor of other attributed that are less discriminating. This could result in more complex decision trees than necessary. So, decision trees perform not well when there are interaction among attributes in data.

Question 7

a)

False. The presence of noise may lead to an over-fitting model that it has good performance over the training data with noise while poor generalization performance over the testing data. From Figure 3.27 in the book, we can easily find that after adding 100 irrelevant attributes, the training error decreases along with the increasing number of nodes, but the test error almost stays the same. This means that the generalization performance of the model is bad and the model cannot represent the true distribution of the original data.

b)

False. Decision Trees can easily handle redundant attributes. Redundant means that an attribute is strongly correlated with another attribute in the data. Since the gains of redundant attributes are almost the same in purity after they are selected for splitting, only one of them will be selected as an attribute test condition in the decision tree algorithm. Since only one of X3 and X4 will be chosen as an attribute test condition, T1 will have the same performance as that of the T2.

c)

True. Both decision tree and rule-based classification can automatically perform variable selection.For decision tree, In the Hunt's algorithm, it will extend the decision tree, determine the attribute test condition for partitioning the training instances associated with a node(gain ratio, entropy) and determine the class label to be assigned to a leaf node. So this algorithm will perform variable selection and give us the optimal solution based on the loss function we chose. For rule-based classification, the algorithm starts with an empty decision list and extracts rules for each class based on the ordering specified by the class prevalence. Then it will iteratively extracts the rules using the Learn-One-Rule function and find one rule that meet the stopping criterion. So, these two classifiers would automatically perform variable selection.