COMPSCI 671D Fall 2020
Homework 1

Due 10:15 PM EST, September 7

1 Information Theory (30 points)

The entropy in information theory is also referred to as Shannon entropy. For a particular random variable X that has n possible discrete outcomes $x_1, ..., x_n$, its entropy is defined as

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i), \tag{1}$$

where $x_1, x_2, \cdots, x_n$ are possible outcomes of X, $P(X = x_i) = p_i, i = 1, 2, \cdots, n$ and $\sum_i p_i = 1$. The entropy is only a function of the distribution of possible outcomes $p_1, ..., p_n$ and not dependent on the outcome values $x_1, ..., x_n$, so we can represent the entropy of X as $H(p)$.

1.1 Assume X obeys a categorical distribution with n possible outcomes. Under what distribution will the entropy of X reach its maximum? What is the maximum? (Hint: Prof. Rudin has discussed when the entropy of X reaches its maximum when $n = 2$. No derivation needed.)

The Conditional Entropy $H(Y|X)$ describes expectation of entropy for the variable X.

$$H(Y|X) = \sum_{i=1}^{n} H(Y|X = x_i)P(X = x_i) \tag{2}$$

The mutual information is the difference between the entropy of Y and the conditional entropy of Y given X. We can also call it the information gain given X:

$$I(X, Y) = H(X) - H(X|Y). \tag{3}$$

1.2 What is the information gain (IG) between X and itself?

1.3 Prove that the information gain is non-negative, i.e. $I(X, Y) \geq 0$. (Hint: (i) You can start by proving $\sum_{i=1}^{n} p(x_i) \log \frac{p(x_i)}{q(x_i)} \geq 0$ first. You might need to know the following expression is true: $\log_2 x \leq (x - 1) \log_2 e$. (ii) Follow the definition of entropy, conditional entropy (Eq.2) and information gain (Eq.3) to represent the information gain as $-\sum_{i=1}^{n} \sum_{j=1}^{m} p(x_i, y_j) \log \frac{p(x_i)p(y_j)}{p(x_i,y_j)}$ and then go back and use hint (i) to finish off the proof).

Training Set			
Outlook	Humidity	Windy	Go fishing?
Rainy	Normal	False	Yes
Sunny	High	True	No
Rainy	Normal	True	Yes
Rainy	High	True	Yes
Sunny	High	False	Yes
Sunny	Normal	True	No
Sunny	Normal	False	Yes
Rainy	High	False	No
Rainy	Normal	False	Yes
Rainy	Normal	True	Yes
Rainy	High	False	No
Sunny	Normal	False	Yes
Rainy	High	False	Yes
Rainy	High	True	No

Table 1: Training Set for problem 1 & 2

Test Set			
Outlook	Humidity	Windy	Go fishing?
Sunny	High	True	No
Rainy	High	False	No
Sunny	Normal	False	No
Rainy	Normal	True	Yes
Sunny	High	False	No

Table 2: Test Set for problem 1 & 2

1.4 Now that you have learned the basics of machine learning from COMPSCI 671, you would like to apply this knowledge to help others. The Duke Fishing Club wants you to help them predict whether tomorrow is a good day to go fishing. They have provided you a dataset of their previous travel records, which is separated into a training set and a test set in Table 1 and Table 2. Denote the last column as label Y, and the first three columns as features $A_1, \cdots, A_3$. Please calculate the entropy of the labels $H(Y)$, and the conditional entropy of the labels with respect to each feature $H(Y|A_i)$. Calculate the information gain of each feature. Which feature should we split on first, if we are building a decision tree using the C4.5 algorithm?

2 Decision Trees (30 points)

2.1 Build a decision tree using the splitting rule from the C4.5 algorithm for the training set in Table 1. Split until all nodes are pure, or until no further splits are possible. Do this manually, do not call a previously implemented version of C4.5. Note that the C4.5 algorithm uses the Information Gain as its splitting criteria, so you can reuse results you have calculated for Problem 1.5 for the first split. If two features have the same Information Gain, please choose the one in lexographic order (i.e., choose in the order of Humidity, Outlook, Windy). If you encountered any leaf nodes that have equal probability on "Yes" and "No", predict "Yes" in your decision tree.

In your answer, please draw the decision tree you derived, and report the ***error rate*** on the ***training set*** and ***test set***. (You may find it easy to draw this on a tablet or in PowerPoint.)

2.2 Build a decision tree using the splitting rule from the C4.5 algorithm for the training set in Table 1 again. But this time, don't split if $IG < 0.04$. If this happens, simply leave the current node to be a leaf node. Again, do this manually, do not call a previously implemented version of C4.5.

In your answer, please draw the decision tree you derived, and report the ***error rate*** on the ***training set*** and ***test set***. Compared to the tree you get in 2.1, does the test performance increase or decrease? Why?

2.3 Calculate the Gini index of each feature and build a decision tree using the splitting criterion from the CART algorithm manually. Report the error rate on the training set and test set. Can you do any pruning on the tree by the cost function $\sum_{\text{leaves}_j} \sum_{x_i \in \text{leaf}_j} \mathbb{1}_{[y_i \neq \text{leaf's class}]} + C[\#\text{leaves in subtree}]$, where $C = 1$? If you can, what is the difference in training error between the original tree and the pruned tree?

3 Programming (40 points)

In this section, you will need to train several models and do experiments on the breast cancer dataset using **Python**. You need to download four files: `data_train.csv`, `data_test.csv`, `data_imbalanced_train.csv` and `data_imbalanced_test.csv` from Sakai. We have also provided a skeleton code for you to read csv file. For this problem, please upload all the code you have written to Gradescope, and also **append it to the end of your writeup**.

3.1 Variable Selection and Cross Validation

(a) Using the skeleton code as a starting point, write the code to read the dataset from `data_train.csv` and `data_test.csv`, and train different models on the training set. Use 5-fold cross validation on the training set only, and compare accuracy and time cost performance of three different algorithms: ID3 (which is extremely similar to C4.5), CART and Random Forest, using `sklearn.tree.DecisionTreeClassifier` and `sklearn.ensemble.RandomForestClassifier`. (You will train 15 models through this process. Note that for these programming problems you do not implement the methods yourself, you will use a package.) Notice that ID3 and CART use entropy and gini index as the splitting criterion respectively. For the forest, you could set the number of estimators to 50 or more and use 'GINI' splitting method. No other hyperparameter tuning is needed here, you can simply use the default parameters. Please turn in the code and the results. Besides the average accuracy, you should also report the standard deviation, and do pairwise t-tests to determine whether there's a significant difference between the best algorithm and the other algorithms. You may find `sklearn.model_selection.KFold` and `scipy.stats.ttest_ind` useful here.

(b) This time, we will use 5-fold cross validation to choose a parameter value. We want to pick a value for the hyperparameter `max_depth` for the Random Forest classifier. Possible candidates for `max_depth` range from 1 to 10. Run 5-fold cross validation on the training set, and report the average accuracy for each `max_depth`. Write out each step and how you divided the dataset into training and validation folds. (You will train 50 random forests in this process.)

Remember never to use the test set for tuning parameters! After you pick the best parameter, train a model on the full training set using that parameter, and report its performance on the test set.

(c) In this question, we will investigate how to use ROC curves and AUC to evaluate the quality of each of our variables. Draw an ROC curve for each of the **features** in the dataset, and put them all on the same plot. (You can use a feature itself as a prediction function and generate the ROC curve according to the features' scores.) You can generate the ROC curves with the helper function provided in the skeleton code we provided. Report the AUC for each of the features in your answer. Do you think some features might be more useful than others?

(d) Instead of the ranking performance on the full range of scores of a classifier, sometimes we may care about the ranking performance on a small range of scores. This is usually evaluated by partial AUC. If we think of the ROC curve as a function of the False Positive rate (denoted as x below), then we have

$$AUC = \int_0^1 ROC(x)dx$$

and similarly,

$$PartialAUC = \frac{\int_{t_0}^{t_1} ROC(x)dx}{t_1 - t_0},$$

where the "x" in the above expression is the false positive rate.

If we choose t_0 to be 0, we are evaluating the quality of the highest scoring observations; for instance, these are the most vulnerable medical patients (using health data) or the equipment that is most likely to fail (using equipment failure data). Calculate the partial AUC for the **first five features** in the range of $(0, 0.2)$.

(e) Run CART on the training set, and calculate the model reliance of the CART model on all of the features it uses. To calculate CART's model reliance for a variable, simply randomly scramble the column corresponding to that variable, and report by what fraction the loss increases.

3.2 Imbalanced Dataset

The dataset we provided in Problem 3.1 is fairly balanced. What if the data is imbalanced? In this problem, we will investigate how class imbalance impacts the performance of various models.

(a) Read the dataset from `data_imbalanced_train.csv` and `data_imbalanced_test.csv`. What is the imbalance ratio?

(b) Class imbalance sometimes causes problems. One of the most popular approaches to deal with class imbalance is to weigh the loss for positives and negatives differently, so that the less common category will receive more attention from the model. Again, we will use the three algorithms from Problem 3.1 to train models on the training set. Please report the confusion matrix on the test set for all three algorithms. (Do not tune parameters.)

(c) While fixing the weight on the more common category to be 1, gradually increase the weight on the less common category from 1 to 20 by steps of 1. Set the maximum depth of all the three algorithms (ID3, CART, RandomForest) to 3 in order to avoid overfitting in all of these experiments. Now, for each algorithm, keep track of 40 values as you increase the weight: 20 training accuracy values for positive points, and 20 training accuracy values for negative points. Plot these accuracies as a function of weights on the two categories on the training set and testing set – create three plots, one for each algorithm, where you plot positive and negative accuracies on the same plot. Please remember to label the axes of your plot and title each plot. You may find `matplotlib.pyplot` package useful for this task.