Homework 2

CSC522: Automated Learning and Data Analysis
Dr. Thomas Price

Spring 2019

Instructions

- **Due Date:** Feb 19, 2019
- **Total Points**: 100
- Make sure you clearly list each team member's names and Unity IDs at the top of your submission.
- Your submission should be a single zip file containing a PDF of your answers, your code, and a README file with running instructions. Please follow the naming convention for your zip file: G(homework group number)_HW(homework number), e.g. G1_HW2.
- If any question is unclear, please post a question on Piazza. If you make any assumptions in your answer, you must state them explicitly (e.g. "Assuming the data in question is normally distributed...").
- If a question asks you to explain or justify your answer, **give a brief explanation** using your own ideas, not a reference to the textbook or an online source.
- If this homework needs to be updated for any reason, **we will post the update on Piazza** and announce it in class, so please stay alert.
- In addition to your group submission, please also *individually* submit your Peer Evaluation form on Moodle, evaluating yours and your teammates' contributions to this homework.
- **Update:** While you will submit *one* homework as a team, you are responsible for *all* of the content on the homework. We recommend attempting to solve each question individually.

R Programming Submission Instructions

- Make sure you clearly list each team member's names and Unity IDs at the top of your submission.
- Your code should be named $hw2.R$. Add this file, along with a README to the zip file mentioned in the first page.
- Failure to follow naming conventions or programming related instructions specified below may result in your submission not being graded.
- Carefully read what the function names have been requested by the instructor. **In this homework or the following ones, if your code does not follow the naming format requested by the instructor, you will not receive credit.**
- For each function, both the input and output formats are provided in the $hw2.R$. Function calls are specified in $hw2\_checker.R$. Please ensure that you follow the correct input and output formats. Once again, if you do not follow the format requested, you will not receive credit. It is clearly stated which functions need to be implemented by you in the comments in hw2.R
- You are free to write your own functions to handle sub-tasks, but the TA will only call the functions he has requested. If the requested functions do not run/return the correct values/do not finish running in specified time, you will not receive full credit.

- DO NOT set working directory (setwd function) or clear memory (rm(list=ls(all=T))) in your code. TA(s) will do so in their own auto grader.
- The TA will have an autograder which will first run `source(hw2.R)`, then call each of the functions requested in the homework and compare with the correct solution.
- Your code should be clearly documented.
- To **test you code**, step through the `hw2_checker.R` file. If you update you code, make sure to run `source('./hw2.R')` again to update your function definitions. You can also check the "Source on save" option in R Studio to do this automatically on save.
- **Update:** Please DO NOT include `install.packages()` in your `hw2.R`. Please note, we are specifying the allowed packages, which means that we already have them installed on our test machine. From HW2, using `install.packages` in your code would result in a penalty of 5 points.

Problems

1. Decision Tree Construction (20 points) [**Song Ju**]. Create decision trees for the `hw2q1.csv` dataset by hand, as explained below, using Hunt's algorithm. Note the following:

   - In the given dataset, all of the input attributes are binary except for the first attribute, V1, which is ratio and continuous.
   - The output label has two class values: T or F.
   - In the case of ties then selecting an attribute, break ties in favor of the leftmost attribute.
   - When considering a split for the continuous V1 attribute, identify the best value to split on (e.g. ≤ 15 and > 15) by testing all possible split values.

   When calculating Information Gain or Gini Index, write down each step of the computation process and show your work step by step. Draw a separate tree after each attribute split, like the example in Figure 1. You can use a program (e.g. tikz with LaTeX) to draw your trees, or draw them by hand on paper and scan your results into the final pdf.

   (a) Construct the decision tree manually, using Information Gain (IG) to select the best attribute to split on, as in the ID3 algorithm. The maximum depth of your tree should be 4 (count the root node as depth 0), meaning that any node at depth 4 will be a leaf node.

   (b) Construct the tree manually using the Gini index. The maximum depth of the tree should be 2.

   (c) How are the trees different? As part of your explanation, give 2 examples of data objects that would be classified differently by the two trees.

   (d) Which decision tree will perform better on the training dataset (`hw2q1.csv`)? Which will perform better on a test dataset? Can we know the answer?

   Solution: For (a) and (b), please refer to "hw2q1_solution.pdf"

   (c) Different metrics (IG or Gini) can generate different tree structures. For the first split, IG shows 11 and 42 are the best split in V1. But GINI shows 27 is the best.

   (d) IG tree will perform better in the training dataset because Gini tree has more misclassified instance. But in test dataset, we don't know which one is better.

2. Evaluation Measures (10 points) [**Song Ju**] Given the decision tree in Figure 1, complete the following tasks:

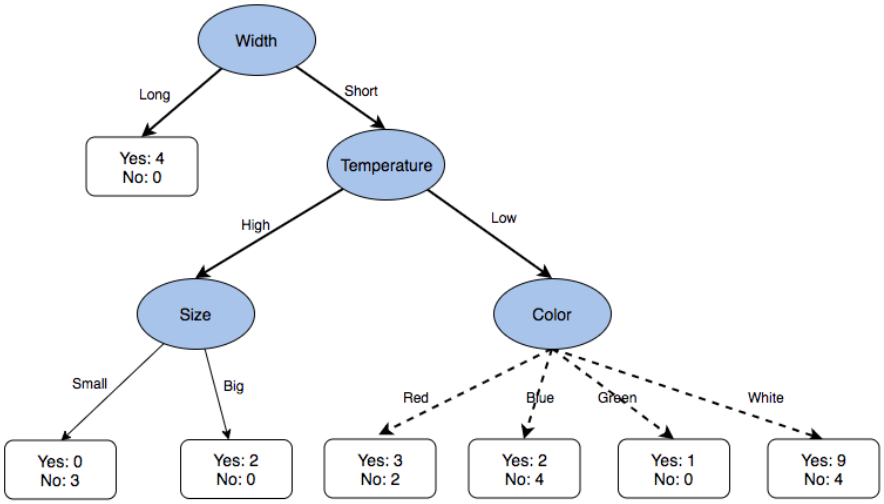

Figure 1: Decision Tree

(a) Compute the training classification error for the classifier using both optimistic error and pessimistic error. For pessimistic training error, use a penalty of 0.5 per leaf node. You may assume that the whole training dataset is represented by the class labels in the leaf nodes of the tree in Figure 1, with a total of 34 training instances.

(b) Use the decision tree above to classify the provided dataset. `hw2q2.csv`. Construct a confusion matrix and report the test Accuracy, Error Rate, Precision, Recall, and F1 score. Use "Yes" as the positive class in the confusion matrix.

Solution: (a)
The optimistic error is $(2+2+4)/34 = 8/34 = 0.235$.
The pessimistic error is $(8+7*0.5)/34 = 11.5/34 = 0.338$.

(b)

	Predicted Yes	**Predicted No**	**Total**
Actual Yes	12	2	14
Actual No	4	2	6
Total	16	4	20

$Accuracy = 14/20 = 0.7$
$ErrorRate = 6/20 = 0.3$
$Precision = 12/16 = 0.75$
$Recall = 12/14 = 0.875$
$F1measure = 12/15 = 0.8$

3. Decision Tree Pruning (10 points) [**Ruth Okoilu**]. Consider the decision tree in Figure 1. We will focus on the sub-tree which splits on the attribute "Color". Answer the following questions and show your work.

(a) Calculate the optimistic training classification error before splitting and after splitting using Color, respectively. If we want to minimize the optimistic error rate, should the node be pruned?

(b) Calculate the pessimistic training errors before splitting and after splitting using Color respectively. When calculating pessimistic errors, each leaf node will add a factor of 0.8 to the error. If we want to minimize the pessimistic error rate, should the node be pruned?

(c) Assuming that the "Color" node is pruned, recalculate the test Error Rate using `hw2q2.csv`. Based on your evaluation using the test dataset in `hw2q2.csv`, was the original tree (with the Color node) over-fitting? Why or why not?

Solution:

(a) Optimistic training error before splitting = 10/34
Optimistic training error after splitting = 8/34
The optimistic error decreases after splitting, so we should not prune this branch. The resulting tree is shown below:

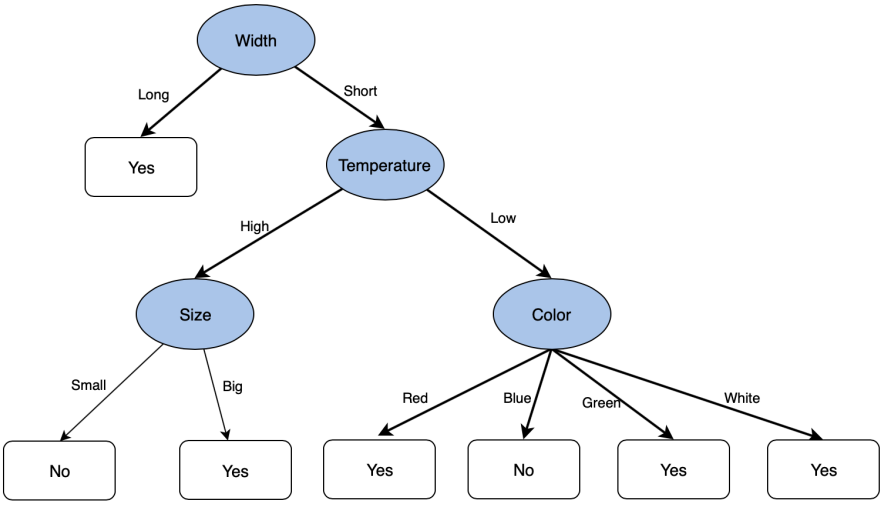

Figure 2: Decision Tree

(b) Pessimistic training error before splitting =(10+4*0.8) /34 = 13.2
Pessimistic training error after splitting
=(8+7*0.8)/34 = 13.6
The pessimistic error increases after splitting, so we should prune this branch. The resulting tree is shown below:

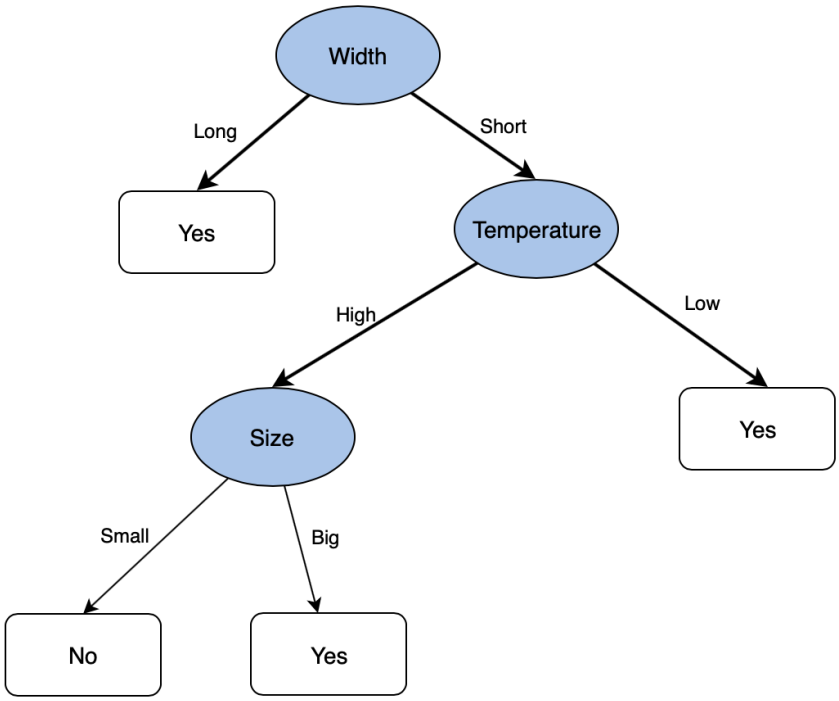

Figure 3: Decision Tree

See Figure 3 for resulting tree after pruning Color node.

(c) Test Error rate before splitting the tree: $7/20 = 0.35$
Test Error rate after splitting the tree: $6/20 = 0.3$
The test error rate after splitting is lower than that before splitting. Therefore the original tree was not over-fitting. Over-fitting occurs when the tree is designed so as to perfectly fit all samples in the training data set such that it does not fit the test set, leading to high test error rate. As we increase the depth of the tree, if the test error increases and this causes over-fitting.

4. 1-NN, Evaluation, Cross Validation (15 points) [**Ruth Okoilu**] Consider the following dataset (9 instances) with 2 continuous attributes (x_1 and x_2) and a class attribute y, shown in Table 1. For this question, we will consider a 1-Nearest-Neighbor (1-NN) classifier that uses euclidean distance.

Table 1: 1-NN

ID	x_1	x_1	**y**
1	35.0	15.0	-
2	2.5	11.0	-
3	10.5	12.5	+
4	44.0	11.0	+
5	1.5	13.0	-
6	48.0	11.0	+
7	45.0	13.0	-
8	38.0	10.0	+
9	7.5	13.5	-

(a) Calculate the distance matrix for the dataset using euclidean distance.

(b) By hand, evaluate the 1-NN classifier, calculating the confusion matrix and testing accuracy (show work). Use the following evaluation methods:

i. A holdout test dataset consisting of last 4 instances

ii. 3-fold cross-validation, using the following folds with IDs: [3,6,9], [1,4,7], [2,5,8] respectively.

iii. Leave one out cross validation (LOOCV)

(c) For a data analysis homework, you are asked to perform an experiment with a binary classification algorithm. You are given a dataset with 20 instances and a class attribute that can be either Positive or Negative. The dataset includes 10 positive and 10 negative instances, each with d attributes. You decide to use LOOCV (i.e 20 singleton test sets) to evaluate the algorithm. As a baseline, you compare your algorithm to a "simple majority classifier," which always predicts the majority class in the training dataset (if there is no majority, one of the classes is chosen at random). You expect the test accuracy of the simple majority classifier to be about 50% using LOOCV. Instead it performs differently. How does it perform and why?

Solution:

(a) See distance matrix in Table 2

	1	2	3	4	5	6	7	8	9
1	0.00	32.75	24.63	9.85	33.56	13.60	10.20	5.83	27.54
2	32.75	0.00	8.14	41.50	2.24	45.50	42.55	35.51	5.59
3	24.63	8.14	0.00	33.53	9.01	37.53	34.50	27.61	3.16
4	9.85	41.50	33.53	0.00	42.55	4.00	2.24	6.08	36.59
5	33.56	2.24	9.01	42.55	0.00	46.54	43.50	36.62	6.02
6	13.60	45.50	37.53	4.00	46.54	0.00	3.61	10.05	40.58
7	10.20	42.55	34.50	2.24	43.50	3.61	0.00	7.62	37.50
8	5.83	35.51	27.61	6.08	36.62	10.05	7.62	0.00	30.70
9	27.54	5.59	3.16	36.59	6.02	40.58	37.50	30.70	0.00

Table 2: Distance Matrix

Table 3: 1-NN

ID	x_1	x_1	y	**1-NN**	**Pred**	**Error**
1	35.0	15.0	-	8	+	1
2	2.5	11.0	-	5	-	0
3	10.5	12.5	+	9	-	1
4	44.0	11.0	+	7	-	1
5	1.5	13.0	-	2	-	0
6	48.0	11.0	+	7	-	1
7	45.0	13.0	-	4	+	1
8	38.0	10.0	+	1	-	1
9	7.5	13.5	-	3	+	1

Table 4: 1-NN using Hold out

	Predicted (+)	**Predicted (-)**
Actual (+)	1	1
Actual (-)	2	0

(b) i. Accuracy = 1/4

ii. Fold 1 contains IDs 3,6,9 Fold 2 contains IDs 1,4,7 Fold 3 contains IDs 2,5,8

Table 5: 1-NN using 3 Fold CV

	Predicted (+)	**Predicted (-)**
Actual (+)	1	3
Actual (-)	2	3

Accuracy = 4/9

Table 6: 1-NN using LOOCV

	Predicted (+)	**Predicted (-)**
Actual (+)	0	4
Actual (-)	3	2

iii. Accuracy = 2/9

(c) For each run, the majority label of the training data will be different from the label of the validation data instance. For example, if the validation data instance is negative, the majority label of the training data (20 positives and 19 negatives) will be positive. Therefore, the accuracy will always be 0% for all runs.

5. R Programming Part 1: KNN and Decision Tree Classifiers (45 points) [**Krishna Gadiraju**]

You are given the following files:

- `hw2_train.csv`: CSV file with 100 rows, 101 columns. The first 100 columns (named V1:V100) refer to your features, final column (called 'Class') refers to the Class variable. You have 4 classes: {1, 2, 3, 4}. This is your training dataset
- `hw2_test.csv`: This is your test dataset and follows the same format as your training dataset. It has 50 rows.
- Data was generated by computing the TF-IDF matrix of a corpus of news papers. 100 top words from several documents were extracted (i.e., feature extraction was already done for you), and the TF-IDF matrix was calculated for these words. A similar transformation was applied to your test dataset as well.

Please note that this exercise has sections where you need to implement methods, and sections where you can use a library. Whether you need to implement/use library is clearly stated:

(a) **Part A: Distance Computation (similar to HW1)**: Recall the `calculate_matrix`, `calculate_euclidean` and `calculate_cosine` methods from HW1. Once again, `calculate_matrix` implementation has already been provided to you, that takes the training data matrix, and test data matrix and computes pairwise distances between every row (sentence/document) in test data matrix and train data matrix. `calculate_euclidean` and `calculate_cosine` follow the same definitions as in HW1. Copy your solutions from HW1 for these two methods in `hw2.R` (make sure to correct them if you had any errors in HW1). We will use these methods to calculate the distance matrix for the KNN algorithm.

(b) **Part B: Vanilla KNN Classification: (Implement)** You are tasked with implementing the KNN algorithm using the dataset provided to you. You will be implementing the KNN classifier in `hw2.R` using the `knn_classifier` method. The input and output variables, and their formats are explained in detail in the form of comments inside the `knn_classifier` function. Read every comment carefully before implementing your method.

i. **Part B-2: KNN using confidence (Implement)**: Read the paper titled "A simple KNN algorithm for text categorization" [1][1]. The authors present a simple solution to text categorization using KNN algorithm. While the primary contribution of the paper is on the feature selection methods used to reduce computational complexity of classifying a large corpus of text data, we will focus on the measure they used to compute the nearest neighbors. [1] defines a confidence measure on top of the similarity measure to calculate the nearest neighbors. While the standard procedure of KNN algorithm (assuming data is normalized and feature selection, if needed is completed) is:

- Step 1: for a test instance, calculate distance to all training instances
- Step 2: Find the k nearest training instances
- Step 3: Choose class based on majority count in k nearest training instances

[1] You can find the paper by searching `scholar.google.com` for the title.

the algorithm in [1] modifies Step 3. Instead of taking the majority class of the K nearest neighbors, [1] weights each neighbor according to its cosine similarity and then choose the class with most total weight among the K nearest neighbors. This weighted score between the test instance and each class is called confidence. In other words, for a test instance t, considering $k_1, k_2, \ldots, k_K$ are its K nearest neighbors, and k_2, k_4, k_5 belong to class c, $Confidence(t, c) = \frac{sum(cosine\ similarities\ of\ t\ and \{k_2,\ k_4,\ k_5\})}{sum(cosine\ similarities\ of\ t\ and \{k_1\ to\ k_k\})}$. The final class label belongs to the class with maximum confidence. Implement this functionality in the method `knn_classifier_confidence`. The input, output formats, as well as allowed packages are described as comments under the function.

(c) **Part C: Decision Tree using Cross Validation (Library):**

i. Using the training dataset `hw2_train.csv`, build a CART decision tree using the `rpart` library and gini index for the splitting criterion. Then using this model, classify the outcomes from the test dataset `hw2_test.csv`. Code for this question has to be written in the function `dtree`.

ii. Use the `caret` library to tune the complexity parameter of the CART classifier, using n-fold cross validation on the outcomes *training* dataset. Then using this model, classify the outcomes from the test dataset `hw2_test.csv`. Code for this question has to be written in the function `dtree_cv`.

The input, output formats, as well as allowed packages are described as comments under the function.

(d) **Part D: Confusion Matrix and Accuracy (Library):** Write a function `calculate_accuracy` that takes the predicted class labels of type vector (factor), ground truth labels of type vector (factor) and returns a list, with the first element as the confusion matrix, and second element as the overall accuracy. The input, output formats, as well as allowed packages are described as comments under the function. An example of how the confusion matrix should be represented is shown in Figure 4. `Prediction` refers to the predicted values, `Reference` refers to the ground truth values.

(e) **Part E: Analysis:** Report these answers in your pdf. Compare the performance of the KNN Classifer using euclidean distance, KNN Classifier using cosine similarity, KNN classifier using confidence and and Decision Tree classifiers (with and without hyperparameter tuning). At the minimum, we expect to see the following comparisons. You are welcome to perform a more in-depth analysis in addition to these analyses.

- Comparison in terms of overall accuracy - which classifier performed best?
- Comparison in terms of confusion matrix - In each classifier, which class had the maximum misclassifications? Did one class perform better than the other in these cases?

```
            Reference
Prediction 1 2 3 4
         1 3 2 0 1
         2 2 3 1 1
         3 2 1 1 1
         4 2 1 2 2
```

Figure 4: Sample Confusion Matrix generated with random data

References

[1] P. Soucy and G. W. Mineau, "A simple knn algorithm for text categorization," in *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on.* IEEE, 2001, pp. 647–648.