Report

Ahmet Dara Vefa

January 9, 2021

1 Part 1: Decision Tree

Quick note: the "empty" leafs are set to the majority class of the parent node. You can see the output images and text(?) files in the "DecisionTreeOutputs" folder.

1.1 Information Gain

infoGain tree accuracy= 0.9525462962962963
Safety is the root of the tree, mostly because it has a direct unacc leaf when safety=low. Persons is the second node after safety because it also has unacc when persons=2. But since the total number of unacc's is smaller in persons=2, it is not selected as root.

1.2 Gain Ratio

gainRatio tree accuracy= 0.9560185185185185
Majority of the tree is the same with info gain(hence the similar accuracies) but the left side of the tree has a tiny bit of difference at the lugboot and maint nodes. This should be because the distribution of each attribute's value's are balanced in the tree.

1.3 Average Gini Index

averageGiniIndex tree accuracy= 0.9525462962962963
Totally same tree with the gain ratio tree. Just the minimizing or maximizing is different in the formula.

1.4 Gain Ratio with Chi-squared Pre-pruning

gain ratio pre pruned tree accuracy= 0.9502314814814815
A lot smaller tree than the gain ratio tree. I guess we traded computation time with a tiny bit of accuracy. Though, I would have expected a higher accuracy since the tree becomes more generalized.

1.5 Gain Ratio with Reduced Error Post-pruning

gain ratio post pruned tree accuracy= 0.9513888888888888
Still a large tree but it is more generalized since there are less leafs in the deepest depth. The accuracy is better than pre pruned tree even though in this tree I used 80 percent of the data to train, 20 percent to validate.(accuracy is still calculated with the same test data)

2 Part 2: Support Vector Machine

You can see the relevant outputs in the "SVMOutputs" folder.

2.1 First Part

As expected, increasing the C value makes the SVM more hard bound(at C=infinity, it is a hard bound svm). C=0.01 is too small and makes the svm too general, i.e useless. At C=1 svm becomes a really nice classifier for our data. At C=10 svm becomes too hard bound and is not as good as C=1 since it does not "generalize" well.

2.2 Second Part

Since this is not a linearly classifiable data linear kernel fails.
Sigmoid kernel also does its best but it still is awful.
rbf kernel does an awsome job, correctly classifies the data with the least support vectors.
Polynomial kernal is semi-decent for the squares but it wrongly classifies half of the circles and it has a lot of support vectors. Not good overall

2.3 Third Part

You can find the exact output values in the "part3 output.txt" file.

Also note that some tables have moved because of pdf magic.

From the outputs we can see that the best accuracy with 5 fold cross validation on train data is 0.7322158830087219. We achieve this accuracy with polynomial kernel with (gamma=0.001 and C=10) and (gamma=0.01 C=0.01). I will use the second option as the gamma and C values are proportionally smaller in the second option, which gives us a more generalized classifier.

Once we train the classifier on train data with select parameters, accuracy for test data is:
accuracy with poly kernel at 0.01 gamma and 0.01 cValue for test data = 0.7956989247311828

gamma	C				
	0.01	0.1	1	10	100
-	0.64	0.67	0.70	0.70	0.70

Table 1: Linear kernel

gamma	C				
	0.01	0.1	1	10	100
0.00001	0.53	0.53	0.53	0.60	0.64
0.0001	0.53	0.53	0.61	0.66	0.71
0.001	0.53	0.58	0.67	0.72	0.73
0.01	0.53	0.53	0.71	0.70	0.70
0.1	0.53	0.53	0.70	0.70	0.70
1	0.53	0.53	0.70	0.70	0.70

Table 2: RBF kernel

2.4 Fourth part

Confusion matrices can be seen in the "SVMOutputs/Task4" folder.

2.4.1 Without handling the imbalance problem

accuracy with rbf kernel at 1 cValue for test data = 0.8333333333333334
No, accuracy can't be a good metric since the data is massively imbalanced. Confusion matrix confirms this since the diagonal of the matrix(i,i'th cell's) is not around the same value/color. Precision would be a better alternative(probably not the best alternative) as a performance metric

2.4.2 Oversampling the minority class

(oversampled) accuracy with rbf kernel at 1 cValue for test data = 0.7964912280701755
The accuracy is lower but it is a "better" metric than before. The balance of the diagonal cells is better but the accuracy is lower. I am guessing that is because the test data is also unbalanced towards label=1

2.4.3 Undersampling the majority class

(undersampled) accuracy with rbf kernel at 1 cValue for test data = 0.5842105263157895
Undersampling is bad for this case because both the training data and the test data is unbalanced towards the label=1. So decreasing the data of the majority class is bad when real life data is also unbalanced towards the same label.

2.4.4 Setting the class_weight to balanced

(classweighted) accuracy with rbf kernel at 1 cValue for test data = 0.7570175438596491
This one is like oversampling but it is less reliable when true label=1 while also

gamma	C				
	0.01	0.1	1	10	100
0.00001	0.53	0.53	0.53	0.53	0.53
0.0001	0.53	0.53	0.53	0.53	0.54
0.001	0.53	0.54	0.65	0.73	0.72
0.01	0.73	0.72	0.72	0.72	0.72
0.1	0.72	0.72	0.72	0.72	0.72
1	0.72	0.72	0.72	0.72	0.72

Table 3: Polynomial kernel

gamma	C				
	0.01	0.1	1	10	100
0.00001	0.53	0.53	053.	0.59	0.62
0.0001	0.53	0.53	0.59	0.62	0.64
0.001	0.53	0.59	0.55	0.53	0.52
0.01	0.55	0.52	0.51	0.50	0.50
0.1	0.55	0.52	0.52	0.52	0.52
1	0.54	0.51	0.51	0.51	0.51

Table 4: Sigmoid kernel

being more reliable when true label=0.